МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет України ``Київський політехнічний інститут імені Ігоря Сікорського"

Кваліфікаційна наукова
праця на правах рукопису

ЗЕЛЕНСЬКИЙ КИРИЛО ХАРИТОНОВИЧ

УДК [678.027.3+678.057.3]:678.073(043.3)

ДИСЕРТАЦІЯ

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ НЕЛІНІЙНИХ ПОЛІМЕРНИХ МАТЕРІАЛІВ В ЕКСТРУДЕРАХ

Спеціальність 01.05.02 – Математичне моделювання та обчислювальні методи

Подається на здобуття наукового ступеню доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

К.Х. Зеленський

Науковий консультант: Трофимчук О.М.,
доктор технічних наук, професор, Член-кореспондент НАН України.

Київ – 2021

АНОТАЦІЯ

Зеленський К.Х. Математичне моделювання нелінійних полімерних матеріалів в екструдерах – кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеню доктора технічних наук за спеціальністю 01.05.02 – математичне моделювання та обчислювальні методи -- Національний технічний університет України ``Київський політехнічний інститут імені Ігоря Сікорського" МОН України, Київ, 2021.

Дисертаційна робота присвячена вирішенню науково-технічної проблеми розробки методів математичного і комп'ютерного моделювання динамічних процесів в об'єктах із розподіленими параметрами, що описуються системами нелінійних і квазілінійних диференційних рівнянь у частинних похідних параболічного типу математичної фізики із урахуванням нелінійних властивостей теплофізичних і реологічних характеристик процесів.

Вирішення цієї проблеми сприяє удосконаленню існуючих конструкцій екструзійного устаткування та проектуванню нового екструзійного обладнання для виготовлення виробів із полімерних матеріалів, зокрема, кабелів на надвисокі напруги із полімерним ізоляційним покриттям.

Використання результатів роботи надає можливість суттєво скоротити терміни і вартість проектування нового обладнання за рахунок математичного і комп'ютерного моделювання процесів в екструдерах замість тривалих і коштовних експериментальних досліджень з впливу нелінійних реологічних і теплофізичних характеристик полімерних матеріалів на процеси переробки полімерних матеріалів з метою виготовлення полімерної продукції і автоматизувати процеси проектування нового екструзійного обладнання.

Проведений аналіз сучасного стану процесів тепло і масоперенесення в екструдерах засвідчив, що підходи до аналізу особливостей нагрівання полімерної суміші у зоні завантаження не враховують процес індукційного

нагріву корпусу екструдера і, природно, не враховують наявність променистого теплообміну між індуктором та корпусом екструдера, що є суттєвим чинником для визначення кількісної характеристики ступеню нагріву корпусу з точки зору його оптимального значення (перегрівання або навпаки). Цей фактор суттєво впливає на геометричні параметри зони завантаження. Крім того, не враховується залежність коефіцієнту тепломісткості від температури нагріву, що знову ж таки суттєво впливає на протяжність цієї зони порівняно із розрахунковим значенням при проектуванні екструдеру.

У значній кількості наукових праць, в яких формулюються математичні моделі процесів у зонах плавлення та дозування розплаву полімерів, що є спрощеними (наприклад, використовуються одновимірні відносно просторових координат моделі, або стаціонарні двовимірні моделі), оскільки ніяк не враховують процеси конвективного перенесення у зонах плавлення і дозування, а також нелінійні властивості параметрів, у тому числі, реологічних. Наявність двохфазних зон лише декларується і зовсім не враховується при комп'ютерному моделюванні цих моделей із застосуванням або різницевих схем, або із застосуванням методів скінченних елементів. Більше того, результати такого моделювання наводяться в одновимірній постановці, що свідчить про спрощений підхід до моделювання цих процесів.

Отже, для коректного опису процесів нагрівання полімерної суміші та її плавлення і подальшої кристалізації необхідно будувати математичні моделі, що описуються рівняннями або системами (у випадку плавлення та кристалізації) нелінійних диференційних рівнянь у частинних похідних математичної фізики із відповідними лінійними або нелінійними межовими умовами.

Існуючі підходи до розробки таких систем, як правило, ґрунтуються на використанні лінійних математичних моделей для опису динаміки об'єктів. Такий підхід до побудови моделей не враховує найсуттєвіші властивості

об'єктів за рахунок або ігнорування нелінійних складових або їхньої лінеаризації. Ця проблема особливо актуальна стосовно процесів із розподіленими параметрами, математичні моделі яких описуються рівняннями із частинними похідними із відповідними додатковими умовами на межі області.

В останні десятиліття об'єктам із розподіленими параметрами приділяється велика увага. Це пояснюється бажанням підвищити ефективність математичного моделювання динаміки таких об'єктів і процесів та розробки систем автоматичного й автоматизованого управління, оскільки добре відомо, що всі фізичні процеси і об'єкти за своєю сутністю є об'єкти із розподіленими параметрами. Але переважна більшість моделей, що описують динаміку об'єктів із розподіленими параметрами є лінійні моделі, хоча добре відомо, що реальні фізичні процеси є за своєю сутністю нелінійні.

Виходячи з цього, виконання математичного моделювання нелінійних процесів є актуальна наукова проблема. Математичне моделювання нелінійних процесів із використанням сучасної обчислювальної техніки надає можливість глибшого і достовірного вивчення цих процесів, суттєвої економії витрат, пов'язаних із традиційним фізичним моделюванням процесів, створення систем автоматичного й автоматизованого управління, що адекватні реальним об'єктам.

Для отримання розв'язків відповідних крайових задач, що описуються системами нелінійних диференційних рівнянь у частинних похідних параболічного типу, автором розроблено числово-аналітичний метод, що ґрунтується на застосуванні скінченних інтегральних перетворень. Застосування цього методу до розв'язання нелінійних крайових задач ґрунтується на апроксимації циліндричних функцій (функцій Неймана) дробово-раціональними функціями із використанням апарату ланцюгових дробів, а також на наближеному поданні конвективних складових у відповідних рівняннях дробово-раціональними функціями. Розроблено алгоритми інтегрування

добутків кількох циліндричних функцій, що ґрунтуються на поданні цих функцій дробово-раціональними виразами.

Розроблено відповідне алгоритмічне та програмне забезпечення, яке надає можливість автоматизувати процес розв'язання цих задач і визначати оптимальні значення реологічних і теплофізичних параметрів об'єкту дослідження.

Досліджено вплив питомої потужності індуктора на довжину зони завантаження із урахуванням променистого випромінювання на межі індуктор -- корпус екструдера. Показано, що визначальним фактором, який впливає на якість кінцевого продукту, є швидкість руху межі фазового переходу тверда суміш – розплав полімеру, яка має дорівнювати радіальній компоненті швидкості обертання шнеку, щоб забезпечити розрахункові геометричні параметри зон екструдера.

Оскільки змінювання швидкості обертання шнеку призводить до змінювання конструктивних характеристик екструдера, слушно управляти процесами тепло і масоперенесення у ньому за рахунок змінювання питомої потужності індуктора у кожній зоні, що нагріває корпус екструдера.

Виконано математичне моделювання процесів гомогенізації та кристалізації розплаву полімеру, що ґрунтується на кінетичних співвідношеннях між гомогенізацією та кристалізацією.

Розроблені у дисертаційній роботі методи математичного і комп'ютерного моделювання процесів масо- і теплоперенесення у полімерах із урахуванням нелінійних властивостей надали можливість суттєво підвищити якість моделювання, розробити рекомендації щодо проектування й удосконалення технологічних процесів з виготовлення кабелів на надвисокі напруги.

Наукова новизна роботи полягає у тому, що вперше запропоновано метод числово-аналітичного розв'язання нелінійних диференційних рівнянь у частинних похідних параболічного типу і його застосування до розв'язання

задач нагріву корпусу екструдера, нагріву полімерної суміші у зоні завантаження, зоні плавлення полімеру і гомогенізації та кристалізації розплаву полімеру у зоні дозування. Виконані дослідження надають можливість забезпечити підтримання розрахункових параметрів екструдера при виготовленні полімерних виробів (ізоляційне покриття кабелів на надвисокі напруги, виготовлення полімерних плівок широкого асортименту тощо), а також розрахунку оптимальних розмірів вказаних зон при проектуванні екструзійного обладнання, призначеного для виготовлення широкого кола продукції із застосуванням екструзійних пристроїв.

Ключові слова: екструдер, задача типу Стефана, інтегральні перетворення, ітераційна процедура, числово-аналітичний метод, фазовий перехід, функції Бесселя, рівняння Нав'є--Стокса, шнек, черв'як.

ABSTRACT

Zelensky K.Kh. Mathematical modeling of nonlinear polymeric materials in extruders - qualifying scientific work on the rights of the manuscript.

The dissertation on competition of a scientific degree of the doctor of technical sciences on a specialty 01.05.02 - mathematical modeling and computational methods - National technical university of Ukraine `` Igor Sikorsky Kyiv Polytechnic Institute " MES of Ukraine, Kiev, 2021.

The dissertation is devoted to the solution of a scientific and technical problem of development of methods of mathematical and computer modeling of dynamic processes in objects with the distributed parameters described by systems of nonlinear and quasilinear differential equations in partial derivatives of parabolic type of mathematical physics taking into account nonlinear properties. .

Solving this problem contributes to the improvement of existing extrusion equipment designs and the design of new extrusion equipment for the manufacture of products from polymeric materials, in particular, ultra-high voltage cables with polymer insulation coating.

The use of the results makes it possible to significantly reduce the time and cost of designing new equipment through mathematical and computer modeling of processes in extruders instead of long and expensive experimental studies on the influence of nonlinear rheological and thermophysical characteristics of polymeric materials on polymer processing and manufacturing of polymer products. automate the design process of new extrusion equipment.

The analysis of the current state of heat and mass transfer processes in extruders showed that the approaches to the analysis of heating and melting in the loading zones of the polymer mixture and the melting and dosing zones of the polymer melt are simplified (for example, one-dimensional relative to spatial coordinates, or stationary for two-dimensional models). because they do not take into

account the processes of convective transfer in the melting and dosing zones, as well as nonlinear properties of parameters, including rheological.

Existing approaches to the development of such systems are usually based on the use of linear mathematical models to describe the dynamics of objects. This approach to building models does not take into account the most significant properties of objects by either ignoring nonlinear components or their linearization. This problem is especially relevant for processes with distributed parameters, the mathematical models of which are described by partial differential equations with corresponding additional conditions at the boundary of the domain. In recent decades, much attention has been paid to objects with distributed parameters. This is due to the desire to increase the efficiency of mathematical modeling of the dynamics of such objects and processes and the development of automatic and automated control systems, as it is well known that all physical processes and objects are essentially objects with distributed parameters. But the vast majority of models describing the dynamics of objects with distributed parameters are linear models, although it is well known that real physical processes are inherently nonlinear.

Based on this, the implementation of mathematical modeling of nonlinear processes is an urgent scientific problem. Mathematical modeling of nonlinear processes using modern computer technology provides an opportunity for deeper and more reliable study of these processes, significant cost savings associated with traditional physical modeling of processes, the creation of automatic and automated control systems that are adequate to real objects.

The results of the analysis of existing approaches to the mathematical description of processes in extruders for processing polymeric materials showed that these models should take into account the features of thermophysical processes in each zone of the extruder (radiant heat transfer during induction heating of the extruder body, nonlinear properties of heat capacity in the loading zone the motion of the polymer melt and the nonlinear properties of the viscosity coefficient, as well as

the presence of a phase transition solid mixture - liquid phase).

To obtain solutions of the corresponding boundary value problems described by systems of nonlinear differential equations in partial derivatives of parabolic type, the author developed a numerical-analytical method based on the application of finite integral transformations. The application of this method to the solution of nonlinear boundary value problems is based on the approximation of cylindrical functions (Neumann functions) by fractional-rational functions using the apparatus of chain fractions, as well as on the approximate representation of convective components in the corresponding equations by fractional-rational functions. Algorithms for integrating the products of several cylindrical functions based on the representation of these functions by fractional-rational expressions have been developed.

Appropriate algorithmic and software has been developed, which provides an opportunity to automate the process of solving these problems and determine the optimal values of rheological and thermophysical parameters of the object of study.

The influence of the specific power of the inductor on the length of the loading zone is investigated taking into account the radiant radiation at the inductor - extruder body boundary. It is shown that the determining factor influencing the quality of the final product is the velocity of the phase boundary of the solid mixture - polymer melt, which should be equal to the radial component of the screw rotation speed to provide the calculated geometric parameters of the extruder zones.

Since the changing the speed of rotation of the auger leads to a change in the design characteristics of the extruder, it is appropriate to control the processes of heat and mass transfer in it by changing the specific power of the inductor in each zone heating the extruder housing.

Mathematical modeling of the processes of homogenization and crystallization of the polymer melt based on the kinetic relations between homogenization and crystallization is performed.

The methods of mathematical and computer modeling of mass and heat transfer

processes in polymers developed in the dissertation taking into account nonlinear properties provided an opportunity to significantly improve the quality of modeling, develop recommendations for designing and improving technological processes for manufacturing ultrahigh cables.

The scientific novelty of the work is that for the first time a method of numerical-analytical solution of nonlinear differential equations in partial derivatives of parabolic type and its application to solve problems of heating the extruder body, heating the polymer mixture in the loading zone, polymer melting zone and homogenization and crystallization of the polymer melt in the dosing zone. The performed researches make it possible to maintain the calculated parameters of the extruder in the manufacture of polymer products (insulating coating of cables for ultra-high voltages, production of a wide range of polymer films, etc.), as well as calculating the optimal size of these zones. devices.

Keywords: extruder, Stefan-type problem, integral transformations, iterative procedure, numerical-analytical method, phase transition, Bessel functions, Navier-Stokes equation, screw, worm.

Список опублікованих праць за темою дисертації

1. Островерхов М.Я., Сільвестров А.М., Зеленський К.Х. Методи дослідження електротехнічних систем і комплексів. Монографія, Київ, Талком, 2019.-- 300 с.

2. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация объектов с распределенными параметрами в режиме нормального функционирования. Изв. Вузов, Приборостроение. Т.XX, №10, 1977. –С.35 –47. (WoS)

3. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Структурная идентификация в частотной области нестационарних объектов с распределенными параметрами. Межв. науч.-техн. сб. Адаптивные системы автоматического управления. 1976, №4. С.19 –29.

4. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация нелинейных объектов с распределенными параметрами. Изв. АН СССР, серия техн. кибернетика, 1978, №2. С. 72 –79. (WoS)

5. Бутаков Г.О., Зеленський К.Х., Ігнатенко В.М. Побудова моделі формування геометрії шва при зварюванні виробів неплавким електродом. Межв. науч.-техн. сб Адаптивні системи автоматичного управління.1998, №1 (21). – С. 69 –76.

6. Зеленський К.Х., Кеменяш Ю.М. Комп'ютерне моделювання процесів демпфування рідини у рухомих ємностях Межв. науч.-техн. сб Адаптивні системи автоматичного управління, 2005, №09(29). С .73—79.

7. Зеленський К.Х., Ліщина В.М. Моделювання динаміки обмеженого обсягу рідини із вільною поверхнею. ІПМС, Збірник наукових праць, вип.38, 2007. С. 135 –141

8. Зеленский К.Х., Игнатенко В.Н. Оптимальное управление в системах с запаздыванием Межв. науч.-техн. сб Адаптивні системи автоматичного управління. –2008, №12(32). –С.112 –117.

9. Зеленський К.Х. Визначення геометрії зварного шва при зварюванні

КДЕ. Міжв. наук.-техн. зб. Адаптивні системи автоматичного управління, 2009, №13(33). – С.118 –125.

10. Зеленський К.Х., Ліщина В.О., Ваврук Є. Математичне моделювання низинних лісових пожеж. Вісник ЛПУ, Комп'ютерні науки та інформаційні технології, №638, 2009. С.95 –99.

11. Зеленський К.Х., Ліщина В.О. Моделювання динаміки обмеженого обсягу рідини із вільною поверхнею. ІРМС, Збірник наукових праць, вип.382007, С.57—63.

12. Зеленський К.Х., Ігнатенко В.М. Оптимальне управління системами із запізненням. Межв. науч.-техн. сб Адаптивні системи автоматичного управління. 2008, №12(32). С.93 –97.

13. Зеленський К.Х., Числово-аналітичний метод розв’язання просторово-часових задач із рухомими межами. Межв. науч.-техн. сб Адаптивні системи автоматичного управління. 2009, №13(33). С. 107 –117.

14. Зеленський К.Х., Ітераційний метод розв’язання нелінійних крайових задач. Наукові нотатки Луцького національного університету, вип. 26, т, 2, 2009. С. 92 –99.

15. Зеленський К.Х., Ліщина В.М. Математичне моделювання аеродинаміки верхових лісових пожеж Наукові нотатки. Міжвузовський збірник, Луцьк, 2010. Вип. 27. с. 110 –115.

16. Бовсуновська К.С., Зеленський К.Х., Прийомов С.Г. Математичне моделювання закручених потоків у циклонних камерах. Вісник Університету "Україна". серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2011, №2. с. 43 –51.

17. Зеленський К.Х., Ігнатенко В.М., Бовсуновська К.С. Комп'ютерне моделювання динаміки повітряних потоків у циклонних камерах. Межв. науч.-техн. сб Адаптивні системи автоматичного управління. 2012, №21(41). С. 132 – 145.

18. Бовсуновська К.С., Зеленський К.Х. Комп’ютерне моделювання руху твердих домішків у циклонних камерах. Науковий журнал ``Комп’ютерно-інтегровані технології: освіта, наука, виробництво", Луцьк, 2013, Вип. №13. С.71 – 78.

19. Zelensky C. Ch., Zelenskaya N. C. Approximation of Bessel functions by rational functions. Electronics and control systems. 2015, N 2 (44). p. 121 –124. DOI: 10.38372/1990-5548.44.8908

20. Зеленский К.Х., Настенко Е.А. Математическое моделирование динамики левого желудочка. Вісник Університету "Україна". серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2016, №1 (18) с. 27 – 38.

21. Зеленська Н.К., Зеленський К.Х. Апроксимація циліндричних функцій дробово-раціональними виразами. Вісник Університету ``Україна"; серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2016, №2(18) с. 17 – 23.

22. Зеленський К.Х., Болховітін В.М. До визначення концентрації домішків у двофазних циклонних пристроях. Вісник Університету ``Україна", серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2016, №2(18) с. 32 –38.

23. Таланчук П.М., Зеленський К.Х., Болховітін В.М. Моделювання процесу ізоляційного покриття. Вісник Університету "Україна". серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2016, №1 (18) с. 5 – 13

24. Зеленський К.Х., Ігнатенко В.М., Стєнін О.А. Структурна властивість оптимальних за витратами палива процесів управління у динамічних системах. Межв. науч.-техн. сб Адаптивні системи автоматичного управління. 2017, Вип. 1(42), Дніпро. С. 97 –103.

25. Болховітін В.М., Зеленський К.Х. Математичне моделювання

температурних режимів полімерного покриття кабельних виробів. Вісник Університету ``Україна", серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2017, №1 (19) с. 50 –53.

26. Zelensky C.Ch., Nastenko E.A. Simulation of vortex flows in the left ventricle of the heart. Electronics and control systems. 2017, N 2 (46). p. 73 –79. DOI: 10.38372/1990-5548.52.1183

27. Зеленский К.Х., Бурлаков М.В. Моделювання процесів плавлення полімерів у гвинтовому каналі шнека. Вісник Університету ``Україна", серія : інформатика, обчислювальна техніка та кібернетика, Київ, 2018, №1 (19) с.166 – 174

28. Kostyuk V.I., Kraskevitch V.E., Zelensky K. Kh.. Frequency domain identificaion of complex systems. Systems Sceince, 1977, V.2 C.5 –12.

29. Зеленський К.Х, Бовсуновська К.С. Математическое моделирование конвективно- диффузионных процессов в циклонных камерах. Актуальные проблемы гуманитарных и естественных наук, Журнал научных публикаций, Москва, 2014. с. 44—49

30. Ie. Nastenko, V. Pavlov, E. Nosovets, K. Zelensky. Optimal complexity models in individual control strategy task for objects that cannot be related. 2019 IEEE 14th International conference on computer Sciences and information technologies p. 207-210. doi: 10.1109/STC-CSIT.2019.8929831 (Scopus)

31. Ie. Nastenko, V. Pavlov, E. Nosovets, K. Zelensky Solving the Individual Control Strategy Tasks Using the Optimal Complexity Models Built on the Class of Similar Objects. Advances in Intelligent Systems and Computing IV. CCSIT 2019. Advances in Intelligent Systems and Computing, vol 1080. doi.org/10.1007/978-3-030-33695-0-36 Springer (Scopus)

32. O. Trofymchuk, K.Zelensky, Ie. Nastenko. Modeling of a temperature field for extruder body. System research and information technologies, 2021, №1 doi.org/10.20535/SRIT.2308-8893.2021.1.03 (Scopus)

33. Зеленський К.Х., Бовсуновська К.С., Болховітін В.М. Алгоритмічне забезпечення розв'язання нелінійних крайових задач тепломасопереносу. Modern engineering and innovative technologies. V.15, №1, 2021, DOI 10.30890/2567-5273.2021-15-01, p. 5 –12. (Index Copernicus)

34. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация нестационарных объектов с распределенными параметрами в частотной области. Труды IV-го симпозиума ИФАК ``Идентификация и оценка параметров систем".--Тбилиси -- ``Мецниереба", 1976, Т.3. С.83 –96.

35. F.Kissilevski, K.Zelensky. Adaptive control for welding robot. Proceedings of intern. conf. Automation and robotisatiion in welding and allied processes. Strasbourg, France, 1985. С.127 –135.

36. Кіселівський Ф.М., Бутаков Г.О., Зеленський К.Х. Adaptive control of velding robots. 2-d Intern. Conf. Developments in Automated and robotic Welding, Cambridge: The welding Institute, 1987. p.21-1 –21-11

37. Зеленский К.Х. Оценка геометрии сварочной ванны при сварке концентрированными потоками энергии. Доклады 2-й межд. конф. по ЭЛТ. Варна, 1988, С.105 –112.

38. Зеленский К.Х. Адаптивное управление процессами сварки неплавящимся электродом. Труды X межд. симпозиума ``Welding--90", 1990. Брно. С. 37 –42.

39. Зеленський К.Х. Математическое моделирование температурного поля сварочной ванны. Матеріали V Міжнародної науково-технічної конференції. Т.2. Аерокосмічні системи моніторингу та керування. Київ, 2003. С. 24.48 – 24.57.

40. Зеленський К.Х. Числово-аналітичний метод розв'язання нелінійних крайових задач математичної фізики. Матеріали IV міжнар. наук. конф. ISDMCI'2009. Євпаторія. 2009.

41. Бовсуновська К.С., Зеленський К.Х. Математичне моделювання

аеродинамічних процесів у циклонних апаратах. ISDMCI'2011. Євпаторія. 2011. С. 38 –40.

42. Зеленський К.Х. Математичне моделювання теплових процесів при вирощуванні монокристалів. Матеріали IV Всеукраїнської науково-практичної конференції ``Комп'ютерні технології: наука і освіта", Україна, Луцьк, 9--11 жовтня 2009. С. 53 –56.

43. Бовсуновська К.С., Зеленський К.Х. Комп'ютерне моделювання руху твердих домішків у циклонних камерах. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. Луцьк, 2013, №13, с. 71 –77.

44. Болховітін В.М., Зеленський К.Х. Моделювання двохфазних течій у турбулентному потоці. Матеріали X Всеукраїнської науково-практичної конференції ``Комп'ютерні технології: наука і освіта", Україна, Київ, 30 –31 березня 2017. С. 75 –78.

45. Зеленский К.Х., Болховітін В.М. Математичне моделювання процесу охолодження розплавів полімерів. Тези ІТОНВ-2019. Міжнародна науково-практична конференція «Інформаційні технології в освіті, науці і виробництві, м. Луцьк, 23-25 травня 2019 р. С. 44 –47.

46. Optimal complexity models in individual control strategy task for objects that cannot be related. IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2019.

47. Зеленский К.Х., Болховітін В.М. 22 наукова і практична конференція ``Theoretical foundations for the implementation and adaptation of scientific achievments in practice", 2020, Helsinki, Finland.

ЗМІСТ

ВСТУП

Актуальність проблеми. Внаслідок великої продуктивності екструдерів, їхньої суттєвої коштовності, а також високої ціни полімерних матеріалів експериментальні дослідження з модернізації обладнання та удосконалення технологічних режимів призводять до коштовних витрат матеріалів та часу. Це спонукає до розвитку теоретичних засад процесів, що досліджуються. Одним із основних інструментів, що сприяє отриманню потрібних результатів і надає можливість мінімізувати коштовні натурні випробування, є математичне моделювання. Але існуючі математичні моделі процесів течії, теплообміну і фазового перетворення полімерів у каналах екструзійного обладнання не забезпечують якісний та кількісний аналіз процесів, оскільки переважна більшість із них формулюються або в одновимірній постановці, або формулюються у тривимірній постановці, але розв'язання відповідних крайових задач не наводяться або на розв'язки посилаються на відомі ``стандартні" пакети чисельного розв'язання, які за своєю природою не пристосовані до вирішення нелінійних моделей, оскільки для розробки методів розв'язання нелінійних рівнянь, особливо для диференційних рівнянь із частинними похідними не може бути ``стандартних" пакетів чисельного розв'язання у силу специфіки наявних нелінійностей у рівняннях.

У зв'язку з цим виникає проблема, що пов'язана із розробкою методів розв'язання систем нелінійних рівнянь із частинними похідними математичної фізики. Як відомо, переважна більшість цих методів грунтується на використанні або різницевих схем різного порядку, або на використанні методів скінченних елементів, що по суті мало відрізняється від різницевих схем. Крім того, переважна більшість таких підходів обмежується задекларованими постановками тривимірних крайових задач, а результати моделювання подаються в одновимірному вигляді відносно просторових координат (у

декартовій системі координат). Вочевидь, це пов'язано із проблемами пошуку розв'язків відповідних систем нелінійних алгебраїчних рівнянь; вочевидь, така задача не може бути вирішена за допомогою ``стандартних'' пакетів тощо.

Тому розробка методів наближеного розв'язання таких задач, що вільні від означених недоліків, є актуальна наукова проблема для широкого класу нелінійних диференційних рівнянь математичної фізики.

Вирішення цієї проблеми стало можливим у зв'язку з інтенсивним розвитком комп'ютерної техніки, що надало можливість автоматизувати процес реалізації наближених алгоритмів розв'язання нелінійних крайових задач і створити передумови для вирішення задач автоматичного управління складними системами із розподіленими параметрами.

Підґрунттям для вирішення цих задач є застосування скінченних інтегральних перетворень до нелінійної крайової задачі із застосуванням апроксимуючих алгоритмів на ґрунті апарату ланцюгових дробів.

Вирішення цих питань є важливим з точки зору покращення якості продукції, підвищення ефективності робіт при проектуванні і модернізації екструзійного обладнання та при удосконаленні технологічних режимів. Тому розвиток теоретичних засад процесів руху і теплообміну нелінійних полімерних середовищ в умовах фазового переходу у каналах екструзійного обладнання за допомогою математичного моделювання процесів, що досліджуються, є актуальний напрямок, що містить наукову новизну, практичну значимість і являє собою теоретичне узагальнення крупної наукової проблеми.

У розвиток методів математичного і комп'ютерного моделювання динамічних об'єктів із розподіленими параметрами вагомий внесок зроблено завдяки науковим працям Бутковського А.Г., Верланя А.Ф. , Пупкова В.В., Пустильнікова Л.М., Рапопорта Є.Я., Самойленка А.М., Сергієнка І.В.,

У розвиток методів дослідження процесів тепло і масоперенесення в екструдерах значний внесок здійснено у наукових працях провідних вчених у

цій галузі: Кіма В.С., Митрошина В., Раувендаля К., Тадмора Е., Труфанової Н.М., Янкова В.И.

Зв'язок роботи з науковими програмами, планами, темами. Роботу виконано у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» на кафедрі теоретичної електротехніки відповідно до пріоритетних напрямків розвитку науки і техніки в Україні (Постанова Кабінету міністрів (КМ) України від 24.12.2001 р., № 1716), Державної програми розвитку промисловості на 2003--2011 роки (Постанова КМ України від 28.07.2003 р., № 1174), а також відповідно до тематики держбюджетних науко-дослідних робіт КПІ ім. Ігоря Сікорського: НДР. "Методи та засоби структурно-параметричної ідентифікації електротехнічних систем технологічної лінії з виробництва вітчизняного кабелю з полімерною ізоляцією на надвисокі напруги", (№2908-п № державної реєстрації 0116U003716; замовник -- МОН України; автор -- виконавець теми)

Мета роботи і завдання досліджень.

Розробка математичних моделей процесів течії і плавлення нелінійних полімерних середовищ у каналах екструзійного обладнання і розв'язання відповідних нелінійних крайових задач, а також дослідження на їх грунті впливу конструктивних і технологічних параметрів на гідродинамічні характеристики пластикуючих екструдерів і формуючих інструментів, визначення оптимальних режимів роботи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Виконати теоретичний опис і розробку просторових математичних моделей процесів нагріву, плавлення, течії і теплообміну полімерних матеріалів в умовах фазового перетворення у каналах екструзійного обладнання із урахуванням нелінійних властивостей фізико-технічних характеристик полімерів.

2. Розробити методи розв'язання комплексу математичних моделей у

різних зонах екструдеру (нагріву, плавлення, дозування), які б забезпечували потрібну якість кінцевого продукту на виході із екструдера.

4. Урахувати вплив нелінійних фізико-технічних характеристик полімерних середовищ на процеси теплоперенесення в екструдері.

5. Розробити відповідне алгоритмічне і програмне забезпечення для розрахунку і аналізу процесів, що досліджуються.

Об'єкт дослідження -- процеси екструзійного перероблення термопластичних полімерних матеріалів.

Предмет дослідження -- процеси масо- і теплоперенесення в екструзійних устаткуваннях полімерних матеріалів.

Методи дослідження. Дослідження грунтуються на методах математичного моделювання із застосуванням теорії суцільних середовищ, методів математичної фізики, методів наближеного аналізу,

Наукова новизна отриманих результатів полягає у створенні наукових засад розв'язання систем нелінійних диференційних рівнянь у частинних похідних математичної фізики та математичного і комп'ютерного моделювання процесів масо- і теплоперенесення процесів переробки полімерних матеріалів в одношнекових екструдерах із урахуванням їхніх в'язкопластичних властивостей та наявністю фазових переходів у системі тверда суміш--розплав полімерів.

Уперше отримані такі наукові результати:

1. Розроблено числово-аналітичний ітераційний метод розв'язання нелінійних крайових задач, що описуються системами нелінійних диференційних рівнянь у частинних похідних математичної фізики, який на відміну від існуючих (відомих?) підходів надає можливість отримати розв'язок цих задач у квадратурах, що сприяє створенню систем автоматичного або автоматизованого управління технологічними процесами у реальному часі.

2. Розроблено математичне та алгоритмічне і програмне забезпечення, що реалізує цей метод. Зокрема, запропоновано метод апроксимації циліндричних

функцій дробово-раціональними виразами. Розроблено алгоритми еквівалентного спрощення складних виразів, що ґрунтуються на використанні апарату ланцюгових дробів. Розроблене алгоритмічне та програмне забезпечення надає можливість автоматизувати процес отримання розв'язків нелінійних крайових задач.

3. Розроблено методи розв'язання нелінійних крайових задач тепломасоперенесення у зоні завантаження і пластикації екструдера, що ґрунтуються на використанні запропонованого ітераційного методу розв'язання нелінійних крайових задач.

4. Розроблено метод розв'язання крайових задач, що описують конвективно- дифузійні процеси у зоні плавлення полімерів та у зоні дозування із урахуванням реологічних властивостей полімерів.

6. Запропоновано метод оптимального управління температурними полями в екструзійних пристроях як управління нелінійним об'єктом із розподіленими параметрами.

Дістали подальший розвиток:

1. Удосконалені математичні моделі процесів нагріву корпусу екструдера, процесу завантаженя, плавлення полімерів у одно шнековому екструдері.

2. Вплив різних чинників на динаміку плавлення полімерних сумішей, що підвищує ефективність проектування і модернізації екструдера.

3. Вплив процесів гомогенізації та кристалізації на якість кінцевого продукту.

На захист виносяться:

-- Постановка задачі математичного моделювання процесами тепломасоперенесення в екструзійних пристроях.

-- Числово-аналітичний ітераційний метод розв'язання нелінійних крайових задач, що описують процеси тепломасоперенесення, які описуються системами нелінійних диференційних рівнянь у частинних похідних

математичної фізики.

-- Методи апроксимації циліндричних функцій дробово-раціональними функціями.

-- Метод розв'язання крайової задачі, що описує температурне поле у зоні завантаження із урахуванням джерела нагріву та залежності питомої теплоємності від температури суміші полімерів, що суттєво підвищує точність отриманого розв'язку.

-- Метод розв'язання крайової задачі, що описує конвективно-дифузійні процеси у зоні плавлення полімеру.

-- Метод розв'язання крайової задачі, що описує процеси формування тришарового ізоляційного покриття рухомої металевої жили з метою забезпечення якісного виготовлення кабелів на надвисокі напруги.

-- Метод автоматичного управління джерелом нагріву з метою стабілізації процесів нагріву, плавлення та охолодження полімерів для запобігання перегріву полімерної маси.

-- Результати числово-аналітичного моделювання процесів гідродинаміки, тепломасоперенесення у шнекових агрегатах і кабельних головках.

Достовірність отриманих результатів забезпечується їх задовільною збіжністю із відомими теоретичними результатами, із результатами проведених експериментальних досліджень, експериментальними даними інших авторів та із наявними точними розв'язками тестових задач.

Практична значимість і реалізація результатів роботи.

Математичні моделі процесів нагрівання, плавлення і течії полімерів у каналах екструзійних машин та інструментах, що формують, які розроблено у дисертаційній роботі, надають можливість:

- проектувати нове обладнання, удосконалювати технологічні режими, при цьому мінімізувати коштовні натурні випробування;

- визначати області локальних перегрівів, що важливо при використанні

сучасних полімерних матеріалів;

- враховувати вплив процесів тепломасоперенесення у шнеці на процеси пластикуючої екструзії;

- розробляти системи автоматичного управління технологічними процесами екструзії, орієнтованими на випуск нової продукції;

Дослідження, виконані автором, розширюють уявлення про процеси тепломасоперенесення, руху нелінійних полімерних середовищ у каналах екструзійного обладнання в умовах фазового переходу.

Практичні рекомендації використовувалися при удосконаленні технологічних режимів роботи пластикуючих екструдерів при виготовленні пластмасової ізоляції на ПНВП «Прикарпатгаз», що підтверджено актом впровадження та ТОВ «Укрекоконсалт». Теоретичні результати дисертаційної роботи використовуються у навчальному процесі на кафедрі біомедичної кібернетики факультету біомедичної інженерії НТУУ «КПІ ім. Ігоря Сікорського»

Особистий внесок здобувача. Здобувачем сформульовано мету і постановку завдань досліджень, розроблено методи та способи досягнення поставленої мети, сформульовано математичну постановку задач, розроблено ітераційний числово-аналітичний метод розв'язання крайових задач, що описуються системами нелінійних диференційних рівнянь математичної фізики. Здобувачем особисто здійснено наукове обґрунтування впливу реологічних та фізико-технічних характеристик полімерних матеріалів на точність математичного моделювання процесів масо- і теплоперенесення в екструдері, що надало змогу вдосконалити існуючі конструкції екструдерів для переробки полімерних матеріалів і створення нових технологічних ліній для переробки цих матеріалів.

Основні результати теоретичних і практичних досліджень, що представлено у дисертаційній роботі, висвітлено у наукових працях, що

наведені у списку публікацій автореферату [1-47]. Зокрема, автором дисертації особисто:

Розроблено ітераційний числово-аналітичний метод пошуку розв'язань крайових задач для систем нелінійних диференційних рівнянь у частинних похідних математичної фізики та алгоритми комп'ютерної реалізації методу.

Комп'ютерне моделювання процесів у зонах змішування та плавлення із урахуванням нелінійних властивостей полімерних матеріалів здійснювалось разом із аспірантом Болховітіним В.М.

Апробація результатів дисертації. Основні засади і результати дисертації доповідалися і обговорювалися на: IV симпозіумі IFAC ``Ідентифікація і оцінка параметрів систем", Тбілісі, 1976; Всеросійській школі-семінарі ``Чутливість систем управління", Владівосток, 1976; міжнародній конференції ``Automation and robotization in weldnig and applied processes", Stasburg, 1985; 2-nd international conference ``Developments in automated and robotic welding", Cambridge, 1987; 2-й міжнародній конференції по електронно-променевому зварюванню, Варна, 1988; Всесоюзній конференції ``Математичне та імітаційне моделювання в системах проектування і управління", Чернігів, 1990; 5-й міжнародній науково -практичній конференції ``Аерокосмічні системи моніторингу та керування", Київ, 2003; міжнародному симпозіумі SISPRO, Москва, 2006; 4-й Всеукраїнській науково-практичній конференції ``Комп'ютерні технології: наука і освіта", 2009; 4-й міжнародній науковій конференції ISDMCI'2009, Євпаторія, 2009; 8-й Всеукраїнській науково-практичній конференції ``Комп'ютерні технології: наука і освіта", 2013; 10-й Всеукраїнській науково -практичній конференції ``Комп'ютерні технології: наука і освіта", 2017; VII-й міжнародній науково-практичній конференції ``ІТОНВ-2019"; 22 науковій і практичній конференції ``Theoretical foundations for the implementation and adaptation of scientific achievments in practice", 2020, Helsinki, Finland.

Публікації. Основні результати дисертаційної роботи опубліковано у 47 друкованих працях, із них: 24 – у фахових виданнях; 6 – статті у наукових періодичних виданнях іноземних держав; 3 – у наукових виданнях, індексованих у наукометричній базі Scopus.

Структура и обсяг работи. Дисертація складається із вступу, 8-ти розділів, висновків, списку літератури, що містить 358 найменувань, і додатків. Загальний обсяг роботи 312 сторінок, основної частини – 232 сторінок, в тому числі 48 рисунків, 4 таблиці.

Розділ 1 ОПИС ТЕХНОЛОГІЇ ВИРОБНИЦТВА КАБЕЛЮ НА НАДВИСОКІ НАПРУГИ

1.1 Електричний кабель із полімерною ізоляцією на надвисокі напруги як продукт виробництва

Розвиток світової електротехніки зі створення кабельних ліній високої пропускної спроможності стримувався труднощами досягнення необхідної електричної міцності ізоляції. Пошук ізоляційних матеріалів в 70-ті роки XX століття привів до технології виготовлення кабелів з твердою ізоляцією із ``зшитого" поліетилену (міжнародне позначення XPLE) [153]. Завдяки високій електричній міцності та надійності з терміном служби не менше 30 років, простоті експлуатації, ремонту та екологічній чистоті високовольтні кабелі з ізоляцією XPLE стали найпоширенішими у діапазоні напруг до 500 кВ. Кабельні лінії на їх основі стають реальним засобом доставки електроенергії у найбільші центри електроспоживання (рис. 1.1). Промислове виробництво кабелів із ізоляцією XPLE на надвисоку напругу 330 кВ освоєно в Україні на заводі ``Південкабель" у м. Харків. В табл. 1.1 приведено позначення марки кабелів з ізоляцією із зшитого поліетилену. Наприклад, позначення кабелю із однією алюмінієвою струмопровідною жилою перерізом 500 мм$^2$ на напругу 330 кВ: ``Кабель АПвЭгаП-330 1х500 ТУ У 31.3-00214534-061: 2008". Марки кабелів з ізоляцією із зшитого поліетилену

Табл. 1.1 Марки кабелів з ізоляцією із зшитого поліетилену

Струмопровідна жила	А	Алюмінієва жила
	–	Мідна жила (без позначення)

Ізоляція	Пв	Ізоляція із зшитого поліетилену
Екран	Э	Мідний екран по ізольованій жилі
	г	Поздовжня герметизація екрану
		водоблокуючими стрічками
	га	Поздовжня і поперечна герметизація
		екрану водоблокуючими матеріалами
		і алюмополімерною стрічкою
Зовнішня оболонка	П	Зовнішня оболонка з поліетилену
		або сополімера поліетилену
	Пу	Посилена поліетиленова оболонка
	Внг	Зовнішня оболонка з ПВХ пластикату,
		що не поширює горіння при
		груповій прокладці кабелів
	Внгд	Зовнішня оболонка з ПВХ пластикату,
		що не поширює горіння і з низьким
		виділенням диму та корозієактивних газів
	Пнг	Зовнішня оболонка з полімерної
		композиції, що не поширює горіння
	Пнг-HF	Зовнішня оболонка з полімерної композиції,
		що не поширює горіння і не містить галогенів

Властивості зшитого поліетилену: електрична міцність за температури $20^{\circ}C$ – близько 50 кВ/мм; діелектрична проникність за температури $20^{\circ}C$ – не більше 2,5; тангенс кута діелектричних втрат за температури $90^{\circ}C$ – не більше 10^{-4}; стійкість до теплової деформації (нагрів зразку до $200^{\circ}C$ під

навантаженням 20 Н/см^2 впродовж 15 хв: подовження під навантаженням за $200^{\circ}C$ – не більше 175 %, залишкове подовження після зняття навантаження і охолодження – не більше 15 %; водопоглинання за температури $85^{\circ}C$ і тривалості випробування 14 діб – не більше 1 мг/см^2. Недоліком перших кабелів високої напруги з ізоляцією із зшитого поліетилену було інтенсивне старіння полімерної ізоляції. Встановлено, що старіння поліетилену в умовах впливу електричного поля визначається наявністю неоднорідностей в ізоляції, що виникають як у процесі виробництва, так і в початковому стані ізоляційного матеріалу. За наявності у полімерній ізоляції кабелю неоднорідностей під дією вологи і електричного поля у процесі експлуатації починають розвиватися провідні канали –``дендрити" (деревовидні утворення) або ``водні тріїнги" (рис. 1.3). Тріїнги розвиваються на ділянках, де існує неоднорідність структури ізоляції на межі із півпровідними екранами по жилі та ізоляції, а також із неоднорідних структур, що наявні у товщі ізоляції (забруднення, включення, мікропорожнечі). Наявність тріїнгів призводить до місцевих концентрацій електричного поля. Область ізоляції з тріїнгом згодом піддається окисленню, ізоляція швидко старіє, у результаті може настати її пробій.

Рис.1.1 Загальний вигляд трифазної кабельної лінії

Зовнішній вигляд конструкції кабелю марки ПвЭгаПу показано на рис. 1.2, де позначено: 1– мідна струмопровідна жила; 2 – внутрішній екструдований півпровідний шар; 3 – екструдована ізоляція із зшитого поліетилену; 4 – зовнішній екструдований півпровідний шар; 5 – обмотка півпровідним водоблокуючим полотном; 6 – мідний екран, виконаний у вигляді повиву мідних дротів, скріплених спірально накладеною мідною стрічкою; 7 – обмотка водоблокуючим полотном; 8 – алюмополімерна стрічка, накладена поздовжньо і зварена із зовнішньою оболонкою; 9 – екструдована посилена зовнішня оболонка з поліетилену високої щільності.

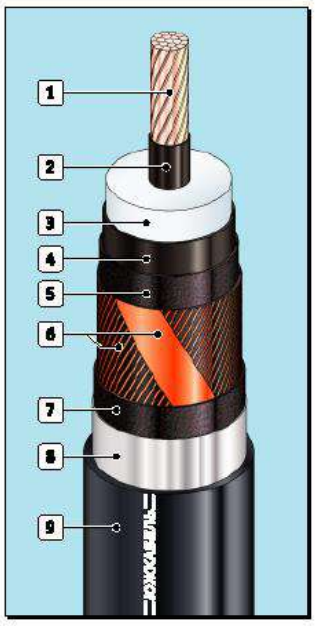


Рис. 1.2 Конструкція кабелю з ізоляцією із зшитого поліетилену заводу ``Південкабель"

Рис.1.3 Тріїнг, що зародився на чужорідному включенні в ізоляції

1.2 Технологічна лінія виробництва електричного кабелю

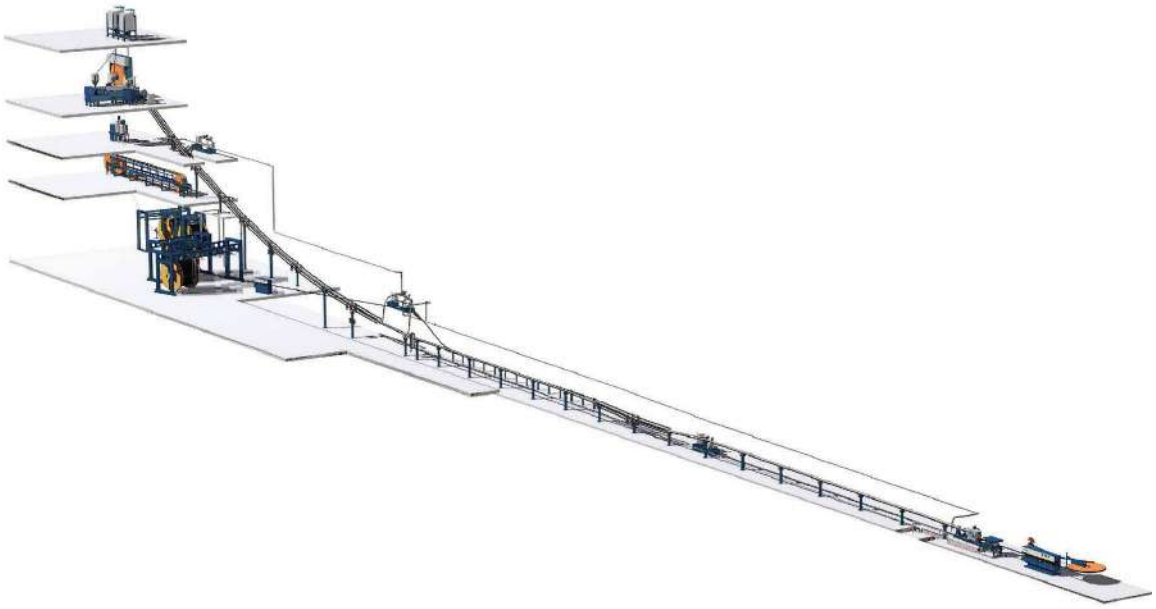


Рис.1.4 Загальний вигляд лінії нахиленого типу з виробництва кабелю

Процес виробництва кабелів з ізоляцією із зшитого поліетилену на заводі ``Південкабель" реалізується на неперервній технологічній лінії нахиленого типу довжиною 250 м та висотою 30 м (рис. 1.4).

Технологічне обладнання заводу відповідає останнім досягненням в області кабельної техніки. Застосовуються матеріали найвищої якості, що пройшли вхідний контроль. Це тріїнгостійкі надчисті ізоляційні і півпровідникові композиції для зшитого поліетилену та поліетилен високої щільності для оболонок кабелю виробництва фірми ``Borealis"; ПВХ пластикати, в т.ч. зниженої горючості і зниженої пожежонебезпеки фірми ``Промінвестпластик"; водоблокуючі матеріали фірм ``Lantor" та ``Geca Tapes". Застосування вакуумної упаковки при транспортуванні ізоляційних матеріалів і закритого процесу їх завантаження і екструзії забезпечує максимальну чистоту ізоляції.

Струмопровідні жили скручуються і ущільнюються на крутильній машині фірми ``Cortinovis ". Застосування ущільнення по повиві жили дозволяє отримати високий коефіцієнт ущільнення жили та гладку поверхню. За потреби накладаються також водоблокуючі матеріали. Одночасне накладення ізоляції та півпровідникових екранів здійснюється на похилих лініях газової вулканізації фірм ``Troester" та ``Maillefer". Вулканізація проходить у середовищі азоту для високих значень температури і тиску (``суха" вулканізація). Це дає можливість виключити потрапляння вологи в ізоляцію і отримати гладку та однорідну ізоляцію без пустот і сторонніх включень, із щільно прилеглими півпровідними екранами. Товщина і ексцентриситет шарів безперервно вимірюється приладами лазерного контролю. Накладення обмоток водоблокуючими стрічками, екранів із мідних дротів та стрічок проводиться на універсальній крутильній машині Drum Twister фірми ``Pourtier". Екструдування зовнішніх оболонок кабелів і накладення алюмополімерних стрічок (за необхідністю) відбувається на екструзійних лініях фірм ``Troester" та ``Maillefer", оснащених приладами вимірювання діаметру, контролю герметичності оболонки і пристроєм для маркування. Комплекс випробувального устаткування фірми ``Hipotronics" дозволяє проводити випробування кабелів на наявність в ізоляції часткових розрядів, а також випробування готових кабелів підвищеною напругою.

Все обладнання має комп'ютеризоване управління технологічними процесами і випробуваннями на базі математичного, програмного та технічного забезпечення фірми ``Siemens".

Підготовлена у процесі виробництва струмопровідна жила за допомогою колісного тягового пристрою зі швидкістю біля 50 м/хв подається в екструзійну головку блоку 1 трьох екструдерів, де на неї одночасно накладається три шари електроізоляції (рис. 1.5) товщина кожного шару ізоляції регулюється автоматизованими електроприводами екструдерів. Зовнішній діаметр кожного шару ізоляції вимірюється датчиками 2 із запізненням в 0,5 с.

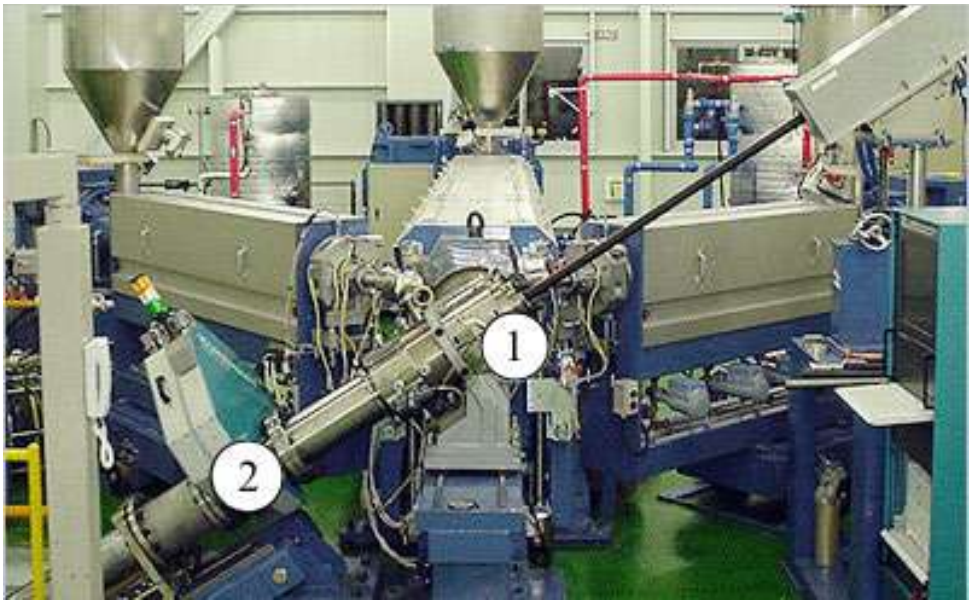


Рис. 1.5 Блок екструдерів (1) та датчики (2)

Для забезпечення максимальної чистоти ізоляції завантаження компонентів полімерної ізоляції в зону екструзії здійснюється з ``чистої'' кімнати (рис. 1.6). Потім ізольована жила надходить у трубу наноструктурного зміцнення (вулканізації), яка заповнена азотом під високим тиском (рис 1.7). У трубі ізоляція зшивається, набуваючи поліпшених термомеханічних характеристик, а далі охолоджується. Довжина камери вулканізації шарів ізоляції разом з камерою охолодження становить 172 м.

Для покращення якості ізоляції застосовується система попередньої температурної обробки на вході у камеру вулканізації. У першій секції труби вулканізації організовується циркуляція холодного азоту. На вході у секцію температура азоту становить 20 –25$^{\circ}C$, на виході підвищується до 45 –60$^{\circ}C$ залежно від температурного режиму і ефективності системи вентиляції. Холодний азот зменшує температуру поверхні ізоляції (зовнішнього електропровідного екрану і частини ізоляції) від 135$^{\circ}C$ (температура після виходу з екструзійної головки) до 35 –40$^{\circ}C$ та до 110 –120$^{\circ}C$ у товщі ізоляції.

Зовнішній шар ізоляції, охолоджуючись, рівномірно застигає, а внутрішній шар перебуває у розплавленому стані, що обумовлює прийняття ізоляцією круглої форми.

Рис. 1.6 Загальний вигляд ``чистої'' кімнати

У кабелях надвисокої напруги найчастіше використовуються мідні жили, які мають велику теплоємність, тому при накладанні на неї розплавленої полімерної маси внутрішній електропровідний шар та внутрішня частина ізоляції додатково охолоджуються жилою. У другій секції труби температура поверхні ізоляції підвищується у результаті нагрівання й її зовнішній шар плавиться та зшивається. Завдяки попередньому охолодженню більша частина ізоляції, що розташована ближче до жили, залишається застиглою. Діаметр цієї твердої циліндричної частини при проходженні вздовж труби вулканізації залишається відносно великим. Та частина ізоляції, що розплавлена, але незшита, залишається відносно малою і не дозволяє зовнішній частині ізоляції значно зміщуватися під власною вагою. При виході із труби вулканізації всі три шари електроізоляційної системи зшиваються повністю. Далі ізольована жила

охолоджується і за допомогою накопичувача (рис. 1.8) надходить на приймальний пристрій (рис. 1.9).

Діаметр ізольованої жили 122 мм, погонна вага 270 Н/м. Металічна жила з алюмінію чи міді може бути перетином 1000—3000 мм^2 з виконанням по типу сегментованої конструкції ``Мілікен". Маса барабана подачі з струмопровідною металічною жилою становить до 20 т, а маса барабану з готовою кабельною продукцією – до 30 т.

Система управління обладнана сучасними промисловими комп'ютерами з інтерфейсом, що дозволяє створювати, зберігати та систематизувати технологічні показники та результати випробувань для прийняття управлінських рішень підсистемою верхнього рівня.

Рис.1.7 Загальний вигляд камери вулканізації та охолодження

Рис. 1.8 Загальний вигляд приймального пристрою

Система управління виробництвом включає в себе такі функції:

— автоматичний розрахунок технологічних параметрів ліній, зокрема, пошарове співвідношення температури як функції часу на основі розрахунку теплопередачі між шарами та температурної залежності періоду півроспаду пероксиду;

— забезпечення синхронізації всіх вузлів лінії залежно від параметрів технологічного процесу та їх змін;

— сигналізацію та моніторинг у разі досягнення одним або кількома технологічними параметрами критичних значень;

— відстеження стабільності параметрів технологічного процесу і забезпечення автоматичної реакції на їх поточні вимірювання.

Рис. 1.9 Загальний вигляд накопичувача

Підприємство оснащене випробувальним обладнанням. Це надає можливість вести випробування і контроль всієї номенклатури кабельно-провідникової продукції. Центральна заводська лабораторія акредитована у системі сертифікації УкрСЕПРО на технічну компетентність. Кабелі піддаються приймально-здавальним, періодичним та типовим випробуванням. У процесі приймально- здавальних випробувань кабелі піддаються таким видам випробувань:

– перевірка герметичності оболонки;

– випробування змінною напругою величиною $2{,}5U_0$;

– вимірювання рівня часткових розрядів протягом 30 хв.;

– вимірювання електричного опору струмопровідної жили;

– перевірка маркування та упаковки.

Автоматизовану систему управління виробництва (АСУ) кабелю можна подати (рис. 1.10) як багаторівневу, де на верхньому рівні шляхом еволюційного планування факторного експерименту у просторі X^* уставок X_i^*, $i=\overline{1,n}$

локальних систем стабілізації технологічних параметрів відшукується вектор x^* їх оптимальних (за показником Λ якості продукції) значень. Автоматизовані електропроводи (АЕП) на основі інформації із підсистеми ідентифікації сигналів x і параметрів $\hat{\mathbf{b}}$ локальних об'єктів вирішують задачу оптимальної стабілізації змінних x відносно оптимальних уставок x^*. Відповідно на нижньому рівні системи (рис. 1.10) реалізується задача вимірювання значень x.

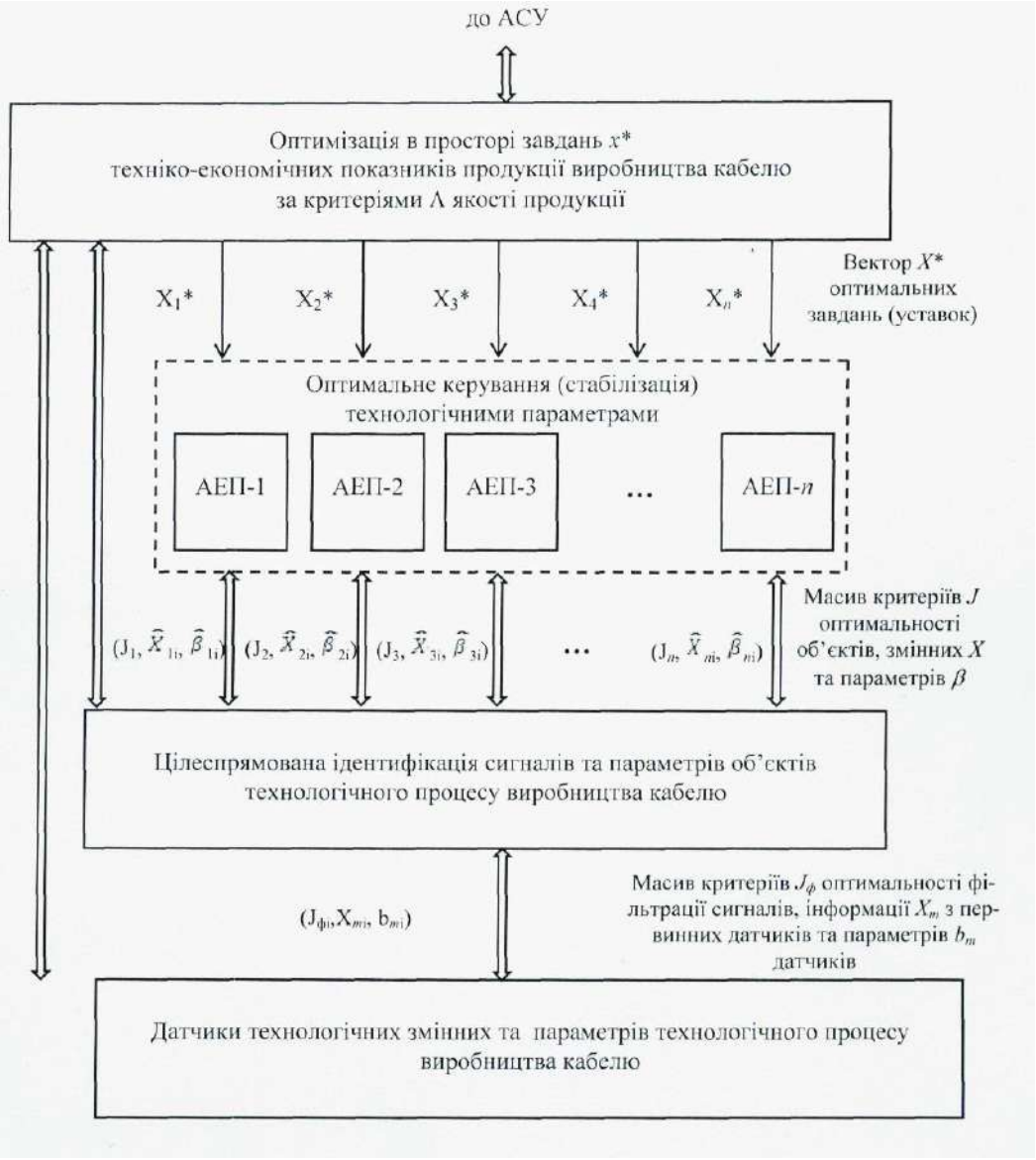


Рис. 1.10 Багаторівнева система виробництва кабелю

Склад основних систем електротехнічного комплексу технологічної лінії

для нанесення на струмопровідну металічну жилу тришарової поліетиленової ізоляції та її вулканізації у виробництві кабелю на надвисокі напруги до 330 кВ наведено на рис. 1.10. Комплекс складається: з системи електроживлення; кількох десятків автоматизованих електроприводів (АЕП); установок електрозварювання металевої жили, індукційного нагрівання жили, зонної електротермії, подачі азоту під тиском та охолодження першої зони ізоляції та ізольованої жили у цілому; локальних систем керування пристроями, установками та механізмами, зокрема систем керування швидкістю та натягом жили, дозування компонент ізоляції, температурних режимів, дегазації та сушки; автоматизованої системи управління технологічним процесом для забезпечення заданих показників якості кабельної продукції та техніко-економічних показників, оперативного надання та зберігання інформації про хід процесу, створення звітів.

Система електроживлення забезпечує комплекс потужністю близько 1320 кВА. Основними споживачами електроенергії є: автоматизовані електроприводи, зокрема електроприводи трьох екструдерів мають потужність близько 450 кВА; установка зонної електротермії – 350 кВА, електрозварювання – 300 кВА, індукційного нагрівання – 120 кВА.

Враховуючи багаторівневість і складність комплексу, для математичного аналізу, адаптації і оптимізації його підсистем і всього комплексу, як системи, слід застосовувати методологічні основи системного (модельного) аналізу.

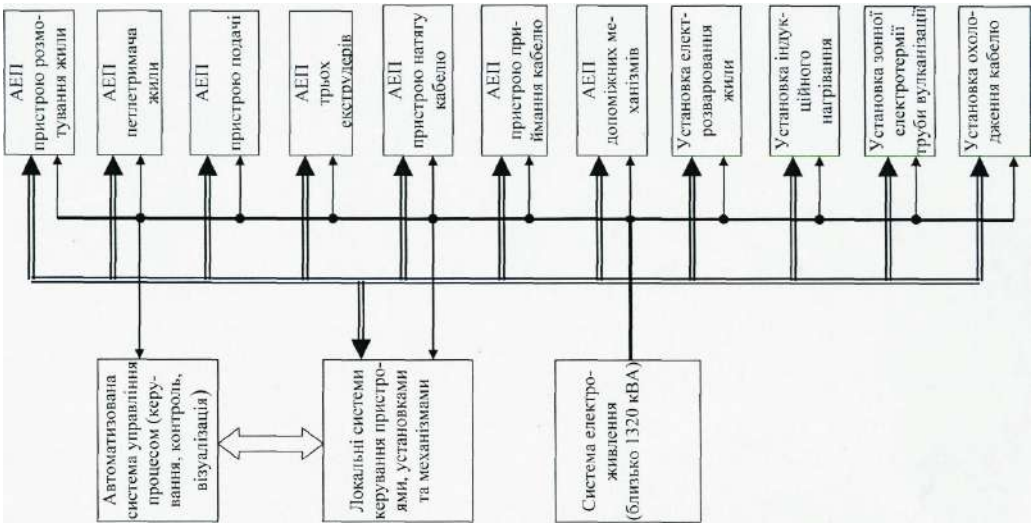


Рис.1.11 Структура електротехнічного комплексу технологічної лінії виробництва кабелю

1.2 Особливості нанесення на жилу ізоляції

Відомо, що реальна СПЕ-ізоляція відрізняється від ідеальної однорідної наномодифікованої полімерної ізоляції наявністю в її об'ємі різних нано- і мікродефектів (мікропорожнин, мікровиступів і впадин тощо), що виникають під час її виготовлення. Причому найбільша питома густина мікродефектів в ізоляції виникає біля поверхні розділу ``електрод - діелектричне середовище", де електродом є струмопровідна жила або мідний екран кабелю, а діелектричним середовищем – власне СПЕ-ізоляція. Більшість поверхневих ефектів в ізоляції виникають у процесі виготовлення кабелів.

У низці публікацій показано, що поверхневі мікродефекти збурюють ЕП в ізоляції сильніше за об'ємні мікродефекти аналогічних розмірів і конфігурацій. Тому при створенні електротехнічних систем (ЕТС) для виготовлення зшитої поліетиленової ізоляції у конструкціях надвисоковольтних кабелів особливу увагу треба приділяти розробці технологічних методів, які спрямовано на запобігання причин, що викликають появу нерівностей на поверхнях

металічних струмопровідних жил та півпровідних екранів кабелів, змінювання товщини і характеристик таких екранів, а також погіршення характеристик шару СПЕ-ізоляції.

Наявність внутрішніх нано- і мікроструктурних ефектів у твердій полімерній ізоляції сприяло накопиченню у них об'ємних електричних зарядів, збуренню електричного поля із суттєвим його підсиленням у локальних мікрооб'ємах ізоляції. Цей ефект підсилювався за недостатнього ущільнення многодротових жил великого перетину та при виготовленні таких жил із ізольованими многодротовими сегментами по типу конструкції ``Міллікен''. Наявність гострих мікровиступів на поверхні металічної жили суттєво збільшувало напруженість ЕП у локальних об'ємах ізоляції, що призводило до інтенсифікації інжекції електронів із поверхні жили в ізоляцію.

Поява тунельного ефекту у конструкціях силових високовольтних кабелів із твердою полімерною ізоляцією складало велику проблему для подальшого підвищення напруги на їхній жилі при підвищенні товщини такої ізоляції. Поверхню металічних жил силових кабельно-провідних виробів намагалися полірувати, покривати металічними стрічками із гладкою поверхнею, але такі технічні рішення не забезпечували розробку технологій, перспективних для промислового виробництва кабелів із твердою ізоляцією на надвисокі напруги.

Для запобігання повітряних порожнин між поверхнями металічної жили і твердою полімерною ізоляцією було розроблено технологію її екструзійного нанесення, яка передбачає розігрів ізоляції до рідинного стану, із подальшим нанесенням її на жилу та охолодженням до затвердіння.

У конструкціях силових кабелів на середні, високі та надвисокі напруги обов’язковим є застосування шару, що екранує, який зазвичай виготовляється із мідних дротів та стрічок, які також можуть мати мікровиступи у тверду полімерну ізоляцію та створювати у ній додаткові локальні збурення ЕП. Тому розроблено технології екструзійного нанесення і другого півпровідного

полімерного шару між твердою полімерною ізоляцією і мідним екраном кабелю.

Для запобігання наявності газових порожнин між ізоляцією і півпровідними шарами розроблено технологію одночасного екструзійного нанесення на жилу трьох полімерних шарів -- двох півпровідних та одного (між ними), що ізолює. Застосування у якості базового матеріалу одного й того самого поліетилену, що здатний до зшивання, забезпечило реалізацію міцного зчеплення їх між собою і зшивання усіх шарів у загальному обладнанні, що вулканізує.

У наш час такий метод нанесення тришарової полімерної ізоляції на жилу є обов'язковий при виготовленні кабелів на всі напруги від 6 до 330 кВ і більше. Але для його використання необхідно враховувати взаємну залежність напруги експлуатації кабелю та характеристик його сегментів: СПЕ-ізоляції, струмопровідних жили та екрану, півпровідних шарів між жилою та ізоляцією, а також між ізоляцією та екраном. Зокрема, треба враховувати, що напруженість ЕП в однорідній ізоляції дорівнює [137]

$$E_{\Delta} = \frac{U}{(R_1 + \Delta)\ln(R_2/R_1)},$$

де E_{Δ} – напруженість ЕП на відстані Δ від поверхні півпровідного шару із радіусом R_1 між жилою та ізоляцією; U – напруга між жилою та екраном; R_2 – радіус півпровідного шару між екраном та ізоляцією. Причому $R_1 = R_{æ} + \Delta_1$, $R_2 = R_{å} - \Delta_2$, $R_{æ}$ і $R_{å}$ – радіуси жили та екрану, а Δ_1, Δ_2 – товщини півпровідних шарів, що нанесені на жилу та ізоляцію.

Додаткове локальне збурення ЕП одиничними поверхневими та об'ємними мікродефектами ізоляції, а також зменшення локальних збурень із-за шорсткостей поверхні струмопровідної жили та екрану можна визначити із застосуванням математичної моделі та концепції аналізу збуреного ЕП. Згідно з

цією концепцією треба враховувати максимальні локальні напруженості ЕП та величину локальних напружених об'ємів у такій ізоляції [ ].

Величина його напруженого об'єму $V_{\text{і}}$ визначається як

$$V_{\text{і}} = \int_V f(E)dV = 2\pi \int_S f(E)r(S)dS,$$

де V – розрахунковий об'єм ізоляції; S – площа змінного перетину напруженого об'єму уздовж ЕП; r – змінний радіус перетину цього об'єму ортогонально полю; $f(E) = 1$ при $E_{\text{l}} \le E \le E_{\max}$ і $f(E) = 0$ при $E > E_{\text{l}}$.

Відомо, що із підвищенням напруги на ізоляції кабелів треба підвищувати якість поверхні струмопровідних жил, яка значною мірою визначає збурення електричного поля в об'ємах півпровідного шару та полімерної ізоляції. Тому величина напруги, що прикладається до ізоляції кабелів та якість поверхні їхніх струмопровідних жил нормують основні вимоги до параметрів і властивостей ізоляції та півпровідних полімерних шарів, що застосовуються у кабелях. При цьому характеристики і властивості елементів ізольованої жили залежать як від середньої напруженості ЕП ізоляції, так й від напруженості поля в її локальних мікрооб'ємах.

Для екструзійного нанесення трьохшарової полімерної ізоляції, що спроможна до зшивання, на металічну жилу високовольтних і надвисоковольтних кабелів та подальшого наноструктурного зшивання усіх шарів ізоляції застосовують електротехнологічні системи (ЕТС) двох типів: вертикального та нахиленого.

На ЕТС вертикального типу простіше забезпечити високу якість ізоляції, але складніше отримати високу швидкість її нанесення, що пропорційна довжині камери вулканізації. Технічно та економічно складніше здійснювати побудову та обслуговування таких систем, особливо за необхідності здійснювати виробництво кабелів на різні напруги та струми. За змінювання перетинів і конструкцій струмопровідних жил кабелів та товщини шарів

полімерної ізоляції необхідно здійснювати змінювання пускових, усталених і перехідних режимів усіх електромеханічних та електротехнологічних пристроїв і блоків ЕТС. На системі вертикального типу швидку і точну переналадку режимів здійснювати складно.

Системи нахиленого типу є продуктивніші і дешевші, вони простіше в обслуговуванні й їх легше переналагоджувати при змінюванні продуктивності, характеристик струмопровідної жили та полімерної ізоляції. Але за збільшення товщини усіх шарів ізоляції у системах нахиленого типу ускладнюється задача оптимізації режимів блоків, що зменшують зміщення шарів ізоляції, що не затверділи, відносно центру металічної жили.

1.3 Основні блоки і характеристики системи

Розрахунок електротехнологічнних режимів усіх пристроїв і блоків ЕТС нанесення многошарової полімерної ізоляції на жилу надвисоковольтних кабелів і подальшого зшивання такої ізоляції у вулканізаційній камері нахиленого типу свідчить, що від електромережі, що живить, необхідно передбачати вживання електричної потужності до 3 МВА.

У розрахунках, експериментальних дослідженнях у промисловому виробництві враховувалося, що сумарна довжина камери наноструктурного зміцнення (вулканізації) усіх шарів полімерної ізоляції і камери їх охолодження складає біля 172 м.

Зовнішній діаметр ізольованої жили складає до 120 мм, а її погонна вага – до 270 Н/м. Діаметр приймального барабану може бути до 3,6 м, вага барабану, що приймає, із металічною жилою – до 20 т, а вага, що приймає, із ізольованою металічною жилою – до 30 т.

Регулювання і стабілізація швидкості пересування жили із трьома полімерними шарами ізоляції у нахиленій частині вулканізаційної камери

здійснюється типовим частотно-регульованим електроприводом, що здатний створити на своєму валу зусилля до $4.5 \cdot 10^7$ Н. Такий електропривід забезпечує переміщення ізольованої жили у вказаних камерах із лінійною швидкістю до 50 м/хв.

У склад ЕТС нанесення полімерної ізоляції на жилу та вулканізації ізоляції входили [142]:

– система силового електроприводу, що описана вище;

– система дозування компонентів, яка забезпечує пневматичну подачу до екструзійної групи усіх необхідних для технологічного процесу компонентів (подача таких компонентів здійснюється у режимі ``чистої кімнати", яка забезпечує запобігання попадання у вихідні полімерні матеріали небажаних сторонніх твердих, рідинних та газоподібних мікровключень;

– система контактного електропідігріву азоту в усіх вулканізаційних камерах діаметром до 0,2 м;

– система електроживлення та охолодження азотом, яка забезпечує неперервну подачу у вулканізаційну трубу промислового азоту (чистотою до 99,5%б продуктивністю до 12 $м^3$/г при тиску 1 атм і температурі 20° С) – для первинного охолодження усіх шарів полімерної ізоляції до отвердіння одразу після її нанесення на жилу (що забезпечувало зменшення зсувів ізоляції відносно металічної жили), а також для вторинного охолодження усіх шарів полімерної ізоляції після завершення її наноструктурного зміцнення (тобто зшивки);

– система охолодження водою і повітрям, яка разом із системою підігріву і системою подачі та охолодження азотом забезпечує оптимальний режим вулканізації та охолодження шарів на струмопровідній жилі до температури цеху;

– система дегазації ізоляції, що нанесено на жилу, тобто видалення із

ізоляції побічних газових компонентів у спеціальні закриті ємності;

– система неперервного контролю та оброблення електричних і технологічних параметрів, яка контролює їх та одночасно формує вихідні сигнали для системи управління технологічними режимами і прийняття рішень операторами.

Основними показниками, що контролюються і використовуються як параметри регулювання, є:

– температура струмопровідної жили на вході у блок із трьома екструдерами, що суміщені;

– за відхиленням такої температури від заданого значення регулюється потужність індукційного підігріву рухомої металічної жили.

Досвід застосування інтегральних методів теорії електричних ланцюгів засвідчив, що за збільшення частоти струмів, що індукуються в ізольованих струмопровідних сегментах жили, вплив скін-ефекту та ефекту близькості підсилюється. Збільшення частоти струмів і наявність вказаних ефектів може викликати таке підвищення локальних температур на поверхні металічних сегментів, яке недопустиме для ізоляційного матеріалу між ними.

Для вирішення цієї задачі розроблено математичну модель і методику, що надають можливість здійснити числовий розрахунок розподілу електромагнітного поля та змінного струму в алюмінієвій і мідній жилах великого перетину (1000—3000 мм^2), що виконано по типу конструкцій ``Міллікен".

Найпростіші математичні моделі, що описують процес вулканізації кабельних виробів, формулюються у вигляді системи нестаціонарних рівнянь теплопровідності для внутрішнього циліндру (метал) та зовнішнього (ізоляція), наприклад, [86].

$$c_1\rho_1 V_0\frac{\partial T_1}{\partial t}=\lambda_1\left(\frac{1}{r}\frac{\partial T_1}{\partial r}+\frac{\partial^2 T_1}{\partial r^2}\right)+Q_p w_p, \tag{1.1}$$

$$c_2\rho_2 V_0\frac{\partial T_2}{\partial t}=\lambda_2\left(\frac{1}{r}\frac{\partial T_2}{\partial r}+\frac{\partial^2 T_2}{\partial r^2}\right)+Q_p w_p, \tag{1.2}$$

$$w_p=\rho_2\frac{d\varphi}{dt};\quad \frac{d\varphi}{dt}=(1-\varphi)k_0 e^{-E/(RT)} \tag{1.3}$$

Початкові умови:

$$T_1(r,0=T_2(r,0)=T_0,\ \varphi_0=0.$$

Межові умови:

$$\frac{\partial T_1}{\partial r}|_{r=0}=0,\quad \left[\lambda_1\frac{\partial T_1}{\partial r}=\lambda_2\frac{\partial T_2}{\partial r}\right]|_{r=r_1},\ T_1=T_2;$$

$$\left[\lambda_2\frac{\partial T_2}{\partial r}=\alpha(T_b-T_2)+\sigma\varepsilon_{pr}(T_b^4-T_2^4)\right]|_{r=r_2},$$

$$T_2=T_2=30^\circ C;\ T_b=227^\circ C.$$

Тут E – енергія активації хімічної реакції, $E=66$ кДж/моль; R – універсальна газова стала, $R=8{,}31$ Дж/(моль К); α – коефіцієнт теплообміну із навколишнім середовищем, $\alpha=1$ Вт/(m^2K); $\varepsilon_{pr}=0{,}5$ – приведений ступінь чорноти; $c_{iz}=c_2=1380$ Дж/кг$^\circ C$ – тепломісткість ізоляції; w_b – ступінь полімеризації ; $\lambda_{iz}=\lambda_2=0{,}16$ Вт/м$^\circ C$ – теплопровідність ізоляції; $c_p=c_1=385$ Дж/кг$^\circ C$ – тепломісткість жили; $\rho_p=\rho_1=8700$ Кг/м^3 – густина жили; $\lambda_p=\lambda_1=400$ Вт/м$^\circ C$; $V_0=0{,}08$ м/с – швидкість руху жил та ізоляції.

Як вважають автори [104], модель описує температурне поле у металічній жилі і полімерній ізоляції. У дійсності, це – одновимірна модель теплорповідноті металевого стрижня. Щодо твердження авторів про двохфазну течію, наведені рівняння не мають жодного стосунку із двофазими процесами, оскільки це передбачає наявність твердої і рідинної фаз.

1.4 Огляд підходів до математичного опису

На рис. 1.12 наведено схему теплових потоків при шнековому формуванні.

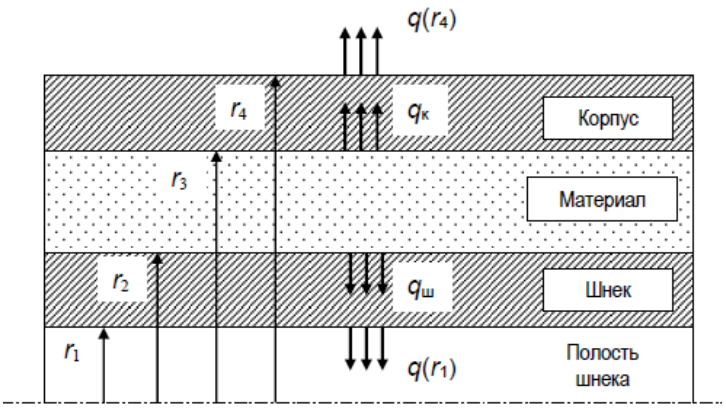


Рис.1.12 Схема теплових потоків при шнековому формуванні

Добре відомо [59], що ці процеси описуються повною системою диференційних рівнянь у частинних похідних Нав'є–Стокса.

Математичному опису процесів тепломасоперенесення при екструзійній пластифікації полімерів присвячено досить багато праць [145,147,151,160,161,164]. Природно, що в основі цих моделей є система диференційних рівнянь, що описує закони збереження маси, руху та енергії, тобто система рівнянь Нав'є—Стокса.

Система рівнянь складається із рівняння нерозривності, рівнянь руху полімерної маси (рівняння відносно компонент швидкості потоку) та рівняння енергії (температурного поля розплаву полімеру):

рівняння нерозривності:

$$\frac{\partial \rho}{\partial t}+\frac{\partial(\rho v_z)}{\partial z}+\frac{1}{r}\frac{\partial(\rho v_\theta)}{\partial r}+\frac{1}{r}\frac{\partial(r\rho v_r)}{\partial r}=0, \qquad (1.4)$$

рівняння руху:

$$\rho\left(\frac{\partial v_r}{\partial t}+v_r\frac{\partial v_r}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_r}{\partial\theta}-\frac{v_\theta^2}{r}+v_z\frac{\partial v_r}{\partial z}\right)$$
$$=-\frac{\partial p}{\partial r}+\frac{1}{r}\frac{\partial}{\partial r}(r\tau_{rr})+\frac{1}{r}\frac{\partial\tau_{r\theta}}{\partial\theta}-\frac{\tau_{\theta\theta}}{r}+\frac{\partial\tau_{rz}}{\partial z}+\rho g_r; \quad (1.5)$$

$$\rho\left(\frac{\partial v_\theta}{\partial t}+v_r\frac{\partial v_\theta}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_\theta}{\partial\theta}-\frac{v_r v_\theta}{r}+v_z\frac{\partial v_\theta}{\partial z}\right)$$
$$=-\frac{1}{r}\frac{\partial p}{\partial\theta}+\frac{1}{r^2}\frac{\partial(r^2\tau_{r\theta})}{\partial r}+\frac{1}{r}\frac{\partial\tau_{\theta\theta)}}{\partial\theta}+\frac{\partial\tau_{\theta z}}{\partial z}+\rho g_\theta; \quad (1.6)$$

$$\rho\left(\frac{\partial v_z}{\partial t}+v_r\frac{\partial v_z}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_z}{\partial\theta}+v_z\frac{\partial v_z}{\partial z}\right)$$
$$=-\frac{\partial p}{\partial z}+\frac{1}{r}\frac{\partial(r\tau_{rz})}{\partial r}+\frac{1}{r}\frac{\partial\tau_{\theta z}}{\partial\theta}+\frac{\partial\tau_{zz}}{\partial z}+\rho g_z; \quad (1.7)$$

рівняння енергії:

$$c_V\rho\left(\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+\frac{v_\theta}{r}\frac{\partial T}{\partial\theta}+v_z\frac{\partial T}{\partial z}\right)$$
$$=\lambda_T\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{1}{r^2}\frac{\partial^2 T}{\partial\theta^2}+\frac{\partial^2 T}{\partial z^2}\right)+Aq_T; \quad (1.8)$$

$k=\lambda_T/(\rho c_V)$ – коефіцієнт температуропровідності.

$$q_T=-T\frac{\partial p}{\partial T}\bigg|_\rho\left[\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{1}{r}\frac{\partial v_\theta}{\partial\theta}+\frac{\partial v_z}{\partial z}\right]+\tau_{rr}\frac{\partial v_r}{\partial r}+\tau_{\theta\theta}\frac{1}{r}\left(\frac{\partial v_\theta}{\partial\theta}+v_r\right)+\tau_{zz}\frac{\partial v_z}{\partial z}+$$
$$+\tau_{r\theta}\left(r\frac{\partial(v_\theta/r)}{\partial z}+\frac{1}{r}\frac{\partial v_r}{\partial\theta}\right)+\tau_{rz}\left(\frac{\partial v_z}{\partial r}+\frac{\partial v_r}{\partial z}\right)+\tau_{\theta z}\left(\frac{1}{r}\frac{\partial v_z}{\partial\theta}+\frac{\partial v_\theta}{\partial z}\right).$$

Компоненти тензору напруження зсуву:

$$\tau_{ij}=\mu\Delta_{ij}+\left(\nu-2/3\mu\right)(\nabla\cdot v)\delta_{ij};\ \Delta_{ij}=\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}.$$

У циліндричній системі координат маємо:

$$\tau_{rr}=\mu_e\left[2\frac{\partial v_r}{\partial r}-\frac{2}{3}(\nabla\cdot v)\right]+\nu(\nabla\cdot v); \quad (1.9)$$

$$\tau_{\theta\theta} = \mu_e\left[2\frac{1}{r}\left(\frac{\partial v_\theta}{\partial\theta} + v_r\right) - \frac{2}{3}(\nabla\cdot v)\right] + \nu(\nabla\cdot v); \tag{1.10}$$

$$\tau_{zz} = \mu_e\left[2\frac{\partial v_z}{\partial z} - \frac{2}{3}(\nabla\cdot v)\right] + \nu(\nabla\cdot v); \tag{1.11}$$

$$\tau_{rz} = \mu_e\left(\frac{\partial v_z}{\partial r} + \frac{\partial v_r}{\partial z}\right); \tau_{r\theta} = \mu_e\left(r\frac{\partial(v_\theta/r)}{\partial r} + \frac{1}{r}\frac{\partial v_r}{\partial\theta}\right); \tau_{z\theta} = \mu_e\left(\frac{\partial v_\theta}{\partial z} + \frac{1}{r}\frac{\partial v_z}{\partial\theta}\right), \tag{1.12}$$

μ_e – ефективна в'язкість при зсувній течії, ν – кінематична в'язкість.

$$\mu_e = \mu_0(I_2/2)^{\frac{n-1}{2}} e^{-\beta(T-T_0)}, \quad n=2 \rightarrow \ \mu_e = \mu_0\sqrt{I_2/2}e^{-\beta(T-T_0)}, \tag{1.13}$$

$$(\nabla\cdot v) = \frac{1}{r}\frac{\partial(rv_r)}{\partial r} + \frac{1}{r}\frac{\partial v_\theta}{\partial\theta} + \frac{\partial v_z}{\partial z}.$$

де μ_0 – початкова в'язкість; I_2 – другий інваріант тензору швидкостей деформації; n – показник аномалії в'язкості; v_r, v_θ, v_z – компоненти вектору швидкості; τ_{ij} $(i,j = r,\theta,z)$– компоненти тензору напружень; T – температура, β – температурний коефіцієнт; T_0 – початкова температура.

Після заміщення виразів для компонент тензору напружень зсуву у рівняння (1.5)–(1.8) отримуємо таку систему рівнянь для компонент швидкості руху і температури:

$$\rho\left(\frac{\partial v_r}{\partial t} + v_r\frac{\partial v_r}{\partial r} + \frac{v_\theta}{r}\frac{\partial v_r}{\partial\theta} - \frac{v_\theta^2}{r} + v_z\frac{\partial v_r}{\partial z}\right)$$

$$= -\frac{\partial p}{\partial r} + \mu_e\left[\Delta v_r + \frac{1}{r}\frac{\partial}{\partial\theta}\left(\frac{\partial v_\theta}{\partial r} - \frac{2}{r}v_\theta\right) + \frac{\partial^2 v_z}{\partial r\partial z}\right] + \rho g_r; \tag{1.14}$$

$$\rho\left(\frac{\partial v_\theta}{\partial t} + v_r\frac{\partial v_\theta}{\partial r} + \frac{v_\theta}{r}\frac{\partial v_\theta}{\partial\theta} - \frac{v_r v_\theta}{r} + v_z\frac{\partial v_\theta}{\partial z}\right) =$$

$$= -\frac{\partial p}{\partial\theta} + \mu_e\left[\Delta v_\theta + \frac{\partial}{\partial\theta}\left(r\frac{\partial v_r}{\partial r} + \frac{v_r}{r} + \frac{\partial v_z}{\partial z}\right) - v_\theta + \frac{\partial v_r}{\partial\theta}\right] + \rho g_\theta; \tag{1.15}$$

$$\rho\left(\frac{\partial v_z}{\partial t}+v_r\frac{\partial v_z}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_z}{\partial \theta}+v_z\frac{\partial v_z}{\partial z}\right)$$

$$=-\frac{\partial p}{\partial z}+\mu_e\left[\Delta v_z+\frac{1}{r}\frac{\partial^2 v_r}{\partial r\partial z}+\frac{1}{r}\frac{\partial}{\partial z}\left(\frac{\partial v_r}{\partial r}+\frac{\partial v_\theta}{\partial \theta}\right)\right]+\rho g_z; \qquad (1.16)$$

рівняння енергії:

$$c_V\rho\left(\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+\frac{v_\theta}{r}\frac{\partial T}{\partial \theta}+v_z\frac{\partial T}{\partial z}\right)$$
$$=\lambda_T\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2}+\frac{\partial^2 T}{\partial z^2}\right)+Aq_T; \qquad (1.17)$$

$$q_T=-T\frac{\partial p}{\partial T}\bigg|_\rho\left[\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{1}{r}\frac{\partial v_\theta}{\partial \theta}+\frac{\partial v_z}{\partial z}\right]+2\mu_e\left[\left(\frac{\partial v_r}{\partial r}\right)^2+\frac{1}{r^2}\left(\frac{\partial v_\theta}{\partial \theta}+v_r\right)^2+\right.$$

$$\left.+\left(\frac{\partial v_z}{\partial z}\right)^2+\left(\frac{\partial v_\theta}{\partial z}+\frac{1}{r}\frac{\partial v_r}{\partial \theta}\right)\cdot\left(\frac{\partial v_\theta}{\partial r}-\frac{v_\theta}{r}+\frac{1}{r}\frac{\partial v_r}{\partial \theta}\right)+\left(\frac{\partial v_z}{\partial r}+\frac{\partial v_r}{\partial z}\right)^2+\left(\frac{\partial v_\theta}{\partial z}+\frac{1}{r}\frac{\partial v_z}{\partial \theta}\right)^2\right]. \qquad (1.18)$$

A – термічний еквівалент роботи; c_v – питома тепломісткість рідини за сталого об'єму. Вираз $(\partial p/\partial T)|_\rho$ визначається із рівняння стану. У разі $p=\rho RT$ він дорівнює $R\rho$.

Що стосується ефективної в'язкості μ_e, слід зауважити, що вираз (1.13) містить експоненціальну складову як функцію T із від'ємним знаком та коефіцієнтом β, а також вираз для другого інваріанту тензору напружень, який є функція похідних від компонент швидкості потоку. Урахування такої залежності в явному вигляді занадто ускладнює постановку задачі та її розв'язання. На наш погляд, доцільним є врахування залежності μ_e від вказаних параметрів у вигляді інтегральної характеристики.

Межові умови:

– на вході – температура полімеру на вході (T_b) і нульовий відносний

тиск;

– на поверхні шнеку – окружна швидкість, що відповідає числу обертів шнека (N_s), температура поверхні шнеку (T_p);

– межові умови корпусу шнека – температура корпусу (T_k);

– межові умови на виході– температура полімеру на виході (T_{out}) і нульовий відносний тиск.

Таке чи подібне формулювання математичної моделі темпломасоперенесення наводиться у майже всіх працях, присвячених дослідженню процесів при екструзії полімерів.

Щоб не повторюватись при огляді математичних моделей у численних працях, в яких розглядаються ці питання, розглянемо один із типових підходів. [164].

Стверджується, що дифузійні процеси уздовж осі каналу зневажливо малі, оскільки довжина каналу суттєво більша за поперечні розміри шнеку. Гвинтовий канал розгорнуто на площину (рис. 2.2). Тоді модель руху і теплообміну розглядається у вигляді системи рівнянь на площині

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0;$$

$$2\frac{\partial}{\partial x}\left(\mu_e\frac{\partial v_x}{\partial x}\right)+\frac{\partial}{\partial y}\left[\mu_e\left(\frac{\partial v_x}{\partial y}+\frac{\partial v_y}{\partial x}\right)\right]=\frac{\partial P}{\partial x},$$

$$2\frac{\partial}{\partial y}\left(\mu_e\frac{\partial v_y}{\partial y}\right)+\frac{\partial}{\partial x}\left[\mu_e\left(\frac{\partial v_x}{\partial y}+\frac{\partial v_y}{\partial x}\right)\right]=\frac{\partial P}{\partial y},$$

$$\frac{\partial}{\partial x}\left(\mu_e\frac{\partial v_z}{\partial x}\right)+\frac{\partial}{\partial y}\left[\mu_e\frac{\partial v_z}{\partial y}\right]=\frac{\partial P}{\partial z},$$

$$\rho C\left(\bar{V}_z\frac{\partial T}{\partial z}+v_x\frac{\partial T}{\partial x}+v_y\frac{\partial T}{\partial y}\right)=\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\Phi,$$

де v_x, v_y, v_z – компоненти швидкості руху розплаву полімеру; $\overline{V}_z$ – середня швидкість (для твердої фази – це швидкість пробки U, для розплаву полімеру – середня швидкість у розплаві полімеру); Φ – функція дисипації; P – тиск; μ_e – ефективна в'язкість розплаву полімеру, що є функція швидкості зсуву та температури.

Така система рівнянь, на думку авторів, сповна описує процеси руху і теплообміну в екструзійній машині із урахуванням фазового переходу твердий полімер – рідинний полімер.

Це спрощуюче припущення властиве більшості праць з цього напрямку [115–121,126,145,147,151,160,161,164,171–174].

Далі робиться посилання на ``стандартні пакети" програм типу ANSYS [177] і наводяться результати моделювання за допомогою цих пакетів. Але ці пакети розраховані на розв'язання типових задач теплопровідності і для моделювання складних нелінійних задач тепломасоперенесення не пристосовані.

Загальною рисою результатів такого ``моделювання" є одновимірні графіки температури полімеру і зовсім відсутні результати стосовно фазових переходів тверде тіло – рідина.

1.5 Реологія полімерів

Неньютонівська рідина – це рідина, крива течії якої нелінійна, тобто в'язкість неньютонівської рідини не є стала за заданих температури і тиску, а залежить від інших факторів, таких як швидкість зсуву.

Реальні рідини із нелінійною кривою течії можна поділити на три групи:

1) системи, для яких швидкість зсуву у кожній точці являє собою деяку функцію тільки від напруження зсуву у цій точці;

2)складніші системи, в яких зв'язок між напруженням і швидкістю зсуву

залежить від часу дії напруження або передісторії рідини;

3) системи, які мають властивості як твердого тіла, так і рідини, і частково проявляється пружне відновлення форми після зняття напруженості (так звані *в'язкопружні рідини*).

Системи першого типу, властивості яких не залежать від часу, можна описати реологічним рівнянням

$$\dot{\gamma} = f(\tau),$$

із якого випливає, що швидкість зсуву у кожній точці є функція напруження зсуву у цій самій точці. Такі речі можна назвати *в'язкими неньютонівськими рідинами*. Залежно від вигляду функції $f(\tau)$ їх підрозділяють на три групи:

1. бінгамовські пластичні рідини (бінгамовські пластини);

2. псевдопластичні рідини (псевдопластики);

3. ділатантні рідини.

Псевдопластики.

Псевдопластичні рідини не виявляють границі текучості, й їхня крива течії показує, що відношення напруження зсуву до швидкості зсуву, тобто уявна в'язкість η_a

$$\eta_a = \tau/\dot{\gamma}, \quad \dot{\gamma} = v_z r,$$

поступово знижується зі зростанням швидкості зсуву. Крива течії стає лінійною тільки за дуже великих швидкостях зсуву. Граничний нахил, що отримав назву в'язкості за нескінченно великого зсуву, позначають як η_∞.

Для опису рідин такого типу застосовують функціональну емпіричну залежність у вигляді степеневого закону:

$$\tau = m\dot{\gamma}^n,$$

де m – міра консистенції рідини (чим вища в'язкість рідини, тим більше m); n -- характеризує ступінь неньютонівської поведінки матеріалу (чим більше n

відрізняється від одиниці, тим чіткіше проявляються його неньютонівські властивості).

Найзагальніше реологічне рівняння для рідини, що не стискається, має вигляд (Рейнер):

$$\tau = \eta \cdot \nabla + \frac{1}{2}\eta_c(\Delta : \Delta),$$

де η_c – коефіцієнт поперечної в'язкості, що є скалярна функція інваріантів I_1, I_2.

Степеневий закон у загальному вигляді можна записати як:

$$\eta = \eta_0 [I_2/2]^{\frac{n-1}{2}},$$

де η_0 – в'язкість при $\dot{\gamma} = 1(1/c)$.

Скалярна величина $(\Delta : \Delta)$ у циліндричних координатах:

$$I_2 = 2\left[\left(v_r r\right)^2 + \left(\frac{1}{r}v_\theta\theta + \frac{v_r}{r}\right)^2\right] + \left[rr\left(\frac{v_\theta}{r}\right) + \frac{1}{r}v_r\theta\right]^2 +$$

$$+\left(\frac{1}{r}v_z\theta + v_\theta z\right)^2 + \left(v_r z + v_z r\right)^2.$$

1.5.1 Залежність в'язкості від температури

Зазвичай в'язкість рідини зі збільшенням температури зменшується, а в'язкість газів збільшується.

Поведінка неньютонівських рідин під час течії залежить від однієї характеристики -- в'язкості, що є функція тільки температури. Найчастіше використовують вираз для залежності в'язкості від температури у вигляді рівняння Арреніуса

$$\mu = Ae^{E/RT},$$

де R – газова стала; E -- енергія активації течії; A – коефіцієнт, що залежить

від природи рідини.

Іноді для уточнення даних залежності в'язкості від температури застосовують інше рівняння:

$$\mu = ae^{-bT},$$

де a,b – емпіричні коефіцієнти.

Значення енергії активації течії $E_{\dot{\gamma}}$ і E_{τ} для поліетилену у температурному інтервалі 108--230 $^{\circ}C$ визначені у []; Енергія активації течії для поліетилену у температурному інтервалі 108–230 $^{\circ}C$

$\dot{\gamma}$,1/с	$E_{\dot{\gamma}}$, кДж/моль	τ, Н/м^2	E_{τ}
0	53,6	0	53,6
10^{-1}	47,7	10^3	62,8
10^0	43,12	10^4	72,5
10^1	35,6	10^5	79,55
10^2	30,14	-	-
10^3	25,54	-	

1.5.2 В'язкопружні рідини

В'язкопружними називають рідини, що проявляють як пружні (пружне відновлення форми), так і в'язкі властивості(в'язка течія).

Модель Кельвіна–Фойгта

Тіло можна подати таким, що складається із пружного та в'язкого елементів, що з'єднані паралельно. Під впливом сталого напруження тіло почне деформуватися, тобто спочатку пружина розтягнута незначно, її реакція відносно мала і більша частина напруження припадає на в'язкий елемент. Для

отримання реологічного рівняння тіла Кельвіна–Фойгта розглянемо геометричний, кінематичний і фізичний аспекти задачі. Із геометричного розгляду задачі маємо

$$\tau_\Sigma = \tau_N + \tau_H;$$

кінематичного:

$$\gamma_\Sigma = \gamma_h = \gamma_N;$$

фізичного:

$$\tau_H = \gamma_H G = \gamma_\Sigma G; \quad \tau_N = \mu\dot{\gamma}_N = \mu d\gamma_\Sigma/dt.$$

Тоді

$$\tau_\Sigma = \gamma_H G + \mu\dot{\gamma}_N = \gamma_H G + \mu\frac{d\gamma_\Sigma}{dt}, \quad \frac{\tau}{G} = \gamma_\Sigma + \frac{\mu}{G}\dot{\gamma}.$$

Цей вираз є реологічне рівняння простої моделі Кельвіна–Фойгта.

Для дослідження поведінки простої моделі Кельвіна--Фойгта за довільних режимів навантаження необхідно розв'язати останнє диференційне рівняння. Для цього розглянемо режим деформування за сталого напруження (повзучість), тобто $\tau = \tau_0 = const$. Тоді

$$\frac{d\gamma_\Sigma}{dt} = \frac{\tau_0}{\mu} - \frac{\lambda_\Sigma}{\mu}G = -\frac{G}{\mu}\left(\gamma_\Sigma - \frac{\tau_0}{G}\right),$$

або

$$\frac{1}{\gamma_\Sigma - \tau_0/G}d(\gamma_\Sigma - \tau_0/G) = -\frac{G}{\mu}dt \rightarrow$$

$$\ln\frac{\gamma_\Sigma - \tau_0/G}{\gamma_0 - \tau_0/G} = -\frac{G}{\mu}t\ln e, \rightarrow \gamma_\Sigma = \gamma_0 e^{-G/\mu\cdot t} + \frac{\tau_0}{G}\left(1 - e^{-G/\mu\cdot t}\right).$$

Це рівняння характеризує змінювання величини деформації у часі за сталого напруження. Із цього рівняння випливає, що за $t = 0$, $\gamma = \gamma_0$, а за $t = \infty$ $\gamma_\Sigma = \tau_0/G$, тобто модель К-Ф веде себе як модель Гука. Величина $t_z = \mu/G$, що має розмірність часу, називається часом запізнення.

n-й елемент характеризується модулем зсуву G_n і в'язкістю μ_n. Тоді час запізнення цього елементу $\lambda_{zp} = \mu_n / G_n$. Із урахуванням того, що $\gamma_0 = \tau_0 / G_n$, отримаємо:

$$\gamma_n(t) = \frac{\tau_0}{G_n}\left(1 - e^{-t/\lambda_{zp}}\right).$$

Повний зсув системи γ буде сумою деформацій окремих елементів:

$$\gamma(t) = \tau_0 \sum_{n=1}^{N}\left[\frac{1}{G_n}\left(1 - e^{-t/\lambda_{zp}}\right)\right].$$

Зазвичай прийнято позначати $I_n = 1/G_n$ – податливість зсуву.

$$\gamma(t) = \tau_0 \sum_{n=1}^{N}\left[I_n\left(1 - e^{-t/\lambda_{zp}}\right)\right].$$

При $n \to \infty$ приходимо до нескінченного ланцюга елементів, для яких час запізнення змінюється неперервно від нуля до ∞. Скінченне число сталих для елементів I_n, λ_{zp} замінюється функцією розподілу $I(\lambda_{zp})$, яка являє собою пружну податливість, що зв'язана із часом запізнення як неперервним параметром. Зазвичай цю функцію називають *розподілом часу запізнення*. При переході до границі $N \to \infty$ маємо

$$\gamma(t) = \tau_0 \int_0^{\infty} I(\lambda_{zp})\left(1 - e^{-t/\lambda_{zp}}\right) d\lambda_{zp}.$$

Для узагальнення моделі запроваджують поняття *функції повзучості* в'язкопружного матеріалу, яка визначається як відношення деформації, що виражена у вигляді функції від t, до напруження за умови, що до матеріалу після релаксації раптово прикладене стале за величиною напруження.

$$\varphi(t) = \gamma(t) / \tau_0.$$

Тоді

$$\varphi(t) = \int_0^{\infty} I(\lambda_{zp})\left(1 - e^{-t/\lambda_{zp}}\right) d\lambda_{zp}.$$

Тепер розподіл часу запізнення $I(\lambda_{zp})$ можна визначити на основі повзучості $\varphi(t)$, отриманої експериментальним шляхом.

1.6 Висновки та постановка задач дослідження

У першому розділі роботи проведено аналіз сучасного стану проблеми математичного моделювання процесів тепломасоперенесення при екструзійній пластикації полімерів.

В результаті проведеного аналізу виявлено, що у зв'язку із відсутністю можливостей проведення натурних (експериментальних) досліджень на існуючих екструзійних лініях неперервного виробництва та їхньою значною коштовністю для визначення оптимальних режимів і параметрів устаткування є доцільне математичне моделювання процесів тепломасоперенесення.

Аналіз існуючих підходів до математичного моделювання процесів тепломасоперенесення засвідчив, що переважна більшість цих підходів грунтується на опису процесів в екструдері за допомогою класичних рівнянь Нав'є--Стокса у лінійній постановці із урахуванням конвективних складових. Слід також зазначити, що навіть у такому підході до математичного моделювання процесів в екструзійних машинах майже всі результати отримуються або чисельними методами, або півчисельними (метод Галеркіна із застосуванням методів скінченних елементів).

На підставі проведеного аналізу доцільно запропонувати методи чисельно-аналітичного розв'язання задач:

– сформулювати математичні моделі, що описують процеси у зонах завантаження, затримки плавлення, плавлення і дозування полімерів як ізоляційне покриття кабелів надвисокої напруги;

– розробити методи чисельно-аналітичного розв'язання відповідних

нелінійних крайових задача, що описують процеси у відповідних зонах шнекового формування ізоляційного покриття;

– розробити відповідне математичне і програмне забезпечення алгоритмів вирішення цих задач;

– розробити методи оптимального управління процесами тепломасоперенесення у відповідних зонах екструдера.

Розділ 2 ФОРМУЛЮВАННЯ МОДЕЛЕЙ ПРОЦЕСІВ

2.1 Огляд підходів до моделювання [43]

Формулюванню математичних моделей цих процесів присвячена значна кількість наукових праць науковців. Із них, як найпоказовіші відокремимо праці [5,6,11,114,121,126,129,143,183,204,223,224,240]. Загальний підхід до формулювання математичних моделей у зоні плавлення полягає у такому.

Запроваджуються припущення: процес стаціонарний зі сталим масовим розходом; гвинтовий канал розгортається на площину (рис. 2.2) і використовується зворотний рух; перенесення тепла вздовж каналу здійснюється за рахунок конвективної складової, тому дифузійні процеси за цією координатою не враховуються; пружні процеси у розплаві не враховуються; градієнтами складових швидкостей у напрямку осі нехтують, оскільки довжина каналу на два-три порядки більша за ширину; а його геометрія за довжиною каналу є стала; інерційні і масові сили є малі порівняно із силами в'язкого тертя. Із урахуванням цих обмежень процес руху і теплообміну полімеру у гвинтовому каналі екструдера моделюється у прямокутному каналі, верхня стінка якого рухається зі сталою швидкістю, що дорівнює окружній швидкості черв'яка під кутом нарізки гвинтової лінії до осі каналу θ. Із урахуванням зазначених припущень система диференційних

рівнянь для розплаву полімеру описується у декартовій системі координат.

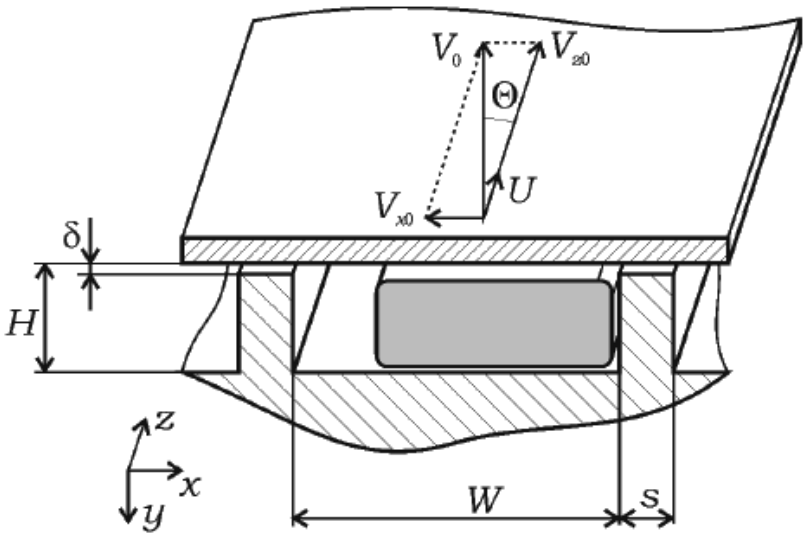


Рис. 2.2 Схема розгортки гвинтового каналу

Із урахуванням цих допущень система рівнянь, що описує рух і теплообмін полімеру у каналі шнека, має вигляд:

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}=0;$$

$$2\frac{\partial}{\partial x}\left(\mu_e\frac{\partial v_x}{\partial x}\right)+\frac{\partial}{\partial y}\left[\mu_e\left(\frac{\partial v_x}{\partial y}+\frac{\partial v_y}{\partial x}\right)\right]=\frac{\partial P}{\partial x},$$

$$2\frac{\partial}{\partial y}\left(\mu_e\frac{\partial v_y}{\partial y}\right)+\frac{\partial}{\partial x}\left[\mu_e\left(\frac{\partial v_x}{\partial y}+\frac{\partial v_y}{\partial x}\right)\right]=\frac{\partial P}{\partial y},$$

$$\frac{\partial}{\partial x}\left(\mu_e\frac{\partial v_z}{\partial x}\right)+\frac{\partial}{\partial y}\left[\mu_e\frac{\partial v_z}{\partial y}\right]=\frac{\partial P}{\partial z},$$

$$\rho C\left(\overline{V}_z\frac{\partial T}{\partial z}+v_x\frac{\partial T}{\partial x}+v_y\frac{\partial T}{\partial y}\right)=\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)+\Phi,$$

де v_x, v_y, v_z – компоненти швидкості руху розплаву полімеру; $\overline{V}_z$ – середня швидкість (для твердої фази – це швидкість пробки U, для розплаву полімеру – середня швидкість у розплаві полімеру); Ф – функція дисипації; P – тиск; μ_e

– ефективна в'язкість розплаву полімеру, що є функція швидкості зсуву та температури.

Ефективна в'язкість визначається

$$\mu_e = \mu_0 e^{-\beta(T-T_0)} (I_2/2)^{(n-1)/2},$$

де β – температурний коефіцієнт в'язкості; n – показник аномалії в'язкості; I_2 – другий інваріант тензору швидкостей деформації; μ_0 – коефіцієнт консистенції розплаву полімеру.

Функція дисипації обчислюється за формулою:

$$\Phi = \mu_e I_2/2.$$

У якості межової умови з температури на вході у канал використовується температура гранул полімеру із завантажуючого бункера. На внутрішній поверхні корпусу задається розподіл температури, що визначається технологічними умовами перероблення полімерного матеріалу. Температура на шнеці визначається за допомогою ітераційної процедури при сумісному розв'язанні цієї задачі та задачі по визначенню температури шнеку. Межа розділу фаз визначається ізотермою, що відповідає деякій середній (в інтервалі фазових перетворень)

Загальною рисою такого підходу є прагнення запобігти необхідності мати справу із циліндричною системою координат, що, своєю чергою, призодвить до оперування циліндричними функціями, для яких, мабуть, немає ефективних методів чи алгоритмів розв’язання відповідних крайових задач.

Розглянемо інший, але теж розповсюджений підхід до математичного моделювання теплоперенесення при обробленні полімерних матеріалів в екструдерах.

Вирішення задачі по визначенню впливу охолодження шнеку на процес екструзії може бути здійснено на ґрунті математичної моделі, що описує процеси тепломасоперенесення розплавів полімеру та охолоджуючого агенту із

різними реологічними і теплофізичними властивостями у каналах шнеку.

Математичний опис процесів течії і теплообміну розплаву полімеру та рідини, що охолоджує, ґрунтується на законах збереження. Рівняння енергії, руху, нерозривності отримано за таких припущень [39]:

- середовище не стискається, без пружних властивостей;
- полімер надходить у канал із торця шнеку у рідинному стані;
- течія осесиметрична;
- теплофізичні характеристики є сталі;
- полімер рухається вздовж шнека;
- поверхня шнека без гребенів.

Система диференційних рівнянь для кожного із потоків набуває вигляду:

$$\frac{1}{r}\frac{\partial r v_r^i}{\partial r}+\frac{\partial v_z^i}{\partial z}=0; \tag{2.1}$$

$$\frac{\partial v_r^i}{\partial t}+\rho\left(v_r^i\frac{\partial v_r^i}{\partial r}+v_z^i\frac{\partial v_r^i}{\partial z}\right)=-\frac{\partial P}{\partial r}$$

$$+\frac{\partial}{\partial r}\left(2\mu_e^i\frac{\partial v_r^i}{\partial r}\right)+\frac{2\mu_e^i}{r}\frac{\partial v_r^i}{\partial r}-\frac{2}{r^2}\mu_e^i v_r^i+\frac{\partial}{\partial z}\left[\mu_e^i\left(\frac{\partial v_z^i}{\partial r}+\frac{\partial v_r^i}{\partial z}\right)\right]; \tag{2.2}$$

$$\frac{\partial v_z^i}{\partial t}+\rho\left(v_r^i\frac{\partial v_z^i}{\partial r}+v_z^i\frac{\partial v_z^i}{\partial z}\right)=-\frac{\partial P}{\partial z}$$

$$+\frac{\partial}{\partial r}\left[\mu_e^i\left(\frac{\partial v_z^i}{\partial r}+\frac{\partial v_r^i}{\partial z}\right)\right]+\frac{\mu_e^i}{r}\frac{\partial}{\partial z}\left(\frac{\partial v_z^i}{\partial r}+\frac{\partial v_r^i}{\partial z}\right)+\frac{\partial}{\partial z}\left(\mu_e^i\frac{\partial v_z^i}{\partial z}\right); \tag{2.3}$$

$$\rho^i C^i\left(\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+v_z\frac{\partial T}{\partial z}\right)=\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda^i\frac{\partial T}{\partial r}\right)+\frac{\partial}{\partial z}\left(\lambda^i\frac{\partial T}{\partial z}\right)+\Phi; \tag{2.4}$$

де індекс i визначає матеріал; r,z – радіальна і поздовжня циліндричні координати; v_r. v_z – компоненти вектору швидкості; ρ – густина; C – тепломісткість; μ_e – ефективна в'язкість; λ – теплопровідність, що є функція швидкості зсуву і температури:

$$\mu_e = \mu_0 e^{-\beta((T-T_0)} (I_2/2)^{(n-1)/2}, \quad (2.5)$$

де μ_0 – початкова в'язкість за T_0, β – температурний коефіцієнт в'язкості; I_2 – другий інваріант тензору швидкостей деформації.

Система рівнянь (1)–(4) замикається такими межовими умовами:

– не нерухомих стінках компоненти швидкості дорівнюють нулю;

– на вході у канали задаються епюри швидкостей, що відповідають заданим витратам матеріалу;

– на виході – межові умови другого роду за швидкістю і температурою;

– температура стінки циліндру зростає за градієнтом від $90^{\circ}C$ до $200^{\circ}C$;

– температура полімерів на вході в екструдер стала – $20^{\circ}C$;

– на поверхнях контакту задаються межові умови 4-го роду і рівність температур.

Полімер:

$$v_z|_{z=0} = f_1(r),\ v_r|_{z=0} = 0, \quad (2.6)$$

$$\frac{\partial v_z}{\partial z}|_{z=L_{\text{øí}}} = 0,\ v_z|_{r=R_{\text{øí}}}\ v_r|_{r=R_{\text{øí}}} = 0, \quad (2.7)$$

$$v_r|_{r=R_c} = 0,\ v_z|_{r=R_c} = 0, \quad (2.8)$$

$$\lambda_p \frac{\partial T}{\partial r}|_{r=R_{\text{øí}}} = \lambda_m \frac{\partial T}{\partial r}|_{r=R_{\text{øí}}}, \quad (2.9)$$

$$T^+|_{r=R_{\text{øí}}} = T^-|_{r=R_{\text{øí}}}, \quad (2.10)$$

$$T|_{z=0} = T_{\text{âõ}},\ T|_{r=R_c} = f(z), \quad (2.11)$$

$$\frac{\partial T}{\partial z}|_{z=L_{\text{øí}}} = 0, \quad (2.12)$$

Шнек (метал):

$$\frac{\partial T}{\partial z}|_{z=0} = 0,\ \frac{\partial T}{\partial z}|_{z=L_{\text{øí}}} = 0. \quad (2.13)$$

Канал охолодження:

$$v_z\,|_{z=0} = f_2(r),\ \frac{\partial v_r}{\partial r}|_{r=0} = 0,\ \frac{\partial v_z}{\partial r}|_{r=0} = 0. \qquad (2.14)$$

Межові умови на твердих поверхнях для компонент швидкості агенту, що охолоджує, відповідають умовам прилипання і непроникнення.

У центрі каналу охолодження виконуються умови):

$$\lambda_{\text{îõ}} \frac{\partial T}{\partial r}|_{r=R_k} = \lambda_m \frac{\partial T}{\partial r}|_{r=R_k}, \qquad (2.15)$$

$$T^+\,|_{r=R_k} = T^-\,|_{r=R_k}, \qquad (2.16)$$

$$T\,|_{z=0} = T_{\text{âõ}},\ \frac{\partial T}{\partial z}|_{z=L_{\text{øí}}} = 0. \qquad (2.17)$$

У якості агенту охолодження використовується вода із $T = 10^{\circ}C$. Тиск агенту $P = 1000$ Па. Вода подається по центру. На рис. 2.1, 2.2, 2.3 [43], наведено залежність температур у полімері за різної глибини введення агенту, що охолоджує. На рис. позначено: 1 – без охолодження; 2 – глибина введення 10%; 3 – 30%; 4 – 50%; 5 – 70%; 6 – 90%. Зауважимо, що ці результати автори [39] отримали, як випливає із представлених графіків, без урахування наведеної вище системи диференційних рівнянь із межовими умовами, за допомогою пакету ANSYS.

У результаті введення агенту на глибину від 50 до 90% вдається знизити перегріви полімеру (температура знижується на $20^{\circ}C$) і вирівняти температуру у каналі при збереженні заданих технологічних параметрів.

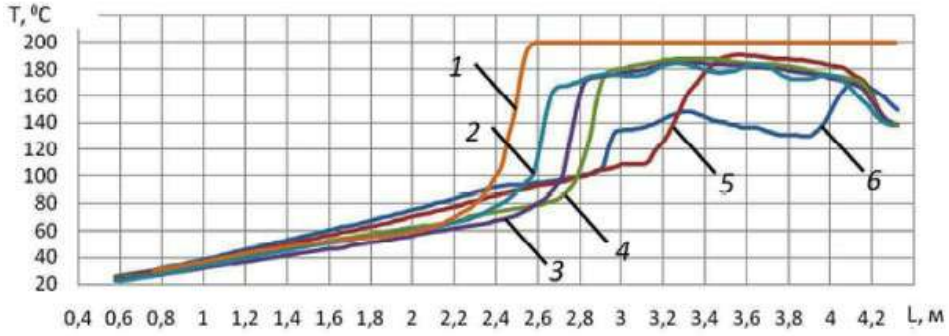


Рис. 2.1 Розподіл мінімальної температури у полімері

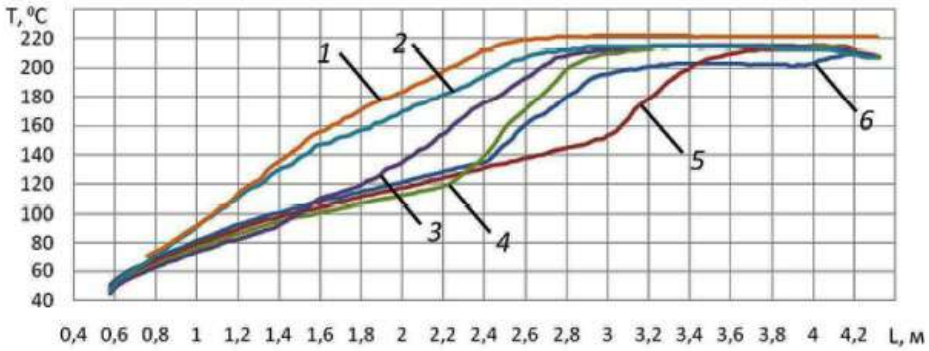


Рис. 2.2 Розподіл середньої температури у полімері

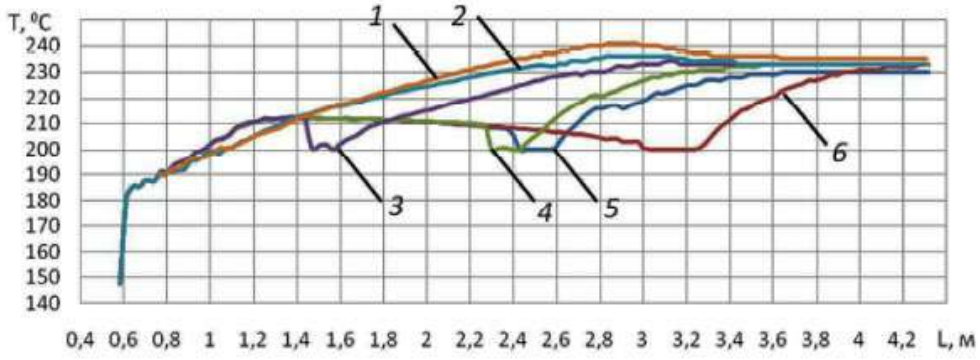


Рис. 2.3 Розподіл максимальної температури у полімері

Кожна зона нагріву циліндру має довжину 350–450 мм. Стандартів по налагодженню температури плавлення не існує. Єдина умова – створення на першій та іноді на другій зоні температур, близьких до показників, які розплав має набути на виході. Підбір температури залежить від особливостей полімеру і конструкції шнека. Регулювання нагріву здійснюється в умовах ефективної тепловіддачі для запобігання перегрівання.

Контролювати температуру розплаву недоцільно. У гвинтовому каналі шнеку завжди спостерігається перепад температурних значень. Вимірювати температуру можна лише в одній точці. Тож контролювати краще температуру, що надходить із корпусу. Контроль нагріву циліндру здійснюється за допомогою спеціальних датчиків. Фактично вимірюється температура поверхні циліндру, тому варто ще враховувати і період стабілізації реального впливу на розплав.

У зоні завантаження шнеку температура має досягати $50^{\circ}C$. Перша зона терморегуляції відповідає відрізку шнека, на якому здійснюється внутрішня течія матеріалу, тертя із циліндром і самим шнеком. Тут максимальний вплив на протискання твердого матеріалу. Головною умовою є прилипання полімеру до циліндричної поверхні та його прослизання по шнеку. Для поліолефелінів на даному відрізку задається температура більше $150^{\circ}C$.

У другій зоні температура має бути вище на 50–$80^{\circ}C$ порівняно із першою зоною.

На ділянці гомогенізації температура може бути у середньому на $10^{\circ}C$ нижче, ніж потрібно для визначеного виду полімеру. У перехідному режимі і у зоні головки також температура знижується на 5–$15^{\circ}C$ відповідно.

Як видно із наведених результатів цієї праці, формулювання математичної моделі у вигляді системи рівнянь Нав'є–Стокса із урахуванням конвективного перенесення матеріалу вважається такою, що описує процеси в усіх зонах екструдера однією і тією самою системою рівнянь, що не відповідає реальним ситуаціям. Далі, ці результати свідчать про те, що ніякого математичного моделювання системи рівнянь типу Нав'є–Стокса не виконувалося, навіть із застосуванням відомих пакетів, про щосвідчать наведені графіки температури.

Опису математичних моделей процесів в екструзійному обладнанні присвячено значну кількість наукових публікацій [127, 154, 159, 169, 171—173]. У переважній більшості цих праць розглядаються математичні моделі стаціонарних процесів у зонах завантаження і плавлення і дозування полімерних сумішей. На наш погляд, це не відповідає сутності процесів, що мають місце у цих зонах при екструзійному виробництві полімерних виробів. Не вдаючись у докладний огляд всіх запропонованих моделей процесів в екструзійному обладнанні, наведемо один із типових підходів до формування

нестаціонарних математичних моделей у зонах завантаження [36,103].

Математична модель процесу ``нагрів–охолодження" полістиролу під час переміщення його по екструдеру подається у вигляді системи диференційних рівнянь нестаціонарної теплопровідності у системі сполучених тіл різної фізичної природи, в яких в якості управляючих впливів розглядаються розподілені вздовж осьової координати екструдера внутрішні джерела тепла.

$$\frac{\partial T_1(r,x,t)}{\partial t} = a_1\left[\frac{\partial^2 T_1(r,x,t)}{\partial r^2} + \frac{1}{r}\frac{\partial T_1(r,x,t)}{\partial r} + \frac{\partial^2 T_1(r,x,t)}{\partial x^2}\right] + W(r,x,t), \quad (2.18)$$

$$r \in [r_1, r_2],\ x \in [0, L];$$

$$\frac{\partial T_2(r,x,t)}{\partial t} + V(x)\frac{\partial T_2(r,x,t)}{\partial x} = a_2\left[\frac{\partial^2 T_2(r,x,t)}{\partial r^2} + \frac{1}{r}\frac{\partial T_2(r,x,t)}{\partial r} + \frac{\partial^2 T_2(r,x,t)}{\partial x^2}\right], \quad (2.19)$$

$$r \in [r_2, r_3],\ x \in [0, L];$$

$$\frac{\partial T_3(r,x,t)}{\partial t} = a_3\left[\frac{\partial^2 T_3(r,x,t)}{\partial r^2} + \frac{1}{r}\frac{\partial T_3(r,x,t)}{\partial r} + \frac{\partial^2 T_3(r,x,t)}{\partial z^2}\right],\ r \in [r_3, 0],\ x \in [0, L]. \quad (2.20)$$

Тут $T_1(r,x,t), T_2(r,x,t), T_3(r,x,t)$ – температура корпусу екструдера, полістиролу і шнека відповідно, $W(r,x,t)$ – функція розподілу потужності внутрішніх джерел тепла; a_1, a_2, a_3 – коефіцієнти температуропровідності; L – довжина нагрівача.

Початкові та межові умови задаються у вигляді:

$$\lambda_1 \frac{\partial T_1(r,x,t)}{\partial x}|_{x=0} = \alpha_1[T_1(r,x,t) - T_c(r,0,t)] \quad (2.21)$$

$$\lambda_1 \frac{\partial T_1(r,x,t)}{\partial x}|_{x=X} = \alpha_2[T_1(r,X,t) - T_c(r,X,t)],\ T_1(r,X,0) = T_{10}; \quad (2.22)$$

$$\lambda_3 \frac{\partial T_3(r,x,t)}{\partial x}|_{x=0} = \alpha_3[T_3(r,0,t) - T_c(r,0,t)],\ T_2(r,X,0) = T_{20}(r,t); \quad (2.23)$$

$$\lambda_3 \frac{\partial T_3(r,x,t)}{\partial x}|_{x=X} = \alpha_4[T_3(r,X,t) - T_c(r,X,t)],\ T_3(r,x,0) = T_{20}(r,x); \quad (2.24)$$

$$\lambda_3 \frac{\partial T_3(r,x,t)}{\partial r}|_{r=R_3} = \lambda_2 \frac{\partial T_2(r,x,t)}{\partial r}|_{r=R_3},\ T_3(R_3,x,t) = T_2(R_2,x,t); \qquad (2.25)$$

$$\lambda_2 \frac{\partial T_2(r,x,t)}{\partial r}|_{r=R_2} = \lambda_1 \frac{\partial T_1(r,x,t)}{\partial r}|_{r=R_2},\ T_1(R_2,x,t) = T_2(R_2,x,t); \qquad (2.26)$$

$$\lambda_1 \frac{\partial T_1(r,x,t)}{\partial r}|_{r=R_1} = \alpha_1[T_1(r,x,t) - T_c(R_1,x,t)],\ \frac{\partial T_1(r,x,t)}{\partial r}|_{r=0} = 0. \qquad (2.27)$$

Тут α коефіцієнт теплообміну; λ_1, λ_2, λ_3 – коефіцієнти теплопровідності матеріалу труби, полістиролу і шнека відповідно.

Аналогічна постановка задачі з визначення температурних полів у зонах завантаження і полістиролу наведена у [103].

Далі стверджується, що розрахунок нестаціонарного режиму теплопровідності системи сполучених тіл аналітичними методами неможливий, і пропонується варіант різницевого методу– метод скінченних елементів. Внутрішні джерела тепла розраховуються на підгрунтті розв'язання рівнянь Максвела.

Позначаючи A – векторний магнітний потенціал; $\mu_a = \mu \cdot \mu_0$; μ_0 – абсолютна магнітна проникненість середовища; J – питома електрична проводимість, враховуючи осьову симетрію та квазістаціонарність магнітного поля за $\mu = \text{const}$, рівняння для комплексної амплітуди векторного потенціалу подається у формі

$$\frac{\partial}{\partial z}\left[\frac{1}{\mu_a(r,z)}\frac{\partial \dot{A}(r,z)}{\partial z}\right] + \frac{\partial}{\partial r}\left[\frac{1}{\mu_a(r,z)}\frac{1}{r}\frac{\partial \dot{A}(r,z)}{\partial r}\right] - j\omega\gamma \dot{A}(r,z) - J_0(r,z) = 0. \qquad (2.28)$$

Далі стверджується, що на підгрунтті розв'язання крайової задачі методом скінченних елементів отримано основні розміри елементів екструдера і параметрів індукційної системи [25]. Джерело внутрішнього тепловиділення описується залежністю

$$W(r,x,t) = F(r)Q(\theta)G(x)U(t),$$

де $F(r), Q(\theta), G(x), U(t)$ – розподіл джерел тепла за відповідними змінними.

У наведеній моделі, як видно, обмежуються дослідженнями процесів нагріву корпусу (рівняння (2.18)), полістиролу (рівняння (2.19) та шнека (рівняння 2.20) із відповідним початковими та межовими умовами.

Математична модель процесу нагрівання корпусу екструдера на ділянці зони завантаження описується рівнянням теплопровідності (2.18) із межовими умовами (2.21), (2.22), при цьому відсутні умови контакту індуктора із зовнішньою поверхнею корпусу екструдера.

Рівняння (2.18) описує процес нагрівання суміші у зоні завантаження, але не врахована умова залежності коефіцієнту тепломісткості речовини від температури, що є суттєвим фактором під час нагрівання суміші.

У наведеній моделі відсутні постановки задач про процеси плавлення і кристалізації полімеру (у зоні дозування).

Крім того, що такий підхід до моделювання процесів плавлення полімерів не відповідає суті процесів, у цих підходах відсутній процес переходу від твердої суміші до розчину що, на наш погляд, є визначальним для забезпечення якості ізоляційного покриття кабельної жили.

Далі, зона дозування, в якій здійснюється процес кристалізації, зовсім не розглядається. Кількість праць, присвячених цьому питанню, досить обмежена. У цих працях питання кристалізації розчиненої субстанції розглядається без зв'язку із дозуванням полімерів.

2.2 Математична модель нагрівання корпусу шнека

2.2.1 Індукційне нагрівання

Основним джерелом нагріву корпусу екструдера вважається індукційний нагрів.

Густина внутрішніх джерел тепла є електромагнітна енергія, що

виділяється за одиницю часу в одиниці об'єму. У силу наявності поверхневого ефекту розподіл внутрішніх джерел тепла є суттєво неоднорідний і залежить від електрофізичних властивостей завантаження, які змінюються у процесі нагрівання.

При цьому весь процес нагрівання поділено на інтервали, у кожному із яких властивості завантаження приймаються незмінними.

$$L_i(m) = \{0{,}9; 1{,}45; 2{,}05; 2{,}65\}.$$

На рис. 2.1 наведено графік розподілу питомої об'ємної потужності за довжиною циліндру [36].

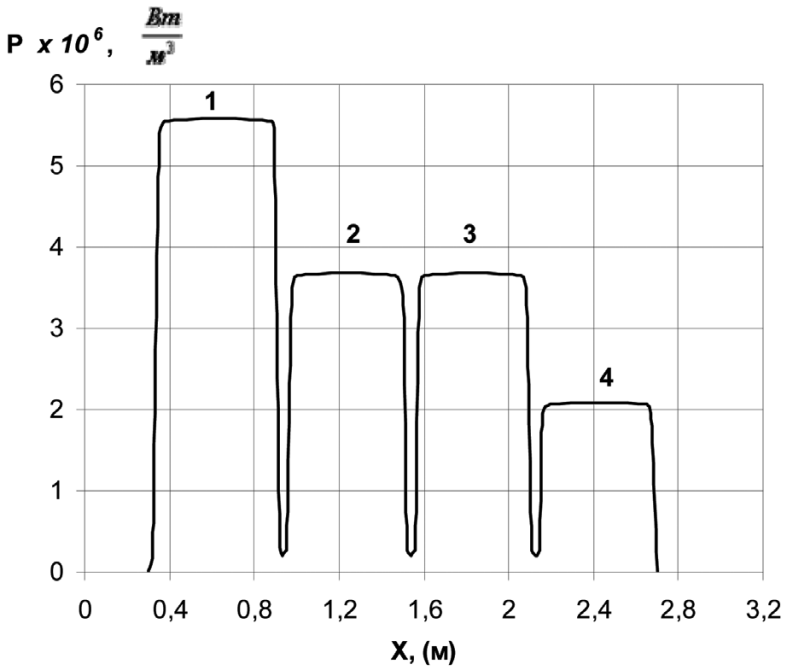


Рис. 2.4 Розподіл питомої об'ємної потужності за довжиною секцій 1—4

Із графіку видно, що на першій ділянці екструдера (перший індуктор) виділяється максимальна потужність, оскільки на цій ділянці необхідно нагріти полімер від початкової температури до температури плавлення, а на решті зон здійснюється підігрівання і підтримка заданої температури.

В [150] розглядаються математичні моделі індукційного нагрівання

корпусу екструдера, процесів нагрівання і плавлення полімерних сумішей.

Основним управляючим впливом в устаткуваннях індукційного нагрівання субстанцій є розподілені за об'ємом внутрішні джерела тепла, що індукуються електромагнітним полем індуктора. При цьому потрібно розділяти процедури розрахунку електромагнітного поля та теплового поля нагрівання субстанції, оскільки вони характеризуються різними інерційностями процесів. Тому електромагнітна задача розглядається як квазістаціонарна, що надає можливість створювати незалежні процедури розрахунків цих полів.

У загальному випадку процес індукційного нагрівання описується нелінійними рівняннями Максвела для електромагнітного поля із відповідними межовими умовами:

$$\mathrm{rot}H = \frac{1}{\rho}E + \frac{\partial D}{\partial t};\ \ \mathrm{rot}E = \frac{\partial B}{\partial t};\ \mathrm{div}H = 0;\ \mathrm{div}E = 0. \qquad (2.30)$$

Тут H, B, E, D – вектори напруженості та індукції магнітного та електричного полів.

Запровадження векторного потенціалу A із урахуванням осесиметричності геометрії шнеку рівняння для комплексної амплітуди векторного потенціалу можна записати як

$$\frac{\partial}{\partial z}\left[\frac{1}{\mu_a}\frac{\partial A}{\partial z}\right] + \frac{\partial}{\partial z}\left[\frac{1}{r}\frac{1}{\mu_a}\frac{\partial A}{\partial r}\right] - i\omega\gamma A(r,z) - J_0(r,z) = 0. \qquad (2.31)$$

Межові умови:

$$A|_{S=0} = 0,\ \frac{\partial A}{\partial n} = 0, \qquad (2.32)$$

В [150] запропоновано розв'язання цієї задачі на грунті варіаційних принципів шляхом мінімізації функціоналу:

$$F(A) = \frac{1}{2}\iint_Q \left[\frac{\partial}{\partial x}\left(\frac{1}{\mu_a}\frac{\partial A}{\partial x}\right) + \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{1}{\mu_a}\frac{\partial A}{\partial r}\right)\right] drdx$$

$$+\frac{1}{2}\iint_{Q} i\omega\gamma \mid A\mid^2 \, drdz + \frac{1}{2}\iint_{Q} J_0 A drdz. \qquad (2.33)$$

Така постановка задачі охоплює найзагальніші електромагнітні явища і надає можливість розраховувати клас пристроїв індукційного нагрівання, який може бути описаний двовимірним рівнянням Пуассона.

Параметри системи: швидкість руху полімеру у зоні завантаження $\theta_{zav} = 0{,}946$ см/с; Швидкість руху у зоні дозування $\theta_{zav} = 3{,}46$ см/с. Потужність індукційної системи 46 кВт; потужність за зонами: [18; 12;9;7] кВт.

2.2.2 Формулювання математичної моделі

Функція розподілу густини внутрішніх джерел енергії визначається шляхом розв'язання рівняння Пуассона відносно електромагнітного поля в індукторі. Цю задачу розглянуто у підрозділі 3.2, де розглядається індукційне нагрівання злитку у декартовій системі координат.

Суттєвим недоліком наведеної постановки задачі про нагрівання корпусу екструдера є відсутність на межі індуктор -- корпус умови променистого випромінювання, що призводить до викривлених значень енергії, яка передається корпусу екструдера від індуктора.

Розподіл температурного поля корпусу T_k описується рівнянням теплопровідності, яке у циліндричній системі координат має вигляд

$$\frac{\partial T_k}{\partial t} = a_k\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_k}{\partial r}\right)+\frac{\partial^2 T_k}{\partial z^2}\right)+\frac{1}{c_k\rho_k}g(r_4,z,t) \qquad (2.34)$$

де $a_k = \lambda_k/(c_k\rho_k)$ – коефіцієнт температуропровідності; λ_k, c_k, ρ_k – коефіцієнт теплопровідності (Вт/(м$^\circ C$), питома тепломісткість і густина матеріалу корпусу відповідно; $g(r,z)$ – функція розподілу густини внутрішніх джерел енергії у матеріалі, Вт/м^3. Із урахуванням того, що глибина проникнення

електромагнітної енергії від індуктора є мала, $\Delta = 1$, будемо вважати, що вона діє на зовнішній поверхні корпусу $r = r_4$. Тоді межові умови для корпусу:

$$\frac{\partial T_k}{\partial r}|_{r=r_3} = -\frac{q_k}{h_1};\ \left[\lambda_1(T_k)\frac{\partial T_k}{\partial r} + \alpha_1(T_k(r,z,t)\right]|_{r=r_4} = L_G + N_G(T_k), \qquad (2.35)$$

$$L_G = \alpha(T_k)T_{0k} + \varepsilon_1\left(\frac{T_{0k}}{100}\right)^4;\quad N_G(T_k) = \varepsilon_1\left(\frac{T_k(r_4,z,t)}{100}\right)^4;\ \varepsilon_1 = 0,65.$$

$$\left[\frac{\partial T_k}{\partial z} - h_1 T_k\right]|_{z=0} = T_0,\ \left[\frac{\partial T_k}{\partial z} + h_1 T_k\right]|_{z=L} = 0.$$

де $h_1 = \alpha_k/\lambda_k$; α_k – коефіцієнт тепловіддачі корпусу у навколишнє середовище. Для поверхні контакту стальної труби із повітрям $\alpha_k = 9$ Вт/(м^2 $^\circ C$).

Значення теплофізичних параметрів для корпусу (сталь) за температури $T = 300K$ дорівнюють: $\rho = 7845$ кг/м^3, $c_V = 0,461$ Квт/(кг$\cdot c \cdot K$), $\lambda = 58$, $\alpha = 10,51/K$.

2.3 Математична модель зони завантаження

При формулюванні крайової задачі процесу нагрівання сухої поліетиленової суміші на ділянці завантаження необхідно враховувати наявність обертального руху шнеку зі сталою швидкістю $V_{\text{øí}}$ із нарізкою під кутом φ. Рівняння теплоперенесення набуває вигляду:

$$\rho_{\text{ï}} c_{v_{\text{ï}}}\left(V^r_{\text{øí}}\frac{\partial T}{\partial r} + V^z_{\text{øí}}\frac{\partial T}{\partial z} + \frac{\partial T_{\text{ï}}}{\partial t}\right) = \lambda_{\text{ï}}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_{\text{ï}}}{\partial r}\right) + \frac{\partial^2 T_{\text{ï}}}{\partial z^2}\right). \qquad (2.36)$$

Для поліетилену $c_{v_{\text{ï}}}$ суттєво залежить від температури нагрівання [94]. На рис. 2.3 наведено графік змінювання $c_{v_{\text{ï}}}$ залежно від температури.

Коефіцієнт тепломісткості $c_{v_{\text{ï}}}$ апроксимується залежністю

$$c_{v_i} = c_0 + c_1 T + c_2 T^2 \approx 2{,}5 - 0{,}024 \cdot T + 5{,}7 \cdot 10^{-4} \cdot T^2. \qquad (2.37)$$

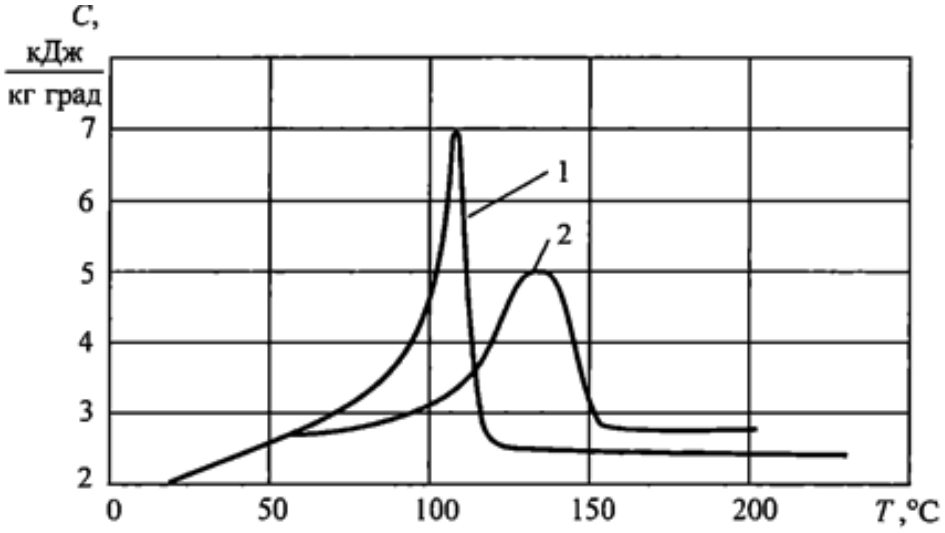


Рис. 2.3 Графік залежності C_{vp} від температури полімеру

На межі корпусу і шнеку має виконуватися умова рівності теплових потоків

$$\lambda_k \frac{\partial T_k}{\partial r}\Big|_{r=r_3} = \lambda_{sn} \frac{\partial T_{sn}}{\partial r}\Big|_{r=r_3}. \qquad (2.38)$$

$$T_{øí}\,|_{z=0} = T_{sn}^0, \qquad (2.39)$$

$$\left[\frac{\partial T_{øí}}{\partial z} + h_2 T_{øí}\right]\Big|_{z=L_1} = 0. \qquad (2.40)$$

2.4 Математична модель зони затримки плавлення

Сформулюємо задачу про рух розплаву полімеру у примежовій області шнек -- корпус шнека із урахуванням конвективної складової. Оскільки плівка розплаву є тонка, досить враховувати розподіл швидкості руху рідини вздовж осьової координати z. Тоді матимемо:

$$\rho\left(\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\frac{\partial^2 v}{\partial z^2}$$

$$c_{ef}=c+\frac{1}{\rho}L\delta(T-T^*).$$

$$[\rho c0_V+L(T-T^*)]\left(\frac{\partial T}{\partial t}+v\text{grad}T\right)=\text{div}(k\text{grad}T)+q_V.$$

Початкові умови:

$$v(z,0)=V_z(z),\ T(z,0)=T_p(z,0).$$

Межові умови:

$$v(t)|_{z=z1}=V_z(t),\ T(t)|_{z=z1}=T_p(t),\ T(r=r_3,z,t)=T_{\text{ml},z,t}.$$

2.5 Математична модель зони плавлення

Процеси у зоні плавлення полімерної суміші є визначальні з точки зору забезпечення якості ізоляційного покриття кабелів на надвисокі напруги.

Нагрів корпусу екструдера здійснюється із меншою потужністю джерела внутрішньої енергії для підтримки у зоні плавлення температури полімерної суміші (пробки), близької до температури плавлення полімеру. Тому алгоритм моделювання процесів нагріву корпусу та полімерної суміші у зоні плавлення є той самий, що і зоні завантаження.

Задача моделювання процесів плавлення полімерної суміші полягає у вирішенні задачі у двохфазній області – тверда фаза – рідинна фаза, що відома як задача типу Стефана. Тому і формулювання задачі плавлення полімеру має бути як двохфазна задача.

Із урахуванням циліндричної геометрії екструдера рівняння руху розплаву полімеру і температурного поля розплаву можна описати такою системою рівнянь:

рівняння нерозривності:

$$\frac{\partial(\rho v_z)}{\partial z}+\frac{1}{r}\frac{\partial(\rho v_\theta)}{\partial r}+\frac{1}{r}\frac{\partial(r\rho v_r)}{\partial r}=0, \tag{2.41}$$

рівняння руху:

$$\begin{aligned}&\rho\left(\frac{\partial v_r}{\partial t}+v_r\frac{\partial v_r}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_r}{\partial\theta}-\frac{v_\theta^2}{r}+v_z\frac{\partial v_r}{\partial z}\right)\\&=-\frac{\partial p}{\partial r}+\frac{1}{r}\frac{\partial}{\partial r}(r\tau_{rr})+\frac{1}{r}\frac{\partial\tau_{r\theta}}{\partial\theta}-\frac{\tau_{\theta\theta}}{r}+\frac{\partial\tau_{rz}}{\partial z}+\rho g_r;\end{aligned} \tag{2.42}$$

$$\begin{aligned}&\rho\left(\frac{\partial v_\theta}{\partial t}+v_r\frac{\partial v_\theta}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_\theta}{\partial\theta}-\frac{v_r v_\theta}{r}+v_z\frac{\partial v_\theta}{\partial z}\right)\\&=-\frac{1}{r}\frac{\partial p}{\partial\theta}+\frac{1}{r^2}\frac{\partial(r^2\tau_{r\theta})}{\partial r}+\frac{1}{r}\frac{\partial\tau_{\theta\theta)}}{\partial\theta}+\frac{\partial\tau_{\theta z}}{\partial z}+\rho g_\theta;\end{aligned} \tag{2.43}$$

$$\begin{aligned}&\rho\left(\frac{\partial v_z}{\partial t}+v_r\frac{\partial v_z}{\partial r}+\frac{v_\theta}{r}\frac{\partial v_z}{\partial\theta}+v_z\frac{\partial v_z}{\partial z}\right)\\&=-\frac{\partial p}{\partial z}+\frac{1}{r}\frac{\partial(r\tau_{rz})}{\partial r}+\frac{1}{r}\frac{\partial\tau_{\theta z}}{\partial\theta}+\frac{\partial\tau_{zz}}{\partial z}+\rho g_z;\end{aligned} \tag{2.44}$$

рівняння енергії:

$$\begin{aligned}&c_V\rho\left(\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+\frac{v_\theta}{r}\frac{\partial T}{\partial\theta}+v_z\frac{\partial T}{\partial z}\right)\\&=\lambda_T\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{1}{r^2}\frac{\partial^2 T}{\partial\theta^2}+\frac{\partial^2 T}{\partial z^2}\right)+Aq_T;\end{aligned} \tag{2.45}$$

$k=\lambda_T/(\rho c_V)$ – коефіцієнт температуропровідності.

$$q_T=-T\frac{\partial p}{\partial T}|_\rho\left[\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{1}{r}\frac{\partial v_\theta}{\partial\theta}+\frac{\partial v_z}{\partial z}\right]+\tau_{rr}\frac{\partial v_r}{\partial r}+\tau_{\theta\theta}\frac{1}{r}\left(\frac{\partial v_\theta}{\partial\theta}+v_r\right)+\tau_{zz}\frac{\partial v_z}{\partial z}+$$

$$+\tau_{r\theta}\left(r\frac{\partial(v_\theta/r)}{\partial z}+\frac{1}{r}\frac{\partial v_r}{\partial\theta}\right)+\tau_{rz}\left(\frac{\partial v_z}{\partial r}+\frac{\partial v_r}{\partial z}\right)+\tau_{\theta z}\left(\frac{1}{r}\frac{\partial v_z}{\partial\theta}+\frac{\partial v_\theta}{\partial z}\right).$$

Компоненти тензору напруження зсуву:

$$\tau_{ij} = \mu\Delta_{ij} + \left(\nu - 2/3\mu\right)(\nabla\cdot v)\delta_{ij};\ \Delta_{ij} = \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}.$$

У циліндричній системі координат маємо:

$$\tau_{rr} = \mu_e\left[2\frac{\partial v_r}{\partial r} - \frac{2}{3}(\nabla\cdot v)\right] + \nu(\nabla\cdot v); \tag{2.46}$$

$$\tau_{\theta\theta} = \mu_e\left[2\frac{1}{r}\left(\frac{\partial v_\theta}{\partial\theta} + v_r\right) - \frac{2}{3}(\nabla\cdot v)\right] + \nu(\nabla\cdot v); \tag{2.47}$$

$$\tau_{zz} = \mu_e\left[2\frac{\partial v_z}{\partial z} - \frac{2}{3}(\nabla\cdot v)\right] + \nu(\nabla\cdot v); \tag{2.48}$$

$$\tau_{rz} = \mu_e\left(\frac{\partial v_z}{\partial r} + \frac{\partial v_r}{\partial z}\right);\ \tau_{r\theta} = \mu_e\left(r\frac{\partial(v_\theta/r)}{\partial r} + \frac{1}{r}\frac{\partial v_r}{\partial\theta}\right);\ \tau_{z\theta} = \mu_e\left(\frac{\partial v_\theta}{\partial z} + \frac{1}{r}\frac{\partial v_z}{\partial\theta}\right), \tag{2.49}$$

μ_e – ефективна в'язкість при зсувній течії, ν – кінематична в'язкість.

$$\mu_e = \mu_0(I_2/2)^{\frac{n-1}{2}}e^{-\beta(T-T_0)},\quad n=2\rightarrow\ \mu_e = \mu_0\sqrt{I_2/2}e^{-\beta(T-T_0)}, \tag{2.50}$$

$$(\nabla\cdot v) = \frac{1}{r}\frac{\partial(rv_r)}{\partial r} + \frac{1}{r}\frac{\partial v_\theta}{\partial\theta} + \frac{\partial v_z}{\partial z}.$$

де μ_0 – початкова в'язкість; I_2 – другий інваріант тензору швидкостей деформації; n – показник аномалії в'язкості; v_r, v_θ, v_z – компоненти вектору швидкості; τ_{ij} $(i,j = r,\theta,z)$ – компоненти тензору напружень; T – температура, β – температурний коефіцієнт; T_0 – початкова температура.

Після заміщення виразів для компонент тензору напружень зсуву у рівняння (2.22)–(2.25) отримуємо таку систему рівнянь для компонент швидкості руху і температури:

$$\frac{\partial v_r}{\partial t} + v_r\frac{\partial v_r}{\partial r} + v_z\frac{\partial v_r}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial r} + \mu_e\left[\Delta v_r + \frac{\partial^2 v_z}{\partial r\partial z}\right] + g_r; \tag{2.51}$$

$$\frac{\partial v_z}{\partial t} + v_r\frac{\partial v_z}{\partial r} + v_z\frac{\partial v_z}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \mu_e\left[\Delta v_z + \frac{1}{r}\frac{\partial^2 v_r}{\partial r\partial z} + \frac{1}{r}\frac{\partial}{\partial z}\left(\frac{\partial v_r}{\partial r}\right)\right] + g_z; \tag{2.52}$$

рівняння енергії:

$$\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+v_z\frac{\partial T}{\partial z}=k\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial z^2}\right)+\frac{A}{\rho c_v}q_T; \tag{2.53}$$

$$q_T=-T\frac{\partial p}{\partial T}|_\rho\left[\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{\partial v_z}{\partial z}\right]+2\mu_e[\left(\frac{\partial v_r}{\partial r}\right)^2$$

$$+\left(\frac{\partial v_z}{\partial z}\right)^2+\left(\frac{\partial v_z}{\partial r}+\frac{\partial v_r}{\partial z}\right)^2+\left(\frac{\partial v_\theta}{\partial z}\right)^2]. \tag{2.54}$$

A – термічний еквівалент роботи; c_v – питома тепломісткість рідини за сталого об'єму. Вираз $(\partial p/\partial T)|_\rho$ визначається із рівняння стану. У разі $p=\rho RT$ він дорівнює $R\rho$.

2.6 Математична модель зони дозування

Перехід полімерів із рідинного стану у твердий супроводжується складними хіміко- фізичними процесами – багатостадійними реакціями полімеризації мономеру, переходом аморфної фази полімеру у кристалічну. Аналіз цього переходу починають із процесу полімеризації, який визначає просторово-часовий розподіл температури для процесу кристалізації. Нестаціонарні, неізотермічні і просторово розподілені явища полімеризації і кристалізації визначають напружений і деформований стан полімеру.

Запроваджуються [162] два макрокінетичні параметри α і β, які визначають питомий внесок відповідно полімерної і кристалічної фаз. Ступінь полімеризації $\alpha(x,t)$ визначає ступінь завершеності процесу полімеризації і може приймати значення від 0 (вміст полімеру – 0) до 1 (весь мономер перейшов у полімер). Ступінь кристалізації $\beta(x,t)$ визначає ступінь завершеності процесу кристалізації і може приймати значення від 0 (вміст

кристалічної фази – 0) до 1 (уся аморфна фаза полімеру перейшла у кристалічну).

За припущення, що технологічні напруження не впливають на температуру і протікання процесу кристалізації, треба вирішувати такі задачі:

1) задачу визначення температурних полів і ступеню кристалізації, або теплокінетичну задачу;

2) крайову задачу визначення напружено-деформованого стану (НДС) системи, що твердіє.

Задача про визначення температурних полів і ступеню полімеризації та кристалізації описується такою крайовою задачею:

$$c(T)\rho(T)\frac{\partial T(x,t)}{\partial t} = \mathrm{div}(\lambda(T)\mathrm{grad}T(x,t)) + \rho(T)\left(Q_\alpha \frac{d\alpha}{dt} + Q_\beta \alpha \frac{d\beta}{dt}\right);\ x \in V; \quad (2.55)$$

кінетичне рівняння полімеризації

$$\frac{d\alpha(x,t)}{dt} = K_\alpha(1-\alpha(x,t))(1+c_0\alpha(x,t))(1+n\alpha(x,t)) \quad (2.56)$$

кінетичне рівняння кристалізації

$$\frac{d\beta(x,t)}{dt} = K_\beta(T)(1+A_0\beta(x,t))(\beta_p(T)-\beta(x,t)),\ x \in V. \quad (2.57)$$

$$K_\alpha = k_\alpha \exp\left(-\frac{U}{RT}\right);\ K_\beta = k_\beta \exp\left(-\frac{E}{RT} - \frac{\Psi}{T_p - T}\right).$$

Початкові та межові умови:

$$\beta(x,0) = 0,\ T(x,0) = T_0; \quad (2.58)$$

$$\lambda(T)n\mathrm{grad}T(x,t) = h(T(x,t) - T_{av});\ n \cdot \mathrm{grad}T(x,t) = 0,\ x \in S_2; \quad (2.59)$$

$$T(x,t) = T^*(x,t);\ x \in S_1. \quad (2.60)$$

де c – питома теплоємність; ρ – густина; λ – коефіцієнт теплопровідності; Q_α, Q_β – інтенсивності теплових джерел, що обумовлені полімеризацією і кристалізацією відповідно; R – універсальна газова стала; $U, E, k_\alpha, k_\beta, c_0, c_1, n$ –

кінетичні сталі, що визначаються експериментально із калориметричних вимірювань; T_p – температура плавлення; β_p – рівноважний ступінь кристалізації.

Теплофізичні властивості полімеру низької густини (ПЕНГ):

c – 2,0–3,5 кДж/кг К; λ – 0,29–0,42 Вт/м К; T_{pl} – 376–388 К; T_c – 138 К; Теплота плавлення – 7 кДж/моль.

Результат розв'язання системи рівнянь (2.55)–(2.60) є визначення температурного поля і ступеню кристалізації, із урахуванням яких вирішується задача напружено- деформованого стану (НДС) виробу.

Конкретизуємо рівняння (2.35) у циліндричній системі координат:

$$c(T)\rho(T)\frac{\partial T}{\partial t}=\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(T)\frac{\partial T}{\partial r}\right)+\frac{\partial}{\partial z}\left(\lambda(T)\frac{\partial T}{\partial z}\right)+\rho(T)\left(Q_\alpha\frac{d\alpha}{dt}+Q_\beta\alpha\frac{d\beta(t)}{dt}\right). \quad (2.61)$$

У рівняннях позначено: λ_0 – коефіцієнт теплопровідності; Q_p – тепловий ефект полімеризації; Q_{cr} – тепловий ефект кристалізації (швидкість питомого тепловиділення при кристалізації ПЕНГ); k_α, k_β – сталі швидкостей полімеризації і кристалізації; U – енергія активації процесу полімеризації; E^a – енергія активації процесу кристалізації; $\varepsilon_1, \varepsilon_2$ – критерії автокаталітичності процесів полімеризації і кристалізації; Ψ – характерна температура полімеру; R – універсальна газова стала; h_0, h_1 – коефіцієнти теплообміну із навколишнім середовищем; $T_p = 415$ К – рівноважна температура плавлення; β_b – рівноважний ступінь кристалічності:

$$\beta_b = 0,52\sqrt{1-(T/T_p)^4}.$$

2.7 Висновки по розділу 2

Як свідчить огляд численних праць [5,6,11,114,121,126, 129,143,183,

204,223,224,240], процеси нагрівання полімерної суміші в екструдерах у зоні завантаження полімерної суміші, плавлення полімерів у зоні плавлення, гомогенізації та кристалізації у зоні дозування та у головці, що формує, описуються квазілінійними рівняннями тепло- масоперенесення, що належать до систем рівнянь Нав'є—Стокса. Такий підхід до побудови математичних моделей у різних зонах екструдера не відповідає суті справи, оскільки:

а) процес індукційного нагрівання корпусу шнека супроводжується втратами теплової енергії внаслідок її випромінювання на межі індуктор--зовнішня поверхня корпусу;

б) процес нагрівання полімерної суміші у зоні завантаження суттєво залежить від теплофізичних параметрів суміші і має описуватися нелінійним рівнянням теплопровідності;

в) процеси у зоні затримки плавлення мають описуватися не рівнянням теплопровідності, а системою рівнянь Нав'є—Стокса із урахуванням фазового переходу тверда суміш—плівка розплаву біля внутрішньої поверхні корпусу шнека;

д) у зоні плавлення полімерної суміші необхідно розглядати систему рівнянь теплопровідності (нагрів суміші) та систему рівнянь Нав'є—Стокса із урахуванням наявної межі фазового переходу тверда суміш—розплав полімеру, що, на наш погляд, відіграє суттєву роль у формуванні вихідного продукту.

е) процеси кристалізації розплаву полімеру на визначеній ділянці (зона дозування) необхідно розглядати як двохфазний процес розплав – гомогенізація – кристалізація полімеру.

Розділ 3 ЧИСЛОВО-АНАЛІТИЧНИЙ МЕТОД РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ КРАЙОВИХ ЗАДАЧ МАТЕМАТИЧНОЇ ФІЗИКИ

3.1 Вступ

Більшість реальних процесів і явищ за своєю сутністю є нелінійні. Прагнення підвищити достовірність і якість дослідження реальних фізичних процесів і явищ диктує необхідність розробки методів аналізу цих процесів, вільних від недоліків застосування чисельних схем й які б забезпечили якість цього аналізу з метою розробки систем управління процесами або прогнозування поведінки об'єктів, що досліджуються.

Під терміном ``числово-аналітичний метод" мається на увазі розробка методу наближеного (або точного у разі лінійної системи) аналітичного розв'язання задачі із подальшою розробкою відповідних алгоритмів реалізації засобами комп'ютерної техніки.

Перша робота автора, що присвячена розв'язанню нелінійних крайових задач математичної фізики опублікована у 1977 р. [100]. Розв'язання нелінійних крайових задач математичної фізики для частинних випадків (електродугове зварювання, індукційний наргів металевих заготовок, моделювання екологічних систем, конвективно-дифузійні процеси у циклонних пристроях тощо) викладено у працях [13--16, 49, 50, 56, 58--61, 96].

В останні роки з'явилися праці, в яких розглядаються моделювання нелінійних крайових задач математичної фізики, що грунтуються на ітераційному методі. Як типовий приклад, робота [41], в якій розглядається апроксимація розв'язку нелінійного рівняння із нелінійними межовими умовами за власними функціями лінійного оператора.

3.2 Огляд методів розв'язання нелінійних рівнянь

3.2.1 Декомпозиція рівнянь

У цьому розділі ми не розглядаємо підходи до розв'язання нелінійних крайових задач для рівнянь із частинними похідними, що грунтуються на використанні різницевих схем різного порядку, оскільки цим підходам присвячено безліч науково-практичних праць. Загальна риса цих праць -- відсутність практичних результатів для тривимірних і навіть двомиірних відносно просторових змінних рівнянь, що скоріш за все, пов'язано із трудощами розв'язання відповідних систем нелінійних алгебраїчних рівянь як наслідок застосування різницевих схем за цими змінними.

Для прикладу, можна послатися на результати моделювання задачі про конвективне перенесення субстанції у двовимірному випадку, які наведено у [66] згідно із різницевою схемою [112]. В [27] розглядається пошук розв'язку тривимірного рівняння теплопровідності шляхом декомпозиції цього рівняння на три одновимірні, а саме, рівняння теплопровідності

$$\frac{\partial^2 v}{\partial x_1^2}+\frac{\partial^2 v}{\partial x_2^2}+\frac{\partial^2 v}{\partial x_2^2}=\frac{1}{\chi}\frac{\partial v}{\partial t},\ t>0 \tag{3.1}$$

у прямокутному паралелепіпеді

$$a_k < x_k < b_k,\ k=1,2,3,$$

де $v(x_1,x_2,x_3,t)$ – шукана функція температури, χ – коефіцієнт температуропровідності.

Для низки типів початкових і межових умов його розв'язок можна подати у вигляді суперпозиції для трьох задач із однією змінною.

Нехай функції $v_k(x_k,t)$ є розв'язки одновимірних задач

$$\frac{\partial^2 v_k}{\partial x_k^2}=\frac{1}{\chi}\frac{\partial v_k}{\partial t} \tag{3.2}$$

із межовими умовами

$$\left[\alpha_k \frac{\partial v_k}{\partial x_k} - \beta_k v_k\right]_{x_k=a_k} = 0;\ \left[\alpha_k \frac{\partial v_k}{\partial x_k} - \beta_k v_k\right]_{x_k=b_k} = 0 \qquad (3.3)$$

Нехай початкові умови:

$$v_(x_k, t) = V_k(x_k)\,|_{t=0}\ .$$

Тоді розв'язком рівняння (3.3) із початковою умовою

$$v\,|_{t=0} = V_1(x_1)V_2(x_2)V_3(x_3)$$

і межовими умовами

$$\left[\alpha_k \frac{\partial v}{\partial x_k} - \beta_k v\right]_{x_k=a_k} = 0;\ \left[\alpha_k \frac{\partial v}{\partial x_k} - \beta_k v\right]_{x_k=b_k} = 0$$

буде

$$v = v_1(x_1,t)v_2(x_2,t)v_3(x_3,t).$$

Заміщення цього виразу у (3.1) дає

$$v_2v_3\frac{\partial^2 v_1}{\partial x_1^2} + v_1v_3\frac{\partial^2 v_2}{\partial x_2^2} + v_2v_1\frac{\partial^2 v_3}{\partial x_3^2} = \frac{1}{\chi}\left(v_2v_3\frac{\partial v_1}{\partial t} + v_1v_3\frac{\partial v_2}{\partial t} + v_2v_1\frac{\partial v_3}{\partial t}\right).$$

Отже, за визначених умов розв'язок задачі (3.1)—(3.3) можна отримати шляхом розв'язання трьох одновимірних крайових задач.

3.2.2 Метод малого параметра

Для стаціонарних задач в [156] запропонований метод розв'язання нелінійних стаціонарних рівнянь теплопровідності, який грунтується на асимптотичному підході (метод малого параметра). Цей підхід застосовано в [149].

А саме, якщо $\lambda = \lambda_0(1+\varepsilon T)$, ε – малий параметр, рівняння теплопровідності записується у вигляді

$$\varepsilon(\nabla T)^2 + (1+\varepsilon T)\Delta T + F_1 = 0. \qquad (3.4)$$

Межові умови розглядаються у вигляді

$$\frac{\partial T}{\partial z}|_{z=0} = 0,\ T|_{z=h} = 0; \qquad (3.5)$$

$$\left[\lambda(T)\frac{\partial T}{\partial n} + \alpha T\right]|_S = 0. \qquad (3.6)$$

S – циліндрична поверхня, твірні якої паралельні осі Oz; α – коефіцієнт тепловіддачі.

Розв'язання цього рівняння із відповідними межовими умовами відшукується у вигляді

$$T = T_1 + \varepsilon T_2, \qquad (3.7)$$

Нелінійна задача (3.4)–(3.6) зводиться до розв'язання двох лінійних задач теплопровідності:

$$\delta T_1 + F_1 = 0,$$

$$\frac{\partial T_1}{\partial z}|_{z=0} = 0,\ T_1|_{z=h} = 0;$$

$$\left[\frac{\partial T_1}{\partial n} + \alpha T_1\right]|_S = 0.$$

$$\delta T_2 + F_2 = 0,$$

$$\frac{\partial T_2}{\partial z}|_{z=0} = 0,\ T_2|_{z=h} = 0;$$

$$\left[\frac{\partial T_2}{\partial n} + \alpha T_2\right]|_S = f_2.$$

$$F_2 = (\nabla T_1)^2 + (1+\varepsilon T_1)\Delta T_1. \qquad (3.8)$$

Структури розв'язань крайових цих задач подаються у вигляді

$$T_1(x,y,z) = \sum_m \overline{T}_1(x,y,\gamma_m)K(z,\gamma_m),$$

$$T_2(x,y,z)=\sum_m \overline{T}_2(x,y,\gamma_m)K(z,\gamma_m),$$

$$K(z,\gamma_m)=\cos[(2m+1)\pi/2h)z];\quad \overline{T}_k(x,y,\gamma_m)=\int_0^h K(z,\gamma_m)T_k(x,y,z)dz,\ k=1,2.$$

$$\overline{T}_1(x,y,\gamma_m)=\sum_{i,j} C_{ij}(\gamma_m)X_{ij}(x,y),$$

$$\overline{T}_2(x,y,\gamma_m)=\overline{\Phi}_0(x,y,\gamma_m)+\sum_{i,j}(\gamma_m)\eta_{ij}(x,y).$$

$$X_{ij}=\varphi_{ij}-\omega D_1\varphi_{ij}+\omega\varphi_{ij}h_0;\quad \eta_{ij}=\psi_{ij}-\omega D_1\psi_{ij}+\omega\psi_{ij}h_0,\ h_0=\alpha/\lambda_0.$$

φ_{ij}, ψ_{ij} – повні системи функцій (поліноми Чебишова, Лежандра тощо).

Стосовно пропонованого підходу до розв'язання рівняння (3.4) варто зазначити, що: 1) врахування нелінійного члена у вигляді виразу (3.6) як розв'язання лінійної задачі, що відшукується розкладанням за однією системою базисних функцій φ_{ij} при розв'язанні лінійного рівняння відносно T_2 за іншою системою функцій ψ_{ij} зовсім не обгрунтовано; 2) у правій частині F_2 рівняння відносно T_2 присутній вираз для T_1, що визначається через систему функцій φ_{ij}; 3) вочевидь, вибір цих базисних функцій ніяк не враховує специфіку межових умов.

Розглянутий підхід до розв'язання нелінійних рівнянь із частинними похідними, на мій погляд, може бути ефективним стосовно задач теплопровідності для процесів із порівняно незначними градієнтами температур для процесів, що досліджуються. Мова йде про можливі значення величин $\lambda_0\varepsilon T$. У разі необхідності врахування квадратичного члена ε^2T^2 такий підхід призводить до необхідності побудови і розв'язання третього рівняння відносно T_3 тощо.

Вочевидь, такий підхід є непридатний для нестаціонарних рівнянь теплопровідності, оскільки коефіцієнт тепломісткості c_v має зовсім іншу

залежність від T.

3.2.3 Числово-аналітичний метод у теорії періодичних розв'язань рівнянь із частинними похідними, [141]

Хоча у роботі питання пошуку періодичних розв'язань рівнянь із частинними похідними не розглядається, але з огляду на те, що стосовно нелінійних рівнянь із частинними похідними це – можливо, єдиний конструктивний підхід до побудови наближених числово-аналітичних розв'язань нелінійних рівнянь із частинними похідними, ми розглянемо підхід, що запропонований академіком А.М. Самойленко до пошуку періодичних розв'язань рівнянь із частинними похідними як лінійних, так і нелінійних.

Спочатку розглянемо питання пошуку періодичних розв'язань лінійних рівнянь із частинними похідними із запізнюванням.

В евклідовому просторі E_n розглядаються системи рівнянь вигляду

$$\frac{\partial^2 u(t,x)}{\partial t \partial x} = B_1(t,x)u(t,x) + B_2(t,x)u(t-\tau,x) + f(t,x) = F(t,x,u,u_x). \qquad (3.9)$$

Тут $u, f \in E_n, B_1, B_2$ – $n \times n$-матриці, x – вектор незалежних змінних. Область визначення функції F:

$$\Omega_0 = \{(t,x,u,u_\tau) \,|\, t \in R;\ x \in [-a,a];\ (u,u_\tau) \in [b,c] \times [b,c]\}.$$

Початкові і межові умови задаються у вигляді

$$u(0,x) = u_0(0) + v(x), \quad u(t,0) = u_0(t) + v(0), \qquad (3.10)$$

де $u_0(t)$ – задана обмежена періодична із періодом T функція із обмеженою похідною:

$$u_0(t+T) = u_0(t),\ |u_0(t)| \le N_0,\ |u_0'(t)| \le N. \qquad (3.11)$$

Припустимо, що функція $u_0(t)$ задана, неперервна, періодична із періодом T. а функція $v(x)$ вибирається так, щоб забезпечити періодичність по t

розв'язку $u(x,t)$ системи (3.9).

Для вирішення поставленої задачі розглядається послідовність періодичних по t із періодом T функцій (позначення авторські).

$$u_0(t,x) = u_0(t) + v_0(x). \tag{3.12}$$

$$u_1(t,x) = u_0(t) + v_0(x) + \delta_1(x) + \int_0^x \int_0^t [F(\xi,\eta,u_0,u_{0\tau}) - \overline{F(s,\eta,u_0,u_{0\tau})}]d\xi d\eta;$$

…

$$u_{n+1}(t,x) = u_0(t) + v_0(x) + \sum_{i=1}^{n+1}\delta_i(x) + \int_0^x \int_0^t [F(\xi,\eta,u_n(\xi,\eta,u_n(\xi-\tau,\eta)) - \overline{F(s,\eta,u_n(s,\eta),u_n(s-\tau,\eta))}]d\xi d\eta.$$

(3.13)

Функції $v_0(x)$ і $\delta_i(x)$ визначаються із умов

$$\overline{F_i} = 0,\ i = 0,1,2,\ldots \tag{3.14}$$

$$F_i = F(t,x,u_i(t,x),u_i(t-\tau,x)).$$

$$\overline{F(t,x,u,v)} = \frac{1}{T}\int_0^T F(t,x,u,v)dt.$$

Функція $F(t,x,u,u_\tau)$ має задовольняти такі умови:

$$1.\ |F(t,x,u,u_\tau)| \le M; \tag{3.15}$$

$$|F(t,x,u,u_\tau) - F(t,x,\overline{u},\overline{u}_\tau) \le K_1|u-\overline{u}| + K_2|u_\tau - \overline{u}_\tau|; \tag{3.16}$$

2. Матриця

$$P_1(x) = \frac{1}{T}\int_0^T [B_1(t,x) + B_2(t,x)]dt \tag{3.17}$$

не вироджена при $x \in [-a,a]$, а матриця P^{-1} визначена співвідношенням

$$(P^{-1}) = \max_{|x|\le a} |\{P_1^{-1}(x)\}|. \tag{3.18}$$

3. Власні значення матриці

$$R = (K_1 + K_2)P^{-1}A^* + A, \tag{3.19}$$

де

$$A^* = a\frac{T}{3}(K_1 + K_2); \quad A = a\frac{T}{3}K_1 + a\left(\frac{T}{3} + \frac{3\alpha^2(\tau)}{8T}\right)K_2,$$

належать кругу одиничного радіусу.

4. Вектори b,c і M і період T задовольняють нерівність

$$c - b \geq SM, \tag{3.20}$$

де

$$S = 2P^{-1}A^*(E - R)^{-1} + aTE.$$

Теорема. *Нехай в області* Ω_0 *функція* $F(t,x,u,u_\tau)$ *задовольняє умови 1.--4. Тоді послідовність* (3.13) *періодичних по* t *із періодом* T *функцій* $u_n(t,x)$ *розв'язків системи* (3.9)—(3.12) *збігається рівномірно відносно* $(t,x) \in (-\infty,\infty) \times \{x : |x| \leq a\}$ *до деякої функції* $u^*(t,x)$. *Ця функція є єдиний періодичний по* t *із періодом* T *розв'язок системи* (3.9)—(3.12), *що задовольняє й інтегральне рівняння*

$$u(t,x) = u_0(t) + v(x) + \int_0^x \int_0^t F(\xi,\eta,u(\xi,\eta),u(\xi-\tau,\eta))d\xi d\eta, \tag{3.21}$$

$$v(x) = \lim_{n\to\infty}[v_0(x) + \sum_{i=1}^{n}\delta_i(x)]. \tag{3.22}$$

Далі цей результат розповсюджується на квазілінійні системи із аргументом, що відхиляється, нейтрального типу та нелінійні системи із запізнюванням.

Розглядається система нелінійних рівнянь нейтрального типу

$$\frac{\partial^2 u(t,x)}{\partial t \partial x} = f[t,x,u(t,x),u(t-\tau,x),u_t'(t,x),u_t'(t-\tau,x),u_x(t,x),u_x'(t,x),u_{tx}''(t-\tau,x)].$$

(2.23)

Як і раніше, u, f – вектори евклідова простору E_n, запізнювання τ – стале ($0 < \tau < T$). Початкові і межові умови вибираються у вигляді

$$u(0,x) = u_0(0) + v(x); \quad u(t,0)u_0(t) + v(0). \tag{3.24}$$

Тут $u_0(t)$ – задана обмежена періодична функція із обмеженою похідною:

$$u(t+T) = u_0(t), \quad |u_0(t)| \le N_0, \; |u_0'(t)| \le N. \tag{3.25}$$

Умови 1. –4. у попередній теоремі замінюються на такі:

2а. Матриця

$$C(x) = \frac{1}{T}\int_0^T \left[\frac{\partial f}{\partial y_2} + \frac{\partial f}{\partial z_2}\right] dt,$$

елементи якої є неперервні функції своїх аргументів в області Ω, є невироджена в Ω, тобто $detC(x) \neq 0$. Існує стала матриця P^{-1} із невід'ємними елементами така, що

$$(P^{-1})_{ij} = \sup_{\Omega} \left| \left\{ \left[\frac{1}{T}\int_0^T \left[\frac{\partial f}{\partial y_2} + \frac{\partial f}{\partial z_2}\right] dt \right]^{-1} \right\}_{ij} \right|. \tag{3.26}$$

3а. Власні значення матриці

$$B = aP^{-1}(K_1 + K_2)$$

і матриці R,

$$R = -[B^* + (K - 1 + K_2)P^{-1}(B^* + E) + E]A,$$

де

$$A = a\frac{T}{2}(K_1 + K_2) + a(K_3 + K_4) + \frac{T}{2}(K_5 + K_6) + K_7,$$

$$B^* = a(K_1 + K_2)(E - B)^{-1}(E + P^{-1}),$$

містяться у крузі одиничного радіусу.

4а. Вектори b, c, M задовольняють нерівності

$$b + \frac{1}{2}SM \le u_0(t) - |c_0| - P^{-1}aM_0^0 < u_0(t) + |c_0| + P^{-1}aM_0^0 \le c - \frac{1}{2}SM,$$

$$S = 2a(E - B)^{-1}(E + P^{-1})A(E - R)^{-1} + aTE.$$

Будується послідовність періодичних наближень:

$$u_0(t,x) = u_0(t) + v_0(x). \tag{3.27}$$

$$\begin{aligned} u_1(t,x) &= u_0(t) + v_0(x) + \delta_1(x) + \int_0^x \int_0^t [f_0 - \overline{f}_0]dtdx \\ &\cdots \\ u_{n+1}(t,x) &= u_0(t) + v_0(x) + \sum_{i=1}^{n+1} \delta_i(x) + \int_0^x \int_0^t [f_n - \overline{f}_n]dtdx. \end{aligned} \tag{3.28}$$

$$n = 1,2,\ldots$$

$$f_n = f[t,x,u_n(t,x),u_n(t-\tau,x),u'_{nt}(t,x),u'_{nt}(t-\tau,x),u_{nx}(t,x),u'_{nx}(t,x),u''_{ntx}(t-\tau,x)];$$

$$\overline{f}_n = \frac{1}{T}\int_0^T f_n dt.$$

Функції $v_0(x)$, $\delta_i(x)$ $(i = 1,2,\ldots)$ вибираються так, щоб

$$\overline{f}_i = 0, \ i = 0,1,2,\ldots \tag{3.29}$$

Теорема. *Нехай функція* $f(t,x,y,z,y_1,z_1,y_2,z_2,z_3)$ *визначена в області* Ω *і задовольняє умови 1а--4а. Тоді послідовність* (3.27)—(3.29) *періодичних по* t *із періодом* T *наближень* $u_n(t,x)$, *у яких* $v_0(x)$ *і* $\delta_i(x)$ *визначені співвідношеннями* (3.28), *а також послідовності* $\{u'_{nt}(t,x)\}$, $\{u'_{nx}(t,x)\}$, $\{u''_{ntx}(t,x)\}$ *збігаються за* $t \to \infty$ *рівномірно відносно* $(t,x) \in (-\infty,\infty)\times[-a,a]$ *до функції* $u^*(t,x)$ *і відповідних її частинних похідних. При цьому послідовність*

$$\{v_n(x)\} = \{v_0(x) + \sum_{i=1}^{n} \delta_i(x)\}$$

за $t \to \infty$ рівномірно відносно $x \in [-a,a]$ збігається до функції $v(x)$.

Гранична функція $u^*(t,x)$ є єдиний періодичний по t із періодом T розв'язок системи (3.23)—(3.26), *що задовольняє і систему інтегро-диференціальних рівнянь*

$$u(t,x) = u_0(t) + v(x) +$$

$$+\int_0^x\int_0^t f[t,x,u(t,x),u(t-\tau,x)u'_t(t,x),u'_t(t-\tau,x),u'_x(t,x),u'_x(t-\tau,x)u''_{tx}(t-\tau,x)]dtdx.$$

Пропонований метод побудови ітераційного процесу відшукання періодичного розв'язку системи нелінійних рівнянь стосується системи одновимірних рівнянь гіперболічного типу и не має відношення до крайових задач для рівнянь із частинними похідними. Він забезпечує збіжність ітераційного процесу за досить жорстких умов (зокрема, власні значення відповідних матриць похідних за просторовими змінними мають міститися у крузі одиничного радіуса, на функцію f накладаються умови неперервності за всіма аргументами тощо). Не обґрунтовано пошук як початкового розв'язку $u_0(t,x)$ так і наближень $u_{n+1}(t,x)$ у вигляді адитивних функцій відносно часу і просторових змінних. Крім того, визначення на кожній ітерації функцій $\delta_i(x)$ за умови $\overline{f_i}=0$ хоча і виглядає привабливим, але не надає можливості алгоритмізувати процес пошуку розв'язання системи нелінійних рівнянь.

Теорема про збіжність процесу, що наведена у роботі, містить кілька невизначених посилань, що унеможливлює її застосування при розв'язанні задачі, що засвідчив наведений у роботі приклад вирішення цієї задачі.

З іншого боку, це -- майже єдина спроба сформулювати умови збіжності ітераційного процесу для пошуку періодичних розв'язків систем рівнянь із частинними похідними гіперболічного типу.

3.3 Метод розв'язання нелінійних крайових задач

Розглянемо систему нелінійних рівнянь

$$a_0\frac{\partial^2 u(t,x)}{\partial t^2}+a_1\frac{\partial u(t,x)}{\partial t}=F(t,x,u(t,x),\ldots)=f(t,x,u(t,x),\ldots)+L(x)u(t,x), \qquad (3.30)$$

$$D=\{0\le x\le X,\ 0\le t\le T\}$$

із початковими умовами

$$u(x,0) = u_0(x), \quad \frac{\partial u(x,t)}{\partial t}|_{t=0} = u_1(x) \tag{3.31}$$

та межовими умовами

$$g(t,x)|_{x=S} = 0. \tag{3.32}$$

У рівнянні (3.30) $a_0 = 0$ у разі рівнянь параболічного типу, $a_0 = 0$, $a_1 = 0$ у разі рівнянь еліптичного типу. Передбачається, що функція $F(t,x,u(t,x),\ldots)$ у правій частині рівняння може бути подана у вигляді лінійного оператора відносно других похідних по просторовим змінним L і нелінійного оператора f. Як відомо, більшість об'єктів із розподіленими параметрами описується саме такими рівняннями.

У разі необхідності нелінійну частину $f(t,x,u(t,x),\ldots)$ можна лінеаризувати шляхом розкладання її у ряд Тейлора в околі лінійного розв'язку $u^{(0)}(x,t)$, обмежуючись лінійними складовими:

$$f(t,x,u(t,x),\ldots) \approx f(t,x,u_0(t,x),u_1(t,x),\ldots) + \frac{\partial f}{\partial x}|_{u_0,u_1}(u(t,x) - u_0) + \ldots$$

Розглядаються функції $u(t,x) \in L_G^2$, тобто функції, що є інтегровні разом із квадратами цих функцій. Запроваджується скалярний добуток у L_G^2

$$(u,v) = \int_G u(x)v(x)dG.$$

Запишемо рівняння (3.30) у такому вигляді:

$$a_0 \frac{\partial^2 u(t,x)}{\partial t^2} + a_1 \frac{\partial u(t,x)}{\partial t} = L(x)u(t,x) + N[t,x,u(t,x)]. \tag{3.33}$$

У роботі пропонується наближений ітераційний числово-аналітичний метод розв'язання нелінійної крайової задачі (3.30) – (3.32), який грунтується на такому.

3.3.1 Розв'язання лінійної задачі

Позначимо $u^{(0)}(t,x)$ с – розв'язок лінійної частини рівняння (3.33):

$$a_0 \frac{\partial^2 u(t,x)}{\partial t^2} + a_1 \frac{\partial u(t,x)}{\partial t} = L(x)u^{(0)}(t,x) \tag{3.34}$$

із початковими умовами (3.31) та лінійними межовими умовами, які можна подати у вигляді

$$\left[\alpha_1 \frac{\partial u^{(0)}(t,x)}{\partial x} + \beta_1 u^{(0)} \right]|_{x=0} = \gamma_1(t); \tag{3.35}$$

$$\left[\alpha_2 \frac{\partial u^{(0)}(t,x)}{\partial x} + \beta_2 u^{(0)} \right]|_{x=X} = \gamma_2(t). \tag{3.36}$$

Коефіцієнти α_i, β_i, $i = 1,2$ -- відомі сталі, що вибираються із міркувань фізичного сенсу задачі, що вирішується. Межові умови (3.35), (3.36) отримують (у разі нелінійної функції g) шляхом лінеаризації її в околі лінійної частини розв'язку крайової задачі. Нелінійна частина функції g у цьому разі враховується у функціях $\gamma_1(t), \gamma_2(t)$ на межі області S.

Після відшукання розв'язку лінійної частини крайової задачі як розв'язання задачі (3.34), (3.31), (3.35), (3.36) – функції $u^{(0)}(t,x)$ – отриманий розв'язок підставляється у нелінійну частину $N_1[t,x,u^{(0)}(t,x)]$ а також у функції $\gamma_1(t,u^{(0)}(t,S))$ і $\gamma_2(t,u^{(0)}(t,S))$.

Оскільки функцію $u^{(0)}(t,x)$ отримано як розв'язок відповідної крайової задачі у аналітичному вигляді, із відповідними виразами можна виконувати операції інтегрування у просторі L_G^2.

Для розв'язання лінійної крайової задачі (3.34), (3.31), (3.35), (3.36) відносно функції $u^{(0)}$ застосуємо метод скінченних інтегральних перетворень у області $\bar{D} = D + S$.

Загальна схема методу скінченних інтегральних перетворень, як відомо, побудована на принципі суперпозиції і полягає у пошуку розв'язків задач Штурма--Ліувілля відносно самосполучених операторів $L(x)v$ за відповідними незалежними змінними. Для простоти розглянемо скалярне рівняння вигляду (3.34):

$$\frac{\partial^l v(t,x)}{\partial t^l} = \frac{\partial^2 v(t,x,y,z)}{\partial x^2} + \frac{\partial^2 v(t,x,y,z)}{\partial y^2} + \frac{\partial^2 v(t,x,y,z)}{\partial z^2} + h(t,x,y,z). \quad (3.37)$$

Функція $h(t,x,y,z)$ – відома функція збурення.

Розв'язок лінійної задачі згідно з принципом суперпозиції подамо у вигляді

$$v(t,x,y,z) = T(t)X(x)Y(y)Z(z).$$

Задача на власні вектори і власні значення відносно змінної x має вигляд:

$$L_x u(t,x,y,z) = \left[\frac{\partial}{\partial x}\left(p(x)\frac{\partial}{\partial x}\right) + q(q)\right] X(x) = -\lambda^2 \overline{X}(x); \quad (3.38)$$

$$\overline{X}(x) = \int_{x_1}^{x_2} X(x)\varphi_x(\lambda_n, x)r(x)dx, \quad (3.39)$$

де ядро $\varphi(\lambda_n, x)$ перетворення за змінною x визначається виразом

$$\varphi_x(\lambda_n, x) = \frac{1}{\mathrm{P}\varphi_x(\lambda_n)\mathrm{P}^2_{L^2_{x_1,x_2}}}\left\{\left[\alpha_1\frac{d\psi_x(\lambda,x)}{dx} + \beta_1\psi_x(\lambda,x)\right]|_{x=x_1}\ \xi_x(\lambda,x) - \right.$$

$$\left. -\left[\alpha_1\frac{d\xi_x(\lambda,x)}{dx} + \beta_1\xi_x(\lambda,x)\right]|_{x=x_2}\ \psi_x(\lambda,x)\right\}, \quad (3.40)$$

де $\xi_x(\lambda,x)$, $\psi_x(\lambda,x)$ -- система ортогональних функцій, що відповідає оператору L_x і межовим умовам.

Власні значення λ_n визначаються із рівняння

$$\alpha_2\frac{\varphi_x(\lambda,x_2)}{dx} + \beta_2\varphi_x(\lambda,x_2) = 0. \quad (3.41)$$

Диференціальному виразу (3.38) відповідає у просторі зображень вираз

$$-\lambda_n^2 \overline{X}(\lambda_n; y,z,t) + R(\lambda_n, y,z,t), \tag{3.42}$$

де $R(\lambda_n; y,z,t)$ визначається виразом

$$\frac{\gamma_2(t)}{\alpha_2}\left[p(x)\varphi_x(\lambda_n,t)\right]_{x=x_2} - \frac{\gamma_1(t)}{\alpha_1}\left[p(x)\varphi_x(\lambda_n,t)\right]_{x=x_1}, \tag{3.43}$$

якщо $\alpha_1 \neq 0$ і $\alpha - 2 \neq 0$, і виразом

$$\frac{\gamma_2(t)}{\beta_2}\left[p(x)\frac{d\varphi_x(\lambda_n,t)}{dx}\right]_{x=x_2} + \frac{\gamma_1(t)}{\beta_1}\left[p(x)\frac{d\varphi_x(\lambda_n,t)}{dx}\right]_{x=x_1}, \tag{3.44}$$

якщо $\alpha_1 = \alpha_2 = 0$.

Зворотне перетворення дає

$$X(x; y,z,t) = \sum_{n=1}^{\infty} \varphi_x(\lambda_n, x)\overline{X}(\lambda_n, t, y, z), \tag{3.45}$$

$$\overline{u}_0(\lambda_n; y,z) = \int_{x_1}^{x_2} u_0(x; y,z)\varphi_x(\lambda_n, x)dx, \tag{3.46}$$

$$\overline{u}_1(\lambda_n; y,z) = \int_{x_1}^{x_2} u_1(x; y,z)\varphi_x(\lambda_n, x)dx, \tag{3.47}$$

Застосування аналогічного скінченного інтегрального перетворення за змінними y та z до рівняння (3.37) із межовими умовами, що аналогічні умовам (3.35), (3.36), призводить до звичайного диференціального рівняння відносно функції $\overline{\overline{\overline{u}}}(t)$:

$$a_0\frac{d^2\overline{\overline{\overline{u}}}(t)}{dt^2} + a_1\frac{d\overline{\overline{\overline{u}}}(t)}{dt} + \mu^2\overline{\overline{\overline{u}}}(t) = \overline{\overline{\overline{h}}}(t) + \overline{\overline{\overline{R}}}(t, \lambda_n, \beta_{nk}, \mu_{nkj}) = H_{nkj}(t) \tag{3.48}$$

$$\overline{\overline{\overline{u}}}(\lambda_n, \beta_{nk}, \mu_{nkj})\big|_{t=0} = \overline{\overline{\overline{u}}}_0(\lambda_n, \beta_{nk}, \mu_{nkj}); \quad \frac{d\overline{\overline{\overline{u}}}(\lambda_n, \beta_{nk}, \mu_{nkj})}{dt}\Big|_{t=0} = \overline{\overline{\overline{u}}}_1(\lambda_n, \beta_{nk}, \mu_{nkj}) \tag{3.49}$$

Застосування до задачі Коші (3.48), (3.49) інтегрального перетворення Лапласа за змінною t дає

$$Q(\mu_{nkj},p)=\frac{(p+\alpha)\overline{\overline{\overline{u}}}_0(\lambda_n,\beta_{nk},\mu_{nkj})+\alpha\overline{\overline{\overline{u}}}_1(\lambda_n,\beta_{nk},\mu_{nkj})}{(p+\alpha)^2\pm\omega^2}+\frac{1}{a_0}\frac{\overline{H}_{nkj}(p)}{(p+\alpha)^2\pm\omega^2},\quad(3.50)$$

де $\alpha=-a_1/(2a_0)$; $\omega_{nkj}=1/(2a_0)\sqrt{a_1^2-\mu_{nkj}^2a_0}$.

$$Q(\mu_{nkj},p)=\int_0^\infty\overline{\overline{\overline{u}}}(\lambda_n,\beta_{nk},\mu_{nkj},t)e^{-pt}dt.$$

Якщо $a_0\neq0$, маємо

$$\overline{\overline{\overline{u}}}(\lambda_n,\beta_{nk},\mu_{nkj},t)=e^{-\alpha t}\begin{cases}p_0\cos\omega_{nkj}t+\dfrac{1}{\omega_{nkj}}[p_1+\alpha p_0]\sin\omega_{nkj}t, & 4\mu_{nkj}^2a_0>a_1^2,\\ p_0ch\omega_{nkj}t+\dfrac{1}{\omega_{nkj}}[p_1+\alpha p_0]sh\omega_{nkj}t, & 4\mu_{nkj}^2a_0<a_1^2\end{cases}+$$

$$+\frac{1}{a_0\omega_{nkj}}e^{-\alpha t}\int_0^t H_{nkj}(\tau)e^{\alpha(t-\tau)}\begin{cases}\sin\omega_{nkj}(t-\tau), & 4\mu_{nkj}^2>a_1^2,\\ \cos\omega_{nkj}(t-\tau). & 4\mu_{nkj}^2<a_1^2\end{cases}d\tau,\qquad(3.51)$$

де $p_0=\overline{\overline{\overline{u}}}_0(\lambda_n,\beta_{nk},\mu_{nkj})$, $p_1=\overline{\overline{\overline{u}}}_1(\lambda_n,\beta_{nk},\mu_{nkj})$.

У разі коли $a_0=0$ маємо випадок параболічного рівняння і

$$\overline{\overline{\overline{u}}}(\lambda_n,\beta_{nk},\mu_{nkj},t)=e^{-\mu_{nkj}^2t}\overline{\overline{\overline{u}}}_0(\lambda_n,\beta_{nk},\mu_{nkj})+e^{-\mu_{nkj}^2t}\int_0^t H_{nkj}(\tau)e^{\omega_{nkj}(t-\tau)}d\tau.\quad(3.52)$$

Розв'язок крайової задачі (3.34), (3.31),(3.35),(3.36) отримуємо у вигляді

$$u(x,y,z,t)=\sum_{n=1}^{\infty}\sum_{k=1}^{\infty}\sum_{j=1}^{\infty}\varphi_x(\lambda_n,x)\varphi_y(\beta_{nk},y)\varphi_z(\mu_{nkj},z)\overline{\overline{\overline{u}}}(\lambda_n,\beta_{nk},\mu_{nkj},t).\quad(3.53)$$

Отриманий розв'язок у вигляді розкладання у нескінченний ряд Фур'є за власними функціями відносно лінійної частини вихідної крайової задачі (3.30)—(3.32) є точний, але для практичної реалізації він непридатний. Тому необхідно розглядати скінченний (усікнений) ряд відносно всіх власних функцій у (3.53).

$$u(x,y,z,t)=\sum_{k=1}^{N_x}\sum_{l=1}^{N_y}\sum_{m=1}^{N_z}\varphi_x(\beta_k^x,x)\varphi_y(\beta_l^y,y)\varphi_z(\beta_m^z,z)T(\mu_{mlk},t).\qquad(3.54)$$

де $T(t) = \overline{\overline{\overline{u}}}(\mu_{mlk}, t)$.

Кількість членів розкладання у ряд (3.54) можна визначити моделюванням цього виразу залежно від необхідної точності отримання результату шляхом аналізу членів розкладання $\overline{\overline{\overline{u}}}(\beta_n^z, \beta_l^y, \beta_m^z, t)$ послідовно під час визначення образів функцій $H(x, y, z, t)$ і початкових умов u_0, u_1.

В отриманому розв'язку суттєвим є побудова власних функцій, які є ортонормовані й, отже, обмежені зверху одиницею. Це, по-перше, дозволяє обмежитися скінченними рядами відносно власних функцій за кожною із незалежних змінних, по-друге, гарантує збіжність ряду Фур'є (у разі обмеженості збурення $h(x, y, z, t)$ у правій частині рівняння (3.30)).

Інший шлях коректного усікнення ряду Фур'є – застосування до розв'язання (54) методу апроксимації дробово-раціональними функціями, який грунтується на апараті ланцюгових дробів, із подальшим переходом у часову область. У цьому разі критерієм точності апроксимації є критерій мінімуму середньоквадратичної похибки апроксимації.

3.3.2 Розв'язання нелінійної крайової задачі

Після отримання розв'язання лінійної частини задачі (3.30)—(3.32) – нульового наближення – подальші дії полягають у такому.

Побудуємо ітераційний процес:

$$a_0 \frac{\partial^2 u^{(m+1)}}{\partial t^2} + a_1 \frac{\partial u^{(m+1)}}{\partial t} = L[u^{(m)}(x, y, z, t)] + N[u^{(m)}(x, y, z, t)], \quad m = 0, 1, 2, \ldots \qquad (3.55)$$

де $N[u^{(m)}(x, y, z, t)]$ – нелінійна функція вихідного рівняння. До цієї функції приєднані нелінійні межові умови (3.32) за їх наявності. Вони приєднуються до функції N автоматично під час розв'язання лінійної частини крайової задачі за методом скінченних інтегральних перетворень за просторовими незалежними

змінними. Далі, оскільки на нульовій ітерації під час пошуку розв'язання лінійної частини рівняння цей розв'язок отримано у вигляді скінченного ряду за умови скінченності функції збурення, будь-який добуток скінченних рядів Фур'є також виявиться скінченним у силу ортонормованості систем базисних функцій $\varphi_k(x)$, $\varphi_l(y)$, $\varphi_m(z)$.

Маючи на увазі розв'язок лінійної частини крайової задачі у вигляді (3.54), підставимо його у вираз для нелінійної частини рівняння

$$N(u^{(0)}] = N\{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{n=1}^{N_z} T_{n,j,i}(t)\varphi_i(x)\varphi_j(y)\varphi_n(z)]$$

$$\left[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{n=1}^{N_z} T_{n,j,i}(t)\left(\frac{d\varphi_i(x)}{dx}\right)^2\left(\frac{d\varphi_j(y)}{dy}\right)^2\left(\frac{d\varphi_n(z)}{dz}\right)^2\right],\ldots\}.$$

Згідно із загальною схемою ітераційної процедури маємо:

$$\overline{\overline{\overline{N}}}[u^{(0)}(t)] = \int_{k=1}^{N_x}\varphi_k(x)\int_{l=1}^{N_y}\varphi_l(y)\int_{m=1}^{N_z}\varphi_m(z)\{[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{n=1}^{N_z} T_{n,j,i}(t)\varphi_i(x)\varphi_j(y)\varphi_n(z)]^2$$

$$\times[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{n=1}^{N_z} T^2_{n,j,i}(t)\left(\frac{d\varphi_i(x)}{dx}\right)^2\left(\frac{d\varphi_j(y)}{dy}\right)^2\left(\frac{d\varphi_n(z)}{dz}\right)^2],\ldots\}dxdydz.$$

У правій частині цього виразу містяться квадрати, куби тощо власних функцій лінійної задачі, а також похідних від цих функцій за просторовими незалежними змінними. Звідси, для визначення $\overline{\overline{\overline{N}}}[u^{(0)}(x,y,z,t)]$ досить обчислити відповідні інтеграли відносно відповідних степенів від базисних функцій.

Якщо позначити значення цих інтегралів через

$$nx1_i^k = \int_{k=1}^{N_x}\varphi_k(x)\varphi_i^2(x)dx;\ nx3_i^k = \int_{k=1}^{N_x}\varphi_k(x)\varphi_i^3(x)dx;$$

$$nx2_i^k = \int_{k=1}^{N_x}\varphi_k(x)\left(\frac{d\varphi_i(x)}{dx}\right)^2 dx;$$

$$ny1_i^l = \int_{l=1}^{N_y} \varphi_l(y)\varphi_j^2(y)dy;\ ny3_i^k = \int_{l=1}^{N_y} \varphi_l(y)\varphi_j^3(y)dy;$$

$$ny2_j^l = \int_{l=1}^{N_y} \varphi_l(y)\left(\frac{d\varphi_j(y)}{dy}\right)^2 dy;$$

$$nz1_i^m = \int_{m=1}^{N_z} \varphi_m(z)\varphi_n^2(z)dz;\ nz3_n^m = \int_{m=1}^{N_z} \varphi_n(z)\varphi_n^3(z)dz;$$

$$nz2_i^m = \int_{m=1}^{N_z} \varphi_n(z)\left(\frac{d\varphi_n(z)}{dz}\right)^2 dz;$$

отримаємо вираз вигляду

$$\overline{\overline{\overline{N}}}[u^{(0)}(x,y,z,t)] = F\left[\sum_{n,j,i}^{m,l,k}\left(T_{n,j,i}^{m,l,k}(t)\right)^2, \left(T_{n,j,i}^{m,l,k}(t)\right)^3, \ldots\right]$$

Оскільки у правій частині цього виразу містяться відомі функції від збурення у вихідному рівнянні, а також від початкової і межових умов крайової задачі, до цього виразу можна застосовувати інтегральне перетворення Лапласа за часовою змінною.

$$\mathsf{L}[\overline{\overline{\overline{N}}}[u^{(0)}(x,y,z,t)] = \mathsf{L}\left\{F\left[\sum_{n,j,i}^{m,l,k}\left(T_{n,j,i}^{m,l,k}(t)\right)^2, \left(T_{n,j,i}^{m,l,k}(t)\right)^3, \ldots\right]\right\} =$$

$$= \mathsf{L}[T1_{n,j,i}^{m,l,k}(t)] = \overline{T}1_{n,j,i}^{m,l,k}(p).$$

У разі крайової задачі для рівнянь параболічного типу $T1_{n,j,i}^{m,l,k}(t)$ можна записати у вигляді

$$T1_{n,j,i}^{m,l,k}(t) = A1_{n,j,i}^{m,l,k} + A2_{n,j,i}^{m,l,k}t + A3_{n,j,i}^{m,l,k}t^2 + \ldots + B1_{n,j,i}^{m,l,k}e^{-\mu_{n,j,i}^{m,l,k}t} + B2_{n,j,i}^{m,l,k}e^{-2\mu_{n,j,i}^{m,l,k}t} +$$

$$\mathsf{L}[T1_{n,j,i}^{m,l,k}(t)] = A1_{n,j,i}^{m,l,k}\frac{1}{p} + A2_{n,j,i}^{m,l,k}\frac{1}{p^2} + A3_{n,j,i}^{m,l,k}\frac{1}{p^3} + \ldots$$

$$+B1_{n,j,i}^{m,l,k}\frac{1}{p+\mu_{n,j,i}^{m,l,k}} + B2_{n,j,i}^{m,l,k}\frac{1}{p+2\mu_{n,j,i}^{m,l,k}} + \ldots$$

У разі рівняння гіперболічного типу --

$$T1^{m,l,k}_{n,j,i}(t) = A1^{m,l,k}_{n,j,i} + A2^{m,l,k}_{n,j,i}\,t + A3^{m,l,k}_{n,j,i}\,t^2 + \ldots + B1^{m,l,k}_{n,j,i}\sin(\mu^{m,l,k}_{n,j,i}t) + B2^{m,l,k}_{n,j,i}\cos(\mu^{m,l,k}_{n,j,i}t) +$$

$$\mathsf{L}[T1^{m,l,k}_{n,j,i}(t)] = A1^{m,l,k}_{n,j,i}\frac{1}{p} + A2^{m,l,k}_{n,j,i}\frac{1}{p^2} + A3^{m,l,k}_{n,j,i}\frac{1}{p^3} + \ldots$$

$$+B1^{m,l,k}_{n,j,i}\frac{\mu^{m,l,k}_{n,j,i}}{p^2 + \left(\mu^{m,l,k}_{n,j,i}\right)^2} + B2^{m,l,k}_{n,j,i}\frac{p}{p^2 + \left(\mu^{m,l,k}_{n,j,i}\right)^2} + \ldots$$

Коефіцієнти у виразах залежать від індексів m,l,k відносно базисних функцій і від індексів n,j,i, що є результат обчислення інтегралів. Отже, за цими індексами можна виконати операції підсумовування, після чого залишаться тільки суми відносно базисних функцій. Позначимо

$$C1_{m,l,k} = \sum_{n,j,i} A1^{m,l,k}_{n,j,i};\ C2_{m,l,k} = \sum_{n,j,i} A2^{m,l,k}_{n,j,i};$$

$$\sum_{n,j,i} B1^{m,l,k}_{n,j,i}\frac{1}{p + \mu^{m,l,k}_{n,j,i}} = P1_{m,l,k}(p).$$

$P1_{m,l,k}(p)$ – дробово-раціональний вираз відносно оператора p, степінь виразу має дорівнювати $M = N_x + N_y + N_z$.

З метою спрощення виразів такого вигляду пропонується застосовувати алгоритм еквівалентного спрощення, який грунтується на апараті ланцюгових дробів. Цей алгоритм викладено у розділі 4.

У результаті застосування алгоритму можна записати вираз для $\mathsf{L}[T1_{m,l,k}(t)]$ у такому вигляді:

$$\overline{T1}_{m,l,k}(p) = \frac{C1_{m,l,k}}{p} + \frac{C1_{m,l,k}}{p^2} + \ldots + \frac{a1 + a2\,p}{(p^2 + \alpha_1)^2 \pm \omega_1^2}\Big|_{m,l,k} + \frac{a3 + a4\,p}{(p^2 + \alpha_2)^2 \pm \omega_2^2}\Big|_{m,l,k}.$$

Такий підхід дозволяє запобігти надмірності у обчисленнях, що призводить до спрощення розв'язання на першій ітерації і отримати вираз у вигляді:

$$u^{(1)}(x,y,z,t) = u^{(0)}(x,y,z,t) + \sum_{m,l,k} Z\varphi_k(x)\varphi_l(y)\varphi_m(z)[C1_{m,l,k} + C2_{m,l,k}t + \ldots +$$

$$+(a1/\omega_1)_{m,l,k}\sin\omega_1 t + a2_{m,l,k}\cos\omega_1 t + (a2/\omega_2)_{m,l,k}\sin\omega_2 t + a2_{m,l,k}\cos\omega_2 t],$$

або, у разі дійсних коренів,

$$u^{(1)}(x,y,z,t) = u^{(0)}(x,y,z,t) + \sum_{m,l,k} \varphi_k(x)\varphi_l(y)\varphi_m(z)[C1_{m,l,k} + C2_{m,l,k}t + \ldots +$$

$$+(a1/\omega_1)_{m,l,k}\omega_1 t + a2_{m,l,k}\omega_1 t + (a2/\omega_2)_{m,l,k}\omega_2 t + a2_{m,l,k}\omega_2 t).$$

Далі ітераційний процес за викладеною схемою продовжується доти, доки не буде отриманий розв'язок із потрібною точністю.

3.3.3 Оцінка точності розв'язання

Оцінка точності розв'язання на n-й ітерації визначається як норма у просторі L^2_Ω різниці отриманих розв'язань на двох сусідніх ітераціях:

$$\| u^{(n)}(x,y,z,t) - u^{(n-1)}(x,y,z,t) \|^2 = \int_\Omega [u^{(n)}(x,y,z,t) - u^{(n-1)}(x,y,z,t)]d\Omega =$$

$$= \left\{ \int_0^{L_x}\int_0^{L_y}\int_0^{L_z}\int_0^{T} \varphi_k(x)\varphi_l(y)\varphi_m(z)[U^{(n)}_{m,l,k}(t) - U^{(n-1)}_{m,l,k}(t)]dzdydxdt \right\}^2 .$$

Оскільки власні функції $\varphi_k(x)$, $\varphi_l(y)$, $\varphi_m(z)$ є ортонормовані, зазначена норма визначається як

$$\| u^{(n)}(x,y,z,t) - u^{(n-1)}(x,y,z,t) \|^2 = \left\{ \int_0^{T} [U^{(n)}_{m,l,k}(t) - U^{(n-1)}_{m,l,k}(t)]dt \right\}^2 .$$

Коефіцієнти у виразах $u^{(n)}(x,y,z,t)$ є функції від інтегральних перетворень за просторовими змінними, і, таким чином, враховується нелінійна залежність шуканого розв'язання.

Критерій точності розв'язання є

$$\| u^{(m+1)} - u^{(m)} \|_{L^2_G} < \varepsilon, \quad m = 1,2,\ldots \tag{3.56}$$

При дослідженні високотемпературних процесів умову (3.56) доцільно

замінити на умову

$$\frac{\left\| u^{(m+1)} - u^{(m)} \right\|_{L_G^2}}{\left\| u^{(m+1)} \right\|_{L_G^2}} < \varepsilon, \quad m = 1,2,\ldots \tag{3.57}$$

3.4 Індукційний нагрів

У якості першого застосування пропонованого методу розв'язання нелінійних крайових задач для рівнянь із частинними похідними математичної фізики розглянемо задачу про вирощування монокристалів.

Розглянемо задачу про розподіл температурного поля злитку при його комбінованому нагріві індукційним струмом із біних сторін і плазмовому нагріві верхнього торця злитку із одночасною наплавкою присадного матеріалу.

Для забезпечення високої чистоти монокристалу необхідно забезпечити оптимальний перепад температур за його товщиною (оптимальний температурний градієнт).

Вплив плазмової дуги на верхній торець злитку розглядатимемо як поверхневе джерело теплового впливу, нехтуючи проникненням стовпа дуги у метал, що розплавився. Для визначеності вважатимемо, що напрямок руху джерела збігається із віссю Оу. Форму рухомого джерела із гаусовим розподілом впливу оцінюють ефективним діаметром d_{ef}. За відстані $d_{\text{ef}}/2$ від центру джерела густина впливу має складати визначену частку $c = 0,05$ її максимального значення. Отже,

$$Q(x,y,t) = q_d(t)\varphi(x,y,t) = q_d(t)\frac{1}{2\pi\sigma_x\sigma_y}\exp\left[-\frac{1}{2}\left(\frac{y - v_y t}{\sigma_y}\right)^2\right]\exp\left[-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2\right].$$

3.4.1 Математична модель

Знайдемо розв'язання крайової задачі, яка описує температурне поле у злитку прямокутного перетину висоти h під впливом внутрішнього джерела тепла w.

$$\rho c_{v_0}\frac{\partial T}{\partial t}=\lambda_0\Delta T+\frac{\partial\lambda}{\partial T}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}+\frac{\partial T}{\partial z}\right)=\lambda_0\Delta T+w+N(T). \tag{3.38}$$

$$N(T)=(a_1+2a_2T)\left[\left(\frac{\partial T}{\partial x}\right)^2+\left(\frac{\partial T}{\partial y}\right)^2+\left(\frac{\partial T}{\partial z}\right)^2\right]-(\frac{c_{v1}}{c_{v0}}T+\frac{c_{v2}}{c_{v0}}T^2)\frac{\partial T}{\partial t} \tag{3.59}$$

Межові умови:

$$\left.\frac{\partial T}{\partial x}\right|_{x=0}=0;\quad -\lambda_0\left.\frac{\partial T}{\partial x}\right|_{x=b/2}=\Delta p_x(T)\,|_{x=b/2} \tag{3.60}$$

$$\left.\frac{\partial T}{\partial y}\right|_{y=0}=0;\quad -\lambda_0\left.\frac{\partial T}{\partial y}\right|_{y=d/2}=\Delta p_y(T)\,|_{y=d/2} \tag{3.61}$$

$$\begin{aligned}\left.\frac{\partial T}{\partial z}\right|_{z=0}&=\Delta p_{z_0}(T)\,|_{z=0};\\ \left.\frac{\partial T}{\partial z}\right|_{z=h}&=\Delta p_{z_h}(T)\,|_{z=h}\,.\end{aligned} \tag{3.62}$$

Початкова умова

$$T(x,y,z,0)=T_0(x,y,z). \tag{3.63}$$

Питома потужність теплових витрат визначається за формулою

$$\Delta p_k(T)=\varepsilon\sigma(T^4-T_{c_k}^4)+\alpha_k(T-T_{c_k})=\alpha_kT-D_g+\varepsilon\sigma T^4 \tag{3.64}$$

$\alpha_0=\alpha_x;\ \alpha_1=\alpha_y;\ \alpha_2=\alpha_{z_0};\ \alpha_4=\alpha_{z_h}$.

$$D_{g_k}=T_c(\alpha_k+\varepsilon\sigma T_{c_k}^3). \tag{3.65}$$

Вважаємо, що коефіцієнти теплообміну із навколишнім середовищем є сталі, але їхні числові значення мають визначатися у процесі моделювання.

Фактично це означає, що у процесі моделювання прямої задачі теплопереносу необхідно вирішувати і зворотну задачу по визначенню коефіцієнтів теплообміну злитка із навколишнім середовищем.

Залежність коефіцієнтів теплопровідності і тепломісткості матеріалу від температури можна виразити залежностями

$$\begin{aligned} \lambda(T) &= \lambda_0 + \lambda_1 T + \lambda_2 T^2 + \lambda_3 T^3; \\ c_v(t) &= c_{v0} + c_{v1} T + c_{v2} T^2. \end{aligned} \tag{3.66}$$

Значення цих коефіцієнтів визначаються на основі експериментальних даних. Для вольфраму отримані такі залежності:

$$\begin{aligned} \lambda_W(T) &= 196,4 - 0,1357T + 0,463 \cdot 10^{-4} T^2 + 0,362 \cdot 10^{-9} T^3 \,(\text{Вт/м}K); \\ c_{v_W}(T) &= 130 + 0,0136T + 0,404 \cdot 10^{-5} T^2 \,(Дж/кгK). \end{aligned} \tag{3.67}$$

Розв'язання сформульованої крайової задачі шукатимемо за ітераційною схемою. Спочатку знайдемо розв'язання лінійної частини цієї задачі.

3.4.2 Розв'язання задачі у лінійному наближенні

Випишемо лінійну частину крайової задачі (3.58)—(3.64).

$$\rho c_{v_0} \frac{\partial T^{(0)}}{\partial t} = \lambda_0 \left(\frac{\partial^2 T^{(0)}}{\partial x^2} + \frac{\partial^2 T^{(0)}}{\partial y^2} + \frac{\partial^2 T^{(0)}}{\partial z^2} \right) + w; \tag{3.68}$$

Межові умови:

$$\frac{\partial T^{(0)}}{\partial x}\Big|_{x=0} = 0; \quad \frac{\partial T^{(0)}}{\partial x} + \frac{\alpha_x}{\lambda_0} T^{(0)}\Big|_{x=b/2} = D_x = D_g/\lambda_0; \tag{3.69}$$

$$\frac{\partial T^{(0)}}{\partial y}\Big|_{y=0} = 0; \quad \frac{\partial T^{(0)}}{\partial y} + \frac{\alpha_x}{\lambda_0} T^{(0)}\Big|_{y=d/2} = D_y = D_g/\lambda_0; \tag{3.70}$$

$$\frac{\partial T^{(0)}}{\partial z} - \frac{\alpha_{z_0}}{\lambda_0} T^{(0)}\Big|_{z=0} = D_{z0}/\lambda_0; \quad \frac{\partial T^{(0)}}{\partial z} + \frac{\alpha_{z_h}}{\lambda_0} T^{(0)}\Big|_{z=h} = D_{zh}/\lambda_0. \tag{3.71}$$

Початкова умова: $T(x, y, z, 0) = T_0(x, y, z)$.

Застосуємо до крайової задачі (3.68)—(3.71) інтегральні перетворення за просторовими координатами. Власні функції отримано у вигляді

$$Z_n(z_n z) = \frac{1}{\mathrm{P}Z_n(z)\mathrm{P}}\left[\frac{\beta_{z0}}{z_n} sin(z_n z) + cos(z_n z)\right]; \tag{3.72}$$

$$Y_m(\nu_m y) = \frac{1}{\mathrm{P}Y_m(y)\mathrm{P}} cos(\nu_m y); \quad X_k(\mu_k x) = \frac{1}{\mathrm{P}X_k(x)\mathrm{P}} cos(\mu_k x). \tag{3.73}$$

Власні значення z_n, ν_m μ_k отримуємо як розв'язання таких рівнянь.

$$s\sin s - d/2\beta_y \cos s = 0; \quad r\sin r - b/2\beta_x \cos r = 0. \tag{3.74}$$

де $\beta_{x_i} = \alpha_{x_i}/\lambda_0$.

$$(\beta_{z_0} + \beta_{z_h})vcos(vh) + (\beta_{z_0}\beta_{z_h} - v^2)\sin(vh) = 0; \quad z_n = v_n. \tag{3.75}$$

Виконаємо ортогоналізацію отриманих систем базисних функцій. Покладемо

$$W_1(z) = \frac{Z_1(z)}{\mathrm{P}Z_1(z)\mathrm{P}}. \tag{3.76}$$

Решту базисних функцій визначимо за формулою

$$W_k(z) = \frac{Z_k(z) - \sum_{i=1}^{k-1} c_{k,i} W_i(z)}{\mathrm{P}Z_k(z) - \sum_{i=1}^{k-1} c_{k,i} W_i(z)\mathrm{P}}. \tag{3.77}$$

Вагові коефіцієнти:

$$c_{k,i} = \int_0^H Z_k(z) W_i(z) dz; \quad i = \overline{1, k-1}; k = \overline{2, N}. \tag{3.78}$$

Для запобігання повторних обчислень інтегралів вигляду (3.78) подамо власні функції $W_k(z)$ у такому вигляді

$$W_k(z) = \sum_{i=1}^{k} e_{k,i} Z_i(z). \tag{3.79}$$

Тоді

$$
\begin{aligned}
\|W_k(z)\| &= \sqrt{\int_0^H [Z_k(z)]^2 dz + \sum_{i=1}^{k-1} c_{k,i}^2 \cdot \|W_i(z)\|^2}; \\
c_{k,i} &= -\int_0^H Z_k(z) \left[\sum_{j=1}^{i} e_{i,j} Z_j(z) \right] dz; \\
e_{k,i} &= \sum_{j=1}^{k-1} e_{k-j,i} \sum_{l=1}^{j} e_{j,l} c_{k,l}; \quad i = \overline{1, k-1}; \quad k = \overline{2, N}. \\
e_{k,k} &= \frac{1}{\|W_k(z)\|}; \quad e_{k,i} = \frac{c_{k,i}}{\|W_k(z)\|}.
\end{aligned}
\tag{3.80}
$$

Застосуємо до крайової задачі (3.68)—(3.71) інтегральне перетворення за змінною z. Отримуємо таку крайову задачу:

$$
\frac{\partial \overline{T^{(0)}}}{\partial t} = a^2 \left(\frac{\partial^2 \overline{T^{(0)}}}{\partial x^2} + \frac{\partial^2 \overline{T^{(0)}}}{\partial y^2} \right) + \frac{1}{\rho c_{v0}} \overline{w}_m + Rz_m; \tag{3.81}
$$

$$
\overline{w}_m(x, y, t) = p_0(t) \overline{w}_{0m}(x, y) = (q0 + q1t) \overline{w}_{0m}(x, y);
$$

Межові умови:

$$
\frac{\partial \overline{T^{(0)}}}{\partial x}\Big|_{x=0} = 0; \quad \frac{\partial \overline{T^{(0)}}}{\partial x} + \frac{\alpha_x}{\lambda_0} \overline{T^{(0)}}\Big|_{x=b/2} = \overline{D_x} = \frac{D_g}{\lambda_0} g z_m; \tag{3.82}
$$

$$
\frac{\partial \overline{T^{(0)}}}{\partial y}\Big|_{y=0} = 0; \quad \frac{\partial \overline{T^{(0)}}}{\partial y} + \frac{\alpha_x}{\lambda_0} \overline{T^{(0)}}\Big|_{y=d/2} = \overline{D_y} = \frac{D_g}{\lambda_0} g z_m; \tag{3.83}
$$

$$
Rz_m = Dz_h \left(\frac{\alpha_{z0}}{\zeta_m} \sin(\zeta_m h) + \cos(\zeta_m h) \right) - D_{z0}.
$$

$$
\overline{T^{(0)}}(x, y, \zeta_m, t) = \int_0^h T^{(0)}(x, y, z, t) W_m(z) dz.
$$

Застосування інтегральних перетворень за змінними y, x призводить до задачі Коші відносно функції часу:

$$
\frac{du(t)}{dt} + \mu_{m,l,k} u(t) = Tg_{m,l,k} + Q1_{m,l,k} t; \tag{3.84}
$$

$$
u(0) = T0 g z_m g y_l g x_k = u0_{m,l,k}. \tag{3.85}
$$

$$\mu_{m,l,k} = a^2(\zeta_m^2 + (\beta_l^y)^2 + (\beta_k^x)^2); \quad a^2 = \lambda_0/(\rho c_{v0});$$

$$Tg_{m,l,k} = Rz_m gy_l gx_k + D_y sy_l gx_k + D_x gy_l sx_k + \frac{1}{\rho c_{v0}} q0 qz_m qy_l qx_k; \qquad (3.86)$$

$$Q1_{m,l,k} = \frac{1}{\rho c_{v0}} q1 qz_m qy_l qx_k; \qquad (3.87)$$

$$u(t) = \overline{\overline{\overline{T^{(0)}}}} = \int_0^h \int_0^{d/2} \int_0^{b/2} T^{(0)}(x,y,z,t) W_m(z) Y_l(y) X_k(x) dz dy dx.$$

У виразах (84)—(86) запроваджено позначення:

$$gx_k = \frac{1}{\mathrm{P}X_k(x)\mathrm{P}} \int_0^{b/2} \cos(\beta_k^x x) dx; \quad gy_l = \frac{1}{\mathrm{P}Y_l(y)\mathrm{P}} \int_0^{d/2} \cos(\beta_l^y y) dy; \quad qz_m = \int_0^h W_m(z) dz.$$

$$qx_k = \frac{1}{\mathrm{P}X_k(x)\mathrm{P}} \cos(\beta_k^x b/2); \quad qy_l = \frac{1}{\mathrm{P}Y_l(y)\mathrm{P}} \cos(\beta_l^y d/2); \quad qz_m = W_m(h).$$

Застосуємо до задачі (3.84), (3.85) інтегральне перетворення Лапласа (за змінною t). Маємо

$$U(p) = \frac{1}{p^2 + \mu_{m,l,k}} \left(\frac{Q0_{m,l,k} + TR_{m,l,k}}{p} + \frac{Q1}{p^2} + u0_{m,l,k} \right) \qquad (3.88)$$

Звідси

$$u_{m,l,k}(t) = \overline{\overline{\overline{T^{(0)}}}}_{m,l,k} = \frac{Q0_{m,l,k} + TR_{m,l,k} - Q1_{m,l,k}/\mu}{\mu} + \frac{Q1_{m,l,k}}{\mu^2} t -$$

$$- \left(\frac{Q0_{m,l,k} + TR_{m,l,k} - Q1_{m,l,k}/\mu}{\mu} + u0_{m,l,k} \right) (1 - e^{-\mu t}). \qquad (3.89)$$

Отже, у лінійному наближенні розв'язання задачі (3.68)—(3.71) набуває вигляду

$$T^{(0)}(x,y,z,t) = \sum_{m=1}^{M} \sum_{l=1}^{L} \sum_{k=1}^{K} W_m(z) Y_l(y) X_k(x) u_{m,l,k}(t). \qquad (3.90)$$

3.4.3 Розв'язання задачі у нелінійному наближенні

Розв'язання нелінійної задачі (58)—(63) відшукуємо за ітераційною процедурою

$$T^{(n+1)}(x,y,z,t) = T^{(0)}(x,y,z,t) + \frac{1}{\rho c_{v0}} \sum_{m=1}^{M}\sum_{l=1}^{L}\sum_{k=1}^{K} W_m(z) Y_l(y) X_k(x) \times$$

$$\times \mathsf{L}_t^{-1}\left[\mathsf{L}_t[\overline{\overline{\overline{N}}}(T^{(n)}(x,y,z,t))] \right];\ n = 0,1,2,\ldots \qquad (3.91)$$

Запишемо загальний вираз нелінійної частини для задачі (3.58)—(3.63).

$$N(T^{(n)}(x,y,z,t))] = \left(\lambda_1 + 2\lambda_2 T^{(n)}\right)\left[\left(\frac{\partial T^{(n)}}{\partial x}\right)^2 + \left(\frac{\partial T^{(n)}}{\partial y}\right)^2 + \left(\frac{\partial T^{(n)}}{\partial z}\right)^2\right] -$$

$$-\rho c_{v1} T^{(n)} \frac{\partial T^{(n)}}{\partial t} + NRG;\ n = 0,1,2,\ldots \qquad (3.92)$$

$$NRG = \varepsilon\sigma((T^{(n)})^4 |_{z=h}\ qz_m gy_l gx_k$$

$$+(T^{(n)})^4 |_{y=d/2}\ gz_m qy_l gx_k + (T^{(n)})^4 |_{x=b/2}\ gz_m gy_l qx_k).$$

Із урахуванням отриманого розв'язку (3.90) нелінійна частина набуває вигляд

$$N[T^{(0)}] = \left[\lambda_1 + 2\lambda_2 \sum_{m,l,k} u_{m,l,k}(t) W_m(z) Y_l(y) X_k(x)\right] \times$$

$$\times\left\{\sum_{m',l',k'} u^2_{m',l',k'}(t)\left[\left(\frac{\partial X_{k'}}{\partial x}\right)^2 + \left(\frac{\partial Y_{l'}}{\partial y}\right)^2 + \left(\frac{\partial W_{m'}}{\partial z}\right)^2\right]\right\} -$$

$$-\rho c_{v1} \sum_{m,l,k} u_{m,l,k}(t) W_m(z) Y_l(y) X_k(x) \sum_{m,l,k} \frac{du_{m,l,k}(t)}{dt} W_m(z) Y_l(y) X_k(x) + NRG_{m,l,k}. \qquad (3.93)$$

Згідно із загальною ітераційною процедурою (3.91)

$$\overline{\overline{\overline{N}}}[T^{(0)}_{m,l,k}(x,y,z,t)] = \int_0^h \int_0^{d/2} \int_0^{b/2} W_m(z) Y_l(y) X_k(x) N[T^{(0)}] dz dy dx. \qquad (3.94)$$

Із урахуванням (3.93) та (3.94) для виконання інтегральних перетворень нелінійної функції $N[T^{(0)}]$ за просторовими координатами треба обчислити інтеграли:

$$nz1_{m,m'} = \int_0^h \left(\frac{\alpha_{h0}}{\zeta_m} \sin \zeta_m z + \cos \zeta_m z \right) \left(\alpha_{h0} \cos \zeta_{m'} z - \zeta_{m'} \sin \zeta_{m'} z \right)^2 dz;$$

$$nz2_{m,m'} = \int_0^h \left(\frac{\alpha_{h0}}{\zeta_m} \sin \zeta_m z + \cos \zeta_m z \right)^2 \left(\alpha_{h0} \cos \zeta_{m'} z - \zeta_{m'} \sin \zeta_{m'} z \right)^2 dz.$$

$$ny1_{l,l'} = \int_0^{d/2} Y_l(y) \left(\frac{\partial Y_{l'}}{\partial y} \right)^2 dy = (\beta_{l'}^y)^2 \int_0^{d/2} \cos(\beta_l^y y) \sin^2(\beta_{l'}^y y) dy;$$

$$ny2_{l,l'} = \int_0^{d/2} Y_l^2(y) \left(\frac{\partial Y_{l'}}{\partial y} \right)^2 dy = (\beta_{l'}^y)^2 \int_0^{d/2} \cos^2(\beta_l^y y) \sin^2(\beta_{l'}^y y) dy;$$

$$nx1_{k,k'} = \int_0^{b/2} X_k(x) \left(\frac{\partial X_{k'}}{\partial x} \right)^2 dx = (\beta_{k'}^x)^2 \int_0^{b/2} \cos(\beta_k^x x) \sin^2(\beta_{k'}^x x) dx;$$

$$nx2_{k,k'} = \int_0^{b/2} X_k^2(x) \left(\frac{\partial X_{k'}}{\partial x} \right)^2 dx = (\beta_{k'}^x)^2 \int_0^{b/2} \cos^2(\beta_k^x x) \sin^2(\beta_{k'}^x x) dx.$$

Для перших двох витків індуктора у верхній частині злитка (28 мм) питома потужність джерела нагріву приймалася рівною $60 Bt/(m^2 \cdot K)$. Для наступних трьох витків (40 мм) $Q_2 = 55$.

Результати моделювання розв'язання задачі наведені нижче.

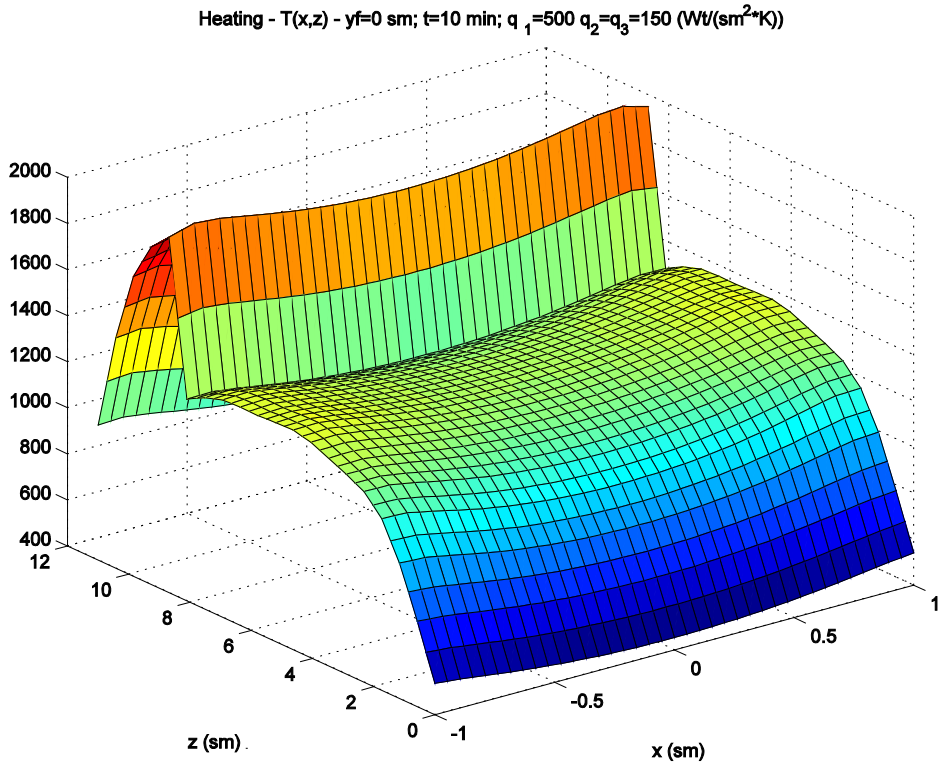


Рис. 3.1 Температурне поле у площині zx; $q_1 = 50; t = 10 min$

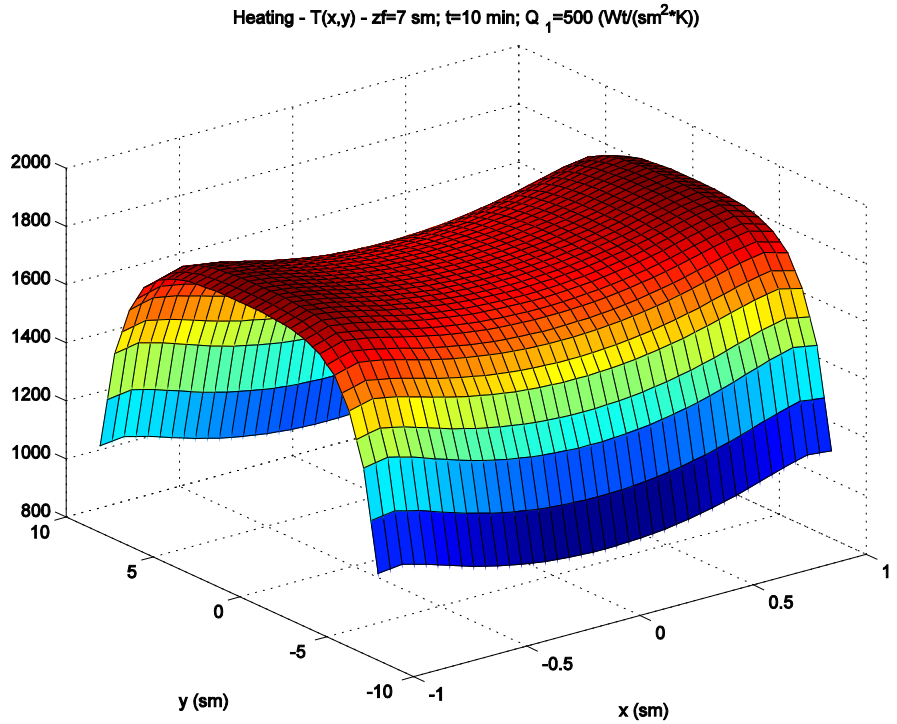


Рис. 3.2 Температурне поле у площині xy; $x = 0$ см. $q_1 = 50$

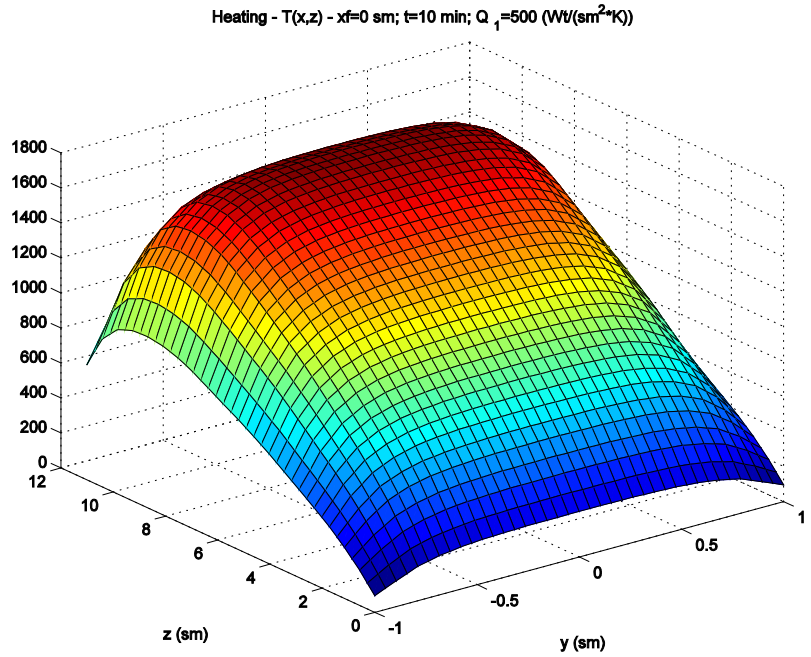


Рис. 3.3 Температурне поле у площині zy; $q_1 = 50; t = 10 min$

3.5 Комбінований нагрів злитка індуктором і плазмовим джерелом

Після нагріву злитка до потрібної температури починається процес плавлення верхнього торця злитка за допомогою впливу на нього плазмового джерела енергії (із одночасною наплавкою металу).

Вплив плазмового джерела енергії на верхній торець розглядатимемо як поверхневе джерело теплового впливу, нехтуючи процесом проникнення стовпа плазмової дуги всередину розплавленого металу під впливом тиску дуги і газів.

Як відомо, [61], поверхневе джерело (плазмова дуга), що рухається вздовж осі Oy із сталою швидкістю v_y, може бути наближено описаний двовимірним гаусовим розподілом

$$Q(x,y,t) = q_d \varphi(x,y,t) = q_d(t) \frac{1}{2\pi\sigma_x\sigma_y} \exp\left[-\frac{1}{2}\left(\frac{y - v_y t}{\sigma_y}\right)^2\right] \exp\left[-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2\right]. \quad (3.95)$$

Форму рухомого джерела із гаусовим впливом оцінюють ефективним діаметром $d_{\text{ýô}}$. При цьому на відстані $d_{\text{ýô}}/2$ від центра джерела густина впливу має складати визначену частку $c = 0,05$ її максимального значення. Для даного значення c ефективний діаметр двовимірного джерела зв'язаний із коефіцієнтом зосередженості співвідношенням

$$d_{\text{ýô}} = \sqrt{4\ln 1/c}\, 2\sigma_x\sigma_y = 3,46 \cdot 2\sigma_x\sigma_y. \qquad (3.96)$$

Відомий також опис рухомого джерела впливу за допомогою експоненційного розподілу розподілу Лапласа)

$$Q(x,y,t) = q_d(t)\frac{1}{4\sigma_x\sigma_y}\exp\left[-\frac{|\,y - v_y t\,|}{2\sigma_y}\right]\exp\left[-\frac{x}{2\sigma_x}\right]. \qquad (3.97)$$

Наведемо графіки функцій розподілу плазмових джерел енергії, що описуються функцією розподілу Гауса і Лапласа відповідно.

3.5.1 Математична модель комбінованого нагріву

Крайова задача, яка описує процес нагріву злитка плазмовою дугою, із урахуванням (3.96) може бути подана у такому вигляді:

$$\frac{\partial T}{\partial t} = a\Delta T + Q_I(x,z,t) + \frac{1}{\rho c_{v0}}N(T). \qquad (3.98)$$

Межові умови:

$$\frac{\partial T}{\partial x}|_{x=0} = 0; \quad \left[\frac{\partial T}{\partial x} + \beta_x T\right]_{x=b} = \beta_x(T_C + f_x(z)) - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_I^4\right)|_{x=b}. \qquad (3.99)$$

$$\frac{\partial T}{\partial y}|_{y=0} = 0; \quad \left[\frac{\partial T}{\partial y} + \beta_y T\right]_{y=d} = \beta_y T_C - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_I^4\right)|_{y=d}. \qquad (3.100)$$

$$\left[\frac{\partial T}{\partial z} - \beta_z T\right]_{z=0} = \beta_z T_c + \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_c^4\right)|_{z=0}. \qquad (3.101$$

$$\frac{\partial T}{\partial z}|_{z=h}=\frac{q_{\text{їё}}}{\lambda_0}(t)\varphi(x,y,t)-\frac{\varepsilon\sigma}{\lambda_0}\left(T^4-T_c^4\right)|_{z=h}. \qquad (1)$$

Початкова умова визначається температурним полем злитка, що сформувалося у результаті його попереднього нагріву в усталеному стані.

$$T(x,y,z,t_{st})=T_1(x,y,z). \qquad (3.103)$$

Із урахуванням отриманого розв'язку крайової задачі про розподіл температурного поля злитку при його індукційному нагріві початковий розподіл температури злитка при комбінованому плазмово-дуговому індукційному нагріві можна записати у такому вигляді:

$$T_0(x,y,z)=\sum_{l,j,i}\{Z_l(z)Y_j(y)X_i(x)Q^I_{l.j.i}+R^Y_{l.j.i}X_i(x)+$$

$$+Z_l(z)\left[R^y_{l.j.i}X_i(x)+Y_j(y)R^x_{l.j.i}\right]+R^z_{l.j.i}Y_j(y)X_i(x)\}. \qquad (3.104)$$

Сформульована крайова задача -- задача нагріву злитка плазмовою дугою – рухомим джерелом енергії. Варто мати на увазі, що із-за відносно малої швидкості руху плазмової дуги уздовж координати y плавлення металу здійснюється у зоні, що обмежена ефективним діаметром рухомого джерела у зоні його дії на поверхню злитку. На ділянці після проходження джерела температура поверхні металу дорівнюватиме приблизно температурі плавлення металу із подальшим його охолодженням і кристалізацією. На ділянці перед джерелом впливу на метал (за ходом руху плазмотрона) температура на поверхні металу дорівнюватиме температурі, що утворилася під впливом дії індуктора (приблизно 2000 $^\circ C$).

Отже, задача про розподіл температури злитка при його комбінованому індукційному і плазмово-дуговому нагріві фактично розбивається на три задачі: задачу власне плавлення поверхневого шару злитка під дією плазмової дуги на ділянці ефективного діаметра джерела енергії у зоні його дії, задачу теплопереносу за зоною дії джерела енергії і задачу теплопереносу при

індукційному нагріві перед зоною дії плазмової дуги.

У математичному формулюванні ці задачі описуються рівнянням (3.98), межовими умовами (3.99)—(3.102) і початковою умовою (3.103). Умова на верхній поверхні злитку $(z = h)$ набуває вигляду

$$\frac{\partial T}{\partial y}|_{y=0} = 0; \quad \left[\frac{\partial T}{\partial y} + \beta_y T\right]_{y=d_1} = \beta_y T_{mc}(z,t). \tag{3.105}$$

$$\left[\frac{\partial T}{\partial z} + \beta_{z_h} T\right]|_{z=h} = \beta_{z_h} T_{\text{їё}} - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_c^4\right)|_{z=h} \text{їðèd} - y + d_{\acute{y}\hat{o}} < \frac{y - v_y t}{2\sigma_y},$$

$$\frac{\partial T}{\partial z}|_{z=h} = \frac{q_{\text{їё}}}{\lambda_0}(t) \quad \varphi(x,y,t) - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_c^4\right)|_{z=h} \text{їðè}\frac{d - y - v_y t}{2\sigma_y} < 2d_{\acute{y}\hat{o}}, \tag{3.106}$$

$$\left[\frac{\partial T}{\partial z} + \beta_{z_h} T\right]|_{z=h} = \beta_{z_h} T_{\text{їё}} - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_c^4\right)|_{z=h} \text{їðèd} - y - d_{\acute{y}\hat{o}} > \frac{y - v_y t}{2\sigma_y}.$$

$$\left[\frac{\partial T}{\partial y}\beta_y T\right]|_{y=d_2} = \beta_y T_{mc}(z,t); \quad \left[\frac{\partial T}{\partial y} + \beta_y T\right]_{y=d} = \beta_y T_C - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_I^4\right)|_{y=d}. \tag{3.107}$$

Дві останні задачі розв'язуються за алгоритмами, аналогічними алгоритму розв'язання задачі індукційного нагріву із межовими умовами (3.104)—(3.107). Тому далі розглянемо розв'язання задачі тепломасоперенесення під впливом плазмової дуги.

Перейдемо до рухомої системи координат, що зв'язана із розташуванням центра рухомого джерела. Для цього зробимо заміну змінних

$$\psi = y - v_y t; \quad \tau = t. \tag{3.108}$$

Після виконання відповідних перетворень приходимо до такого рівняння.

$$\frac{\partial T}{\partial t} - v_y \frac{\partial T}{\partial \psi} = a\Delta T + Q_I(x,z,t) + \frac{1}{\rho c_{v0}} N(T). \tag{3.109}$$

Звільнимося від конвективного члена шляхом заміни

$$T(x,\psi,z,t) = u(x,\psi,z,t)\exp[-v_y/(2a)(\psi + v_y/2t)]. \tag{3.110}$$

Тоді рівняння (3.104) набуває стандартний вигляд

$$\frac{\partial u}{\partial t} = a\Delta u + Q_I(x,z,t)\exp\left[\frac{v_y}{2a}\left(\psi + \frac{v_y}{2}t\right)\right] + N(T). \qquad (3.111)$$

Межові умови за такої заміни записуються у вигляді

$$\frac{\partial u}{\partial x}|_{x=0} = 0; \qquad (3.112)$$

$$\left[\frac{\partial u}{\partial x} + \beta_x u\right]_{x=b} = \beta_x[T_C + f_x(z)]e^{v_y/2a(\psi+v_y/2t)} - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_I^4\right)|_{x=b}. \qquad (3.113)$$

$$\frac{\partial u}{\partial \psi}|_{\psi=0} = 0; \qquad (3.114)$$

$$\left[\frac{\partial u}{\partial \psi} + \beta_y u\right]_{y=d} = \beta_y[T_C e^{v_y/2a(\psi+v_y/2t)} - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_C^4\right)|_{x=b}. \qquad (3.115)$$

$$\left[\frac{\partial u}{\partial z} - \beta_z u\right]_{z=0} = \beta_z T_C e^{v_y/2a(\psi+v_y/2t)} + \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_C^4\right)|_{z=0}. \qquad (3.116)$$

$$\frac{\partial T}{\partial z}|_{z=h} = \frac{1}{\lambda_0}q_{\text{їё}}(t)\varphi(x,y,t)e^{v_y/2a(\psi+v_y/2t)} - \frac{\varepsilon\sigma}{\lambda_0}\left(T^4 - T_C^4\right)|_{z=h}. \qquad (3.117)$$

Початкова умова набуває вигляд

$$u(x,\psi,z,t_{st}) = T_I(x,y,z)\exp[v_y/(2a)(\psi + v_y/2t_{st})]. \qquad (3.118)$$

Знайдемо розв'язання лінійної частини сформульованої крайової задачі.

Застосуємо до крайової задачі інтегральні перетворення за просторовими змінними. Власні функції задачі отримані у вигляді

$$W_n(w_n z) = \frac{1}{\mathrm{P}W_n(z)\mathrm{P}}\left[\frac{\beta}{w_n}sin(w_n z) + cos(w_n z)\right]; \qquad (3.119)$$

$$Y_m(\xi_m y) = \frac{1}{\mathrm{P}Y_m(z)\mathrm{P}}cos(\xi_m y); \quad X_k(\mu_k x) = \frac{1}{\mathrm{P}X_k(z)\mathrm{P}}cos(\mu_k x). \qquad (3.120)$$

Позначимо

$$sy_m = \sqrt{\frac{2}{d}}\int_0^d \cos(\xi_m\psi)e^{v_y/(2a)\psi}\,d\psi = \sqrt{2/d}\,\frac{\xi_m(-1)^m e^{v_y/(2a)d} - v_y/(2a)}{\xi_m^2 + [v_y/(2a)]^2}. \qquad (3.121)$$

$$gy_m = sy_m + \sqrt{\frac{2}{d}} \int_{-d_{ef}}^{d_{ef}} \cos(\xi_m \psi) \exp\left[-\frac{1}{2}\left(\frac{v_y}{\sigma_y}\psi \right)^2 \right] d\psi. \qquad (3.122)$$

$$gx_k = \frac{1}{\mathrm{P}X_k(x)\mathrm{P}} \int_{-d_{ef}}^{d_{ef}} \cos(\mu_k x) \exp\left[-\frac{1}{2}\left(\frac{v_y}{\sigma_x}x \right)^2 \right] dx. \qquad (3.123)$$

$$H_I = \left(Q_z^I f_x - sx_k \beta_z T_c \right) sy_m; \quad H_{\text{їё}} = W_n(H) \frac{q_{\text{їё}}}{\lambda_0} gy_m gx_k.$$

У результаті застосування до крайової задачі (111)—(117) інтегральних перетворень за просторовими змінними z, ψ, x приходимо до такої задачі Коші.

$$\begin{aligned} \frac{dU(t)}{dt} \quad & + a_t \gamma_{n,m,k} U(t) = \left[\bar{Q}^I_{n,m,k} f_x sy_m + \frac{1}{\lambda} q_{\text{їё}} cw_n gy_m gx_k \right] e^{v_y^2/(4a)t} + \\ & + sy_m \left[sw_n cx_k (\beta_x T_I + \beta_z \hat{F}_z) - \beta_z T_c sx_k \right] e^{v_y^2/(4a)t} + N_T(u); \\ & U(t_0) = T^I_{n,m,k}. \end{aligned} \qquad (3.124)$$

Розв'язання лінійної частини задачі можна записати у такому вигляді

$$u^{(0)}(x,\psi,z,t) = \sum_{n,m,k} W_n(z) Y_m(\psi) X_k(x) \left[F1_{n,m,k} e^{-\alpha_{n,m,k} t} + F2_{n,m,k} e^{v_a t} \right]. \qquad (3.125)$$

У цій формулі позначено:

$$v_a = \frac{v_y^2}{4a}; \quad F2 = \frac{H_I + H_{\text{їё}}}{\alpha_{n,m,k} + v_a}; \quad F1 = T_I^H - F2_{n,m,k}. \qquad (3.126)$$

Повертаючись до вихідної змінної y, отримаємо вираз для температурного поля злитку у лінійному наближенні:

$$\begin{aligned} T^{(0)} \quad & (x,y,z,t) = u^{(0)}(x,\psi,z,t) \exp\left[-v_y/(2a)(\psi + v_y/2t \right] = \\ & = \sum_{n,m,k} W_n(z) Y_m(y) X_k(x) e^{-v_y/(2a)y} \left[F1_{n,m,k} e^{-(\alpha_{n,m,k} - v_a)t} + F2_{n,m,k} e^{2v_a t} \right]. \end{aligned} \qquad (3.127)$$

Результати моделювання отриманого розв'язку наведені нижче.

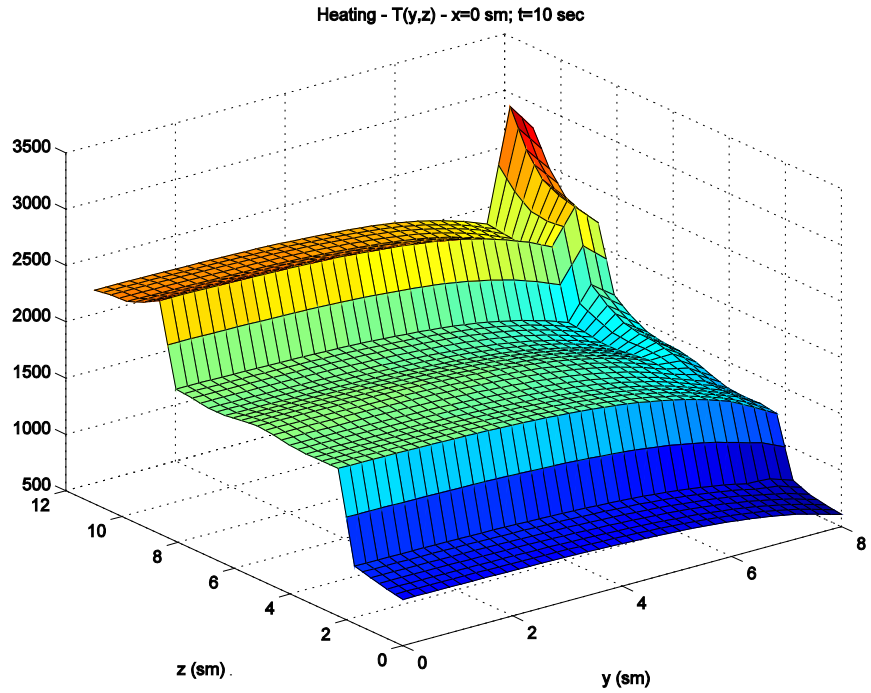


Рис 3.4 Температурне поле у площині zy ; $t = 10sec$

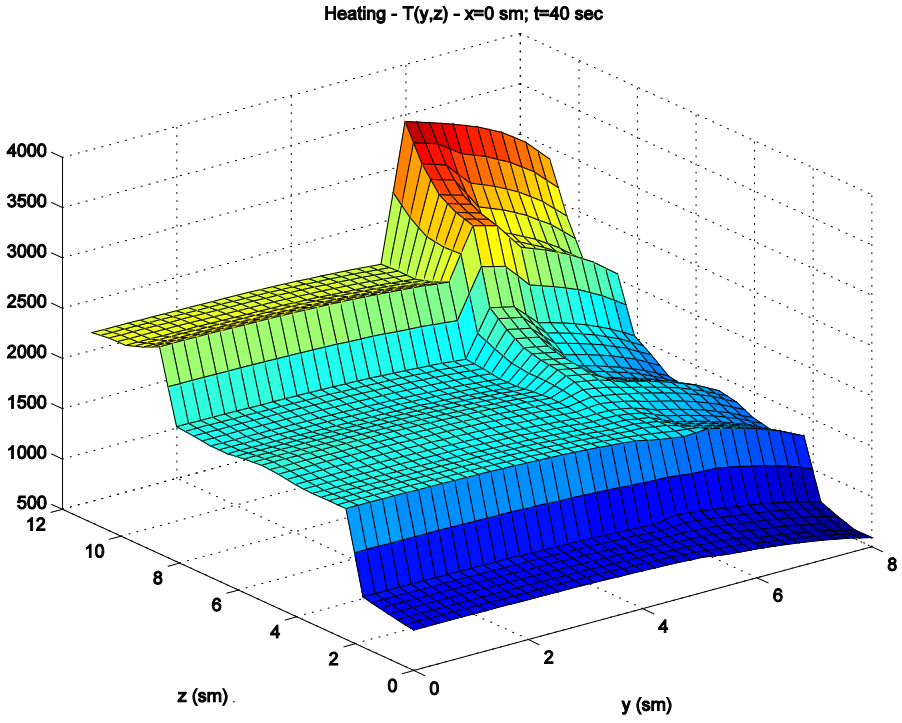


Рис. 3.5 Температурне поле у площині zy ; $t = 40sec$

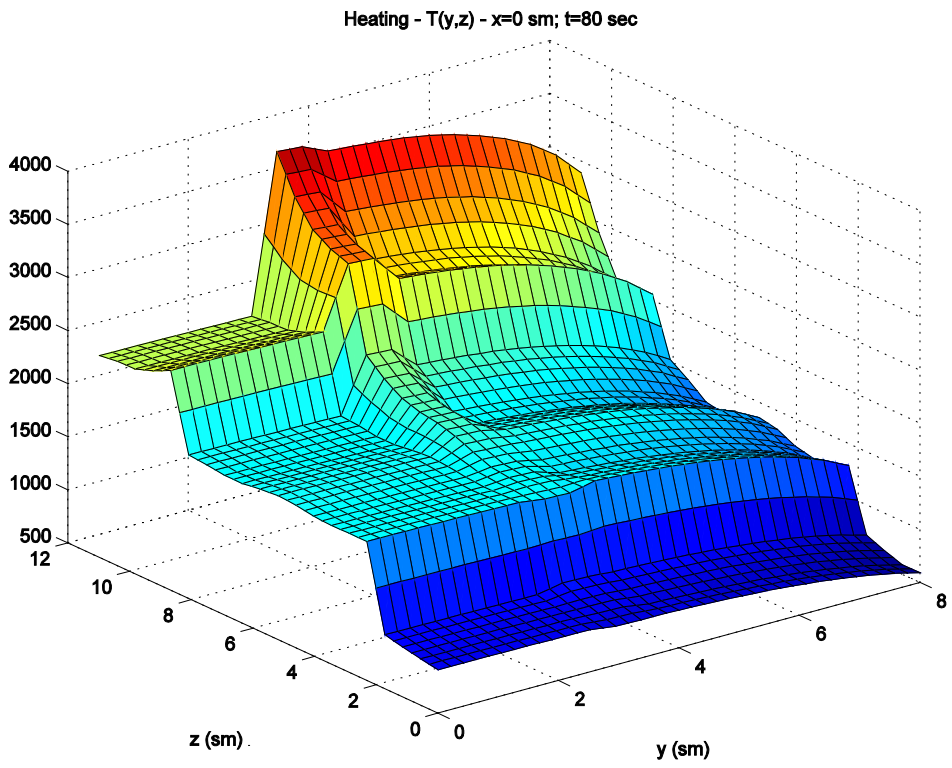


Рис. 3.6 Температурне поле у площині zy ; $t = 80sec$

3.6 Розв'язання крайових задач зі змінними межами (задач типу Стефана)

3.6.1 Постановка задачі

З точки зору обчислювальних алгоритмів важливе значення має той факт, що задача Стефана допускає формулювання, за якого умови на межі фазового переходу включаються у само рівняння теплопровідності. Таке формулювання має назву ентальпійного, [139].

Область $\Omega^+(t)$ рідинної фази, де температура перевищує температуру фазового переходу T^*, є $\Omega^+(t) = \{(x,y,z) \in \Omega, T(x,y,z) > T^*\}$. Відповідно, область $\Omega^-(t) = \{(x,y,z) \in \Omega, T(x,y,z) < T^*\}$. Аналогічні позначення використаємо і для теплофізичних величин у кожній фазі.

У твердій фазі маємо рівняння теплопровідності

$$c^{-}\rho^{-}\frac{\partial T^{-}}{\partial t} = \operatorname{div}(k^{-}\operatorname{grad}T^{-}) + f^{-},\ (x,y,z) \in Q^{-}, \tag{3.128}$$

де $Q^{-} = \{(x,y,z,t) \,|\, (x,y,z) \in \Omega^{-}, 0 < t < t_{\max}\}$.

Із урахуванням конвективного перенесення у рідинній фазі, отримаємо

$$c^{+}\rho^{+}\left(\frac{\partial T^{+}}{\partial t} + v\operatorname{grad}T^{+}\right) = \operatorname{div}(k^{+}\operatorname{grad}T^{+}) + f^{+},\ (x,y,z) \in Q^{+}. \tag{3.129}$$

$$[T] = 0,\ (x,y,z) \in S. \tag{3.130}$$

де $[\cdot]$ позначає стрибок на межі S фазового переходу.

Фазовий перехід супроводжується виділенням/поглинанням певної кількості тепла. Тому тепловий потік на межі фазового переходу є розривний і визначається величиною

$$\left[\frac{\partial T}{\partial n}\right] = -LV_{n},\ (x,y,z) \in S. \tag{3.131}$$

Тут L – ентальпія фазового переходу, V_n швидкість руху межі фазового переходу за нормаллю.

На межі фазового переходу виконуються умови 1-го роду:

$$T(x,y,z,t) = T^{*},\ (x,y,z) \in S(t). \tag{3.132}$$

Умови (3.130)– (3.131) – умови Стефана, а відповідна задача для рівнянь (3.128), (3.129) називається задачею Стефана.

Урахування теплоти фазового переходу еквівалентне завданню ефективної тепломісткості:

$$c_{ef} = c + \frac{1}{\rho}L\delta(T - T^{*}).$$

$$\rho c_{ef}\left(\frac{\partial T}{\partial t} + v\operatorname{grad}T\right) = \operatorname{div}(k\operatorname{grad}T) + q_{V},\ (x,y,z) \in Q. \tag{2}$$

q_V – об'ємна густина внутрішніх джерел.

3.6.2 Задача плавлення і випаровування матеріалу

Розглядається задача про плавлення і випаровування матеріалу, що поглинає потік енергії потужності Q. Математична модель, що описує цей процес, може бути записана у такому вигляді:

$$\rho c_i(T)\left(t_r \frac{\partial^2 T_i}{\partial t^2} + \frac{\partial T_i}{\partial t} + v_{ld}\frac{\partial T_i}{\partial x} + u\frac{\partial T_i}{\partial t} \right) = \mathrm{div}[\lambda(T_i)\mathrm{grad}T_i],\ i=1,2, \quad (3.134)$$

$$\left[\frac{\partial T_i}{\partial z} - \beta(T_L - t_0) \right]|_{z=0} = 0, \quad (3.135)$$

$$T_l(x,y,z,t)|_{S_l} = T_s(x,y,z,t)|_{S_s} = T_\mu; \quad (3.136)$$

$$T_l(x,y,z,t)|_{S_l} = T_e; \quad (3.137)$$

$$\lambda(T_l)\frac{\partial T_l}{\partial n_l}|_{S_l} = \lambda(T_s)\frac{\partial T_s}{\partial n_s}|_{S_s} - Q_m\frac{dn_s}{dt}; \quad (3.138)$$

$$T_s(x,y,z,t)|_{S_s} = T_0, \quad (3.139)$$

У цій моделі позначено: T_l, T_s – температура рідинної та твердої фаз відповідно; Q_i, Q_m – теплота випарювання і плавлення відповідно; T_l, T_m, T_0 – температура випарювання, плавлення та оточуючого середовища відповідно; $c_i(T), \lambda_i(T)$ – коефіцієнти об'ємної теплоємності та теплопровідності; Q – густина потоку енергії; S_l, S_s – рухомі межі розділу фаз ``парогазовий канал – рідинна фаза", та ``рідинна фаза – тверда фаза"; u – поздовжня складова швидкості руху рідинної фази; n_s, n_l – нормалі до поверхонь S_s, S_l; v_{ld} – швидкість зварювання; t_r – швидкість розповсюдження теплових збурень (для швидкоплинних процесів).

Залежності теплофізичних параметрів від температури визначаються за експериментальними дослідженнями. Вони можуть бути записані у вигляді

$$\lambda(T) = \lambda_0 + \lambda_1 T + \lambda_2 T^2,\ c_v(t) = c_{v0} + c_{v1}T + c_{v0} + c_{v2}T^2. \quad (3.140)$$

Значення цих коефіцієнтів залежать від матеріалу. Рівняння нормалі до

кривої $S-$ межі фазового переходу:

$$n = \text{grad}S,\ |n|^2 = \left(\frac{\partial S}{\partial x}\right)^2 + \left(\frac{\partial S}{\partial y}\right)^2 \neq 0.$$

Зробимо заміну змінних $\gamma = [r - v_1(t)]/a$, $\beta = -z/[b + s_2(t)]$, $\gamma \in [0,1]$, $\beta \in [0,1]$. Позначимо $q_0 = v_1/a$, $v_1 = v(t_0)$ – початкове значення; $\sigma = c_0/\lambda_0$; $\alpha = \gamma + q_0$. Тоді замість рівняння (134) матимемо:

$$\sigma\left(\frac{\partial T_i}{\partial t} + \frac{c_1}{\lambda_1}\frac{\partial T_i}{\partial \alpha} + \frac{u(\beta)}{\lambda_1 b}\frac{\partial T_i}{\partial \beta}\right) = \frac{1}{a^2}\left(\frac{\partial^2 T_i}{\partial \alpha^2} + \frac{1}{\alpha}\frac{\partial T_i}{\partial \alpha}\right) + \frac{1}{\lambda_1}\frac{1}{b^2}\frac{\partial^2 T_i}{\partial \beta^2} + N_1(T_i); \quad (3.141)$$

$$N_1(T_i) = \left(\frac{\lambda_1}{\lambda_0} + \frac{\lambda_2}{\lambda_0}T_i\right)\{T_i\left[\frac{1}{a^2}\left(\frac{\partial^2 T_i}{\partial \alpha^2} + \frac{1}{\alpha}\frac{\partial T_i}{\partial \alpha}\right) + \frac{1}{b^2}\frac{\partial^2 T_i}{\partial \beta^2}\right]$$

$$+\left[\frac{1}{a^2}\left(\frac{\partial T_i}{\partial \alpha}\right)^2 + \frac{1}{b^2}\left(\frac{\partial T_i}{\partial \beta}\right)^2\right]\}$$

$$+\left(\frac{c_0}{\lambda_0} + \frac{c_1}{\lambda_0}T_i + \frac{c_2}{\lambda_0}T_i^2\right)\left(\frac{1}{\alpha}\frac{dv}{dt}\frac{\partial T_i}{\partial \alpha} + \frac{1}{b}\frac{ds}{dt}\frac{\partial T_i}{\partial \beta}\right). \quad (3.142)$$

Межові умови для рівняння (3.141) набувають вигляду

$$T(\alpha,\beta,t)|_{\alpha=0} = T_s,\ \left(\frac{\partial T}{\partial \alpha} - T\right)|_{\beta=0} = 0;\ T(\alpha,\beta,t)|_{\alpha=1} = T_s;\ T(\alpha,\beta,t)|_{\beta=1} = T_s. \quad (3.143)$$

Задача (3.141), (3.143) доповнюється умовами сполучення на межах фаз (3.135), (3.139).

Будемо розв'язувати цю задачу у три етапи. Спочатку знайдемо розв'язок лінійної частини задачі (при $N_1(T) = 0$). Потім за ітераційною процедурою знайдемо розв'язок відповідної нелінійної задачі, задовольнивши нерівність

$$\|T^{(m+1)}(\alpha,\beta,t) - T^{(m)}(\alpha,\beta,t)\|^2_{L^2} < \varepsilon. \quad (3.144)$$

Далі, поклавши $T(\alpha,\beta,t) \approx T^{(m+1)}(\alpha,\beta,t)$, маємо розв'язок лінійної частини задачі (3.141), (3.143):

$$T^{(0)}(\alpha,\beta,t)=\sum_{n,k}R(\gamma_{nk}\alpha)Z(\sigma_k\beta)tt_{nk}^{(0)}(t); \qquad (3.145)$$

$$tt_{nk}^{(0)}(t)=A_{nk}^{(0)}+B_{nk}^{(0)}e^{-\beta_{nk}^{(0)}t}+e^{-\gamma_{nk}^{(0)}t}[C_{nk}^{(0)}\omega_{nk}^{(0)}t+D_{nk}^{(0)}\omega_{nk}^{(0)}t]. \qquad (3.146)$$

Вираз (3.146), а саме, коефіцієнти $A_{nk}^{(0)}$, $B_{nk}^{(0)}$, $C_{nk}^{(0)}$ і $D_{nk}^{(0)}$, а також $\omega_{nk}^{(0)}$ обчислюються за алгоритмами обчислення інтегральних перетворень, еквівалентного спрощення, що викладені у попередніх розділах.

Замістимо вираз (3.145) із урахуванням (3.146) у нелінійну частину (3.142) рівняння (3.141). Ця процедура докладно описана у розділі , тому тут наведено результат, який можна записати у вигляді виразу, аналогічному виразу (3.145).

$$T^{(m)}(\alpha,\beta,t)=\sum_{n,k}R(\gamma_{nk}\alpha)Z(\sigma_k\beta)tt_{nk}^{(m)}(t); \qquad (3.147)$$

$$tt_{nk}^{(m)}(t)=A_{nk}^{(m)}+B_{nk}^{(m)}e^{-\beta_{nk}^{(m)}t}+e^{-\gamma_{nk}^{(m)}t}[C_{nk}^{(m)}\omega_{nk}^{(m)}t+D_{nk}^{(m)}\omega_{nk}^{(m)}t]. \qquad (3.148)$$

Кількість ітерацій m визначається за умови виконання нерівності (3.144).

На третьому етапі із урахуванням отриманих розв'язків для рідинної та твердої фаз і заміщення цих виразів в умову сполучення (3.138) після перетворень отримуємо вираз для швидкості руху межі, що поділяє тверду і рідинну фази.

$$s(t)=s_1+e^{-\tau_s t}(s_2\sin\omega_s t+s_3\cos\omega_s t).$$

3.7 Висновки по розділу 3

1. Запропоновано числово-аналітичний метод розв'язання нелінійних крайових задач математичної фізики, сутність якого полягає у поданні вихідної крайової задачі у вигляді суми лінійної та нелінійної частин.

2. Для пошуку розв'язання задачі запропоновано ітераційний підхід, який грунтується на застосуванні скінченних інтегральних перетворень за просторовими змінними.

3. Точність запропонованого ітераційного методу визначається як норма у просторі $L^2[a,b]$ інтегровних функцій різниці між двома поточними ітераціями.

4. Запропоновано ітераційний метод розв'язання лінійних та нелінійних крайових задач із рухомими межами, відомих як задачі типу Стефана. Основу цього методу складає ітераційний числово-аналітичний метод розв'язання нелінійних крайових задач математичної фізики.

5. Запропоновані числово-аналітичні методи розв'язання нелінійних крайових задач із фіксованими і рухомими межами надають можливість виконувати математичне моделювання нелінійних задач математичної фізики широкого класу (теплових процесів, процесів фільтрації поверхневих і підземних вод, грунтових процесів, процесів хімічного виробництва тощо).

Розділ 4 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Постановка задач

У випадку декартових прямокутних координат інтегральні перетворення від нелінійних функцій зводяться до обчислення інтегралів у скінченних межах від добутку тригонометричних функцій, що є стандартна процедура і не викликає ніяких ускладнень. Приклад розв'язання подібної задачі наведено у попередньому розділі.

Ситуація стає суттєво складнішою при вирішенні нелінійних задач у циліндричній або сферичній системах координат. До таких задач належить широке коло задач, пов'язаних із гідродинамічними процесами в областях із циліндричною геометрією, що описуються системами нелінійних рівнянь Нав'є--Стокса, теорії пружності, теорії деформацій тощо.

Застосування інтегральних перетворень за просторовими змінними в ітераційній процедурі розв'язання нелінійних крайових задач пов'язано із необхідністю обчислення інтегралів від добутку двох, трьох та більше циліндричних чи сферичних функцій. Як відомо, інтеграли від добутку двох чи більше циліндричних функцій у квадратурах не існують (за винятком інтегралів від двох циліндричних функцій одного й того самого порядку (інтеграл Ломмеля).

Спроба обчислення такого типу інтегралів числовими методами призводить до суттєвих похибок внаслідок специфіки циліндричних функцій.

Отже, виникає необхідність у розробці методів наближеного інтегрування добутків циліндричних функцій, що забезпечували б прийнятну точність інтегрування.

Друга проблема, що виникає під розв'язання нелінійних крайових задач

математичної фізики, пов'язана із необхідністю виконання інтегральних перетворень від добутку шуканих розв'язків у разі розв'язання систем нелінійних рівнянь. Типовий приклад -- наявність у рівняннях Нав'є--Стокса конвективних складових

$$u\frac{\partial u}{\partial r}+v\frac{\partial u}{\partial \varphi}+w\frac{\partial u}{\partial z}$$

До цієї групи належать також добутки від шуканих функцій та їхніх похідних.

Розробка алгоритму обчислення інтегральних перетворень від такого роду функцій надає можливість автоматизувати весь процес розв'язання крайових задач для систем нелінійних рівнянь математичної фізики.

Питання еквівалентного спрощення дробово-раціональними виразами досліджувалося у кількох працях [2,158].

4.2 Апроксимація циліндричних функцій раціональними виразами

4.2.1 Відомості із теорії ланцюгових дробів [158]

Раціональні функції мають фундаментальну властивість залишатися раціональними при переміщенні й розтягуванні незалежної змінної. Найбільш важливою властивістю раціональних функцій є те, що раціональними функціями можна приблизити такі функції, які приймають нескінченні значення для кінцевих значень аргументу.

Нехай дана функція, для якої можна обчислити вузли. Необхідно за цими вузлами та значеннями функції у цих вузлах побудувати раціональну функцію

$$y = f(x) \approx \frac{N(x)}{D(x)}, \qquad (4.1)$$

де $N(x), D(x)$ – поліноми.

Розглянемо далі апроксимацію функцій раціональними виразами на основі апарату ланцюгових дробів. Найпростішим ланцюговим дробом називається вираз

$$b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4} + \cdots}}} \tag{4.2}$$

У більш загальному випадку параметри $a_i, b_i, i = 0,1,\ldots$ означають незалежні змінні. Будемо вважати $a_i, b_i, i = 0,1,\ldots$ додатними числами, b_0 – довільне дійсне число. Якщо кількість цих елементів обмежена, дроб виду (4.2) називається скінченним, або n-членовим ланцюговим дробом.

Запишемо (4.2) у компактнішому вигляді

$$b_0 + \frac{a_1}{b_1} + \frac{a_2}{b_2} + \cdots + \frac{a_n}{b_n} + \cdots, \tag{4.3}$$

а скінченний ланцюговий дроб у вигляді

$$b_0 + a_1 b_1 + a_2 b_2 + \cdots + a_n b_n. \tag{4.4}$$

Будь-який скінченний ланцюговий дроб виду (4.4), що отримано як результат скіченної кількості раціональних дій над його елементами, є раціональна функція цих елементів і може бути представлений як частка двох поліномів

$$P_n(a_1, a_2, \ldots, a_n; b_0, b_1, \ldots, b_n) Q_n(a_1, a_2, \ldots, a_n; b_0, b_1, \ldots, b_n),$$

яка має назву n-го підходящого дроба ланцюгового дроба (4.2). Чисельник та знаменник підходящого дроба визначається за рекурентним співвідношенням

$$\begin{aligned} P_n &= b_n P_{n-1} + a_n P_{n-2}; \\ Q_n &= b_n Q_{n-1} + a_n Q_{n-2}; \; n = 1,2,\ldots \end{aligned} \tag{4.5}$$

При цьому вважають

$$\frac{P_0}{Q_0}=\frac{1}{0};\quad \frac{P_1}{Q_1}=\frac{0}{1}. \tag{4.6}$$

Різниця між сусідніми підходящими дробами

$$\frac{P_n}{Q_n}-\frac{P_{n-1}}{Q_{n-1}}=\prod_{i=1}^{n}a_i\frac{(-1)^{n+1}}{(Q_{n-1}Q_n)},\ n=1,2,\ldots \tag{4.7}$$

Наведемо деякі властивості ланцюгових дробів. Множення елементів a_n, b_n, a_{n+1} на будь-яке скінчене число $p_n\neq 0$, $n=1,2,\ldots$ не змінює значення ланцюгового дроба:

$$b_0+\frac{a_1}{b_1}+\frac{a_2}{b_2}+\ldots+\frac{a_n}{b_n}=b_0+\frac{p_1a_1}{p_1b_1}+\frac{p_1p_2a_1}{p_2b_2}+\ldots+\frac{p_{n-1}p_na_n}{p_nb_n}+\ldots \tag{4.8}$$

Якщо покласти $p_n=1/a_n$,

$$\begin{aligned} &b_0+\frac{a_1}{b_1}+\frac{a_2}{b_2}+\quad\ldots+\frac{a_n}{b_n}=\frac{c_1}{1}+\frac{c_2}{1}+\ldots+\frac{c_n}{1}+\ldots,\\ &c_1=\frac{a_1}{b_1};\qquad c_k=\frac{a_k}{b_{k-1}b_k};\ k=2,3,\ldots \end{aligned} \tag{4.9}$$

Операція стиску:

$$\begin{aligned} b_0\quad+\quad&\frac{a_1}{b_1}+\frac{a_2}{b_2}+\ldots+\frac{a_n}{b_n}=b_0+\frac{a_1b_2}{b_1b_2+a_2}-\\ &-\frac{a_2a_3b_4}{(b_2b_3+a_3)b_4+b_2b_4}-\ldots-\frac{a_{2n-2}a_{2n-1}b_{2n-1}b_{2n}}{(b_{2n-2}b_{2n-1}+a_{2n-1})b_{2n}+b_{2n-2}b_{2n}} \end{aligned}$$

Стислий дроб для (9) має вигляд

$$\frac{c_1}{1}+\frac{c_2}{1}+\ldots+\frac{c_n}{1}=\frac{c_1}{1+c_3+c_4}-\ldots-\frac{c_{2n-2}c_{2n-1}}{1+c_{2n-1}+c_{2n}}-\ldots \tag{4.10}$$

Розглянемо розкладання дробово-раціональної функції у ланцюговий дроб.

$$f(x) = \frac{\alpha_{10} + \alpha_{11}x + \alpha_{12}x^2 + \ldots}{\alpha_{00} + \alpha_{01}x + \alpha_{02}x^2 + \ldots} = \frac{\alpha_{10}}{\alpha_{00}} + \frac{\alpha_{20}x}{\alpha_{10}} + \frac{\alpha_{30}x}{\alpha_{20}} + \ldots, \tag{4.11}$$

$$\alpha_{ik} = \alpha_{i-1,0}\alpha_{i-2,k+1} - \alpha_{i-2,0}\alpha_{i-1,k+1}; i = \overline{1,2n}; k = \overline{1,n}$$

за умови $\alpha_{k,0} \neq 0; \quad \alpha_{0,0} = 1$. Якщо деякий з коефіцієнтів $\alpha_{k,0} = 0$,

$$f(x) = \frac{\alpha_{1,0}}{1} + \frac{\alpha_{2,0}x}{\alpha_{1,0}} + \ldots + \frac{\alpha_{k-1,1}x}{\alpha_{k-2,0}} + \frac{\alpha_{k+1,1}x^2}{\alpha_{k-1,0}} + \frac{\alpha_{k+2,1}x^2}{\alpha_{k,0}} + \ldots \tag{4.12}$$

Розглянемо замість дроба (4.12) відповідний до нього підходящий дроб

$$f_1(x) = \frac{P_n(x)}{Q_n(x)}. \tag{4.13}$$

Підходящий дроб (4.13) апроксимує ланцюговий дроб (4.12) з похибкою

$$| f(x) - f_1(x) | \leq \frac{1}{\sqrt{5}Q_n(x)}.$$

Функцію $f_1(x)$ запишемо у такому вигляді

$$f_1(x) = \frac{\sum_{i=0}^{s} d_i x^i}{\sum_{j=0}^{r} c_j x^j}, \quad s < r, c_0 = 1. \tag{4.14}$$

Метод, що розглянуто вище, дозволяє апроксимувати функцію дробово-раціональними виразами виду (4.14).

4.2.2 Реалізація апроксимації раціональними виразами

Мета алгоритму реалізації апроксимації функцій раціональними виразами полягає у обчисленні коефіцієнтів $d_i, c_j; \quad i = \overline{0,s}; \quad j = \overline{0,r}$ за допомогою наступних формул.

• Обчислення коефіцієнтів ланцюгових дробів.

$$\begin{aligned}
&\alpha_{ik} && = \alpha_{i-1,0}\alpha_{i-2,k+1} - \alpha_{i-2,0}\alpha_{i-1,k+1}.\\
&\alpha_{k,0} \neq 0; && \alpha_{0.0} = 1; \quad i = \overline{1,n}; \quad k = \overline{1,n}.\\
&e_1 && = \alpha_{1,0}\alpha_{0,0}; e_{i+1} = \alpha_{i+1,0}\alpha_{i-1,0}\alpha_{i,0}; \quad i = \overline{1,n-1}.
\end{aligned}$$

• Обчислення коефіцієнтів апроксимуючого дробово-раціонального виразу.

$$\begin{aligned}
&d_0 && = b_0; c_0 = a_0 = 1;\\
&b_{2,2} && = 0; b_{3,2} = d_{2,1}e_0; b_{i,1} = e_1; a_{i,1} = 1; \ i = \overline{1,n};\\
&a_{2,2} && = e_1; a_{3,2} = e_1 + e_3;\\
&b_{k,i} && = e_k b_{k-2,i-1} + b_{k-1,1}; \ k = \overline{3,m}; i = \overline{2,[k/2]};\\
&a_{k,i} && = e_k a_{k-2,i-1} + a_{k-1,1}; \ k = \overline{3,m-1}; i = \overline{2,[k/2+1]};\\
&d_i && = b_{m,i}; \quad c_i = a_{m,i}; \quad i = \overline{1,n-1}.
\end{aligned}$$

За допомогою наведеного алгоритму із достатньою точністю можна апроксимувати поліноми високого порядку до полінома досить низького порядку (третього чи другого). Наприклад,

$$\sum_{k=0}^{6} \frac{b_{2k}p + b_{2k+1}}{p^2 + a_{2k+1}p + a_{2k+2}} = \frac{d_2 p + d_1}{p^2 + c_1 p + c_0}.$$

Вираз вигляду

$$\sum_{k=0}^{8} e^{\alpha_k t} \to \frac{b_2 p + b_1}{p^2 + a_1 p + a_2} \to e^{\gamma t}(d_1 \sin \omega t + d_2 \cos \omega t);$$

4.2.3 Апроксимація добутку циліндричних функцій

Спочатку розглянемо апрксимацію дробово-раціональним виразом функції Бесселя першого роду. Застосування функцій такого типу досить поширене при розв'язанні крайових задач для диференціальних рівнянь. Функції Бесселя записуються у вигляді

$$J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{n+2k}}{k!\Gamma(n+k+1)}. \qquad (4.15)$$

Поставимо задачу домогтися точності апроксимації $\varepsilon = 0,00001$.

Згідно з (4.15) та викладеним вище алгоритмом обчислення коефіцієнтів ланцюгового дробу одержуємо для дробу (9) $(n = 0)$:

$$c^{(0)} = \begin{bmatrix} 1 & 3.5 & -2.625 & 0.162 & -0.512 & 0.05449 & -0.2082 \\ 0.02744 & -0.1121 & 0.01652 & -0.06989 & 0.01103 & \cdots & \end{bmatrix}.$$

Апроксимуємо (4.15) дробово-раціональною функцією

$\overline{J}_0(z = (x/2)^2)$;

$$\frac{1 - 3.2556z + 2.2378z^2 - 0.54686z^3 + 0.054185z^4 - 0.0018907z^5}{1 + 0.24435z + 0.0305z^2 + 0.00254z^3 + 10^{-5}(15.23z^4 + 0.642z^5 + 0.0154z^6)}. \qquad (4.16)$$

Перетворимо цей дробово-раціональний вираз у суму ланок другого степеню. Для цього необхідно визначити корені знаменника. Маємо:

$$z^6 + 41.6z^5 + 9861.6z^4 + 4 \cdot 10^4 z^3 + 1.97 \cdot 10^5 z^2 1.58 \cdot 10^6 z + 6.47 \cdot 10^6 =$$
$$(179.26 + 16.793z + z^2)(164.03 + 24.529z + z^2)(220.11 + 0.24981z + z^2).$$

Після цього з метою визначення коефіцієнтів у чисельниках виразів другого степеню треба розв'язати відповідну систему лінійних алгебраїчних рівнянь. ($Ab = d$). Маємо:

$$A = \begin{bmatrix} 3.6110^4 & 0 & 3.9510^4 & 0 & 2.9410^4 & 0 & 1 \\ 5440 & 3.6110^4 & 3740 & 3.9510^4 & 7150 & 2.9410^4 & -3.26 \\ 390 & 5440 & 404 & 3740 & 755 & 7150 & 2.24 \\ 24.8 & 390 & 17 & 404 & 41.3 & 755 & -0.547 \\ 1 & 24.8 & 1 & 17 & 1 & 41.3 & 0.0542 \\ 0 & 1 & 0 & 1 & 0 & 1 & -0.002 \end{bmatrix};$$

$$b = \begin{bmatrix} 0.10496 & 0.010166 & -0.0782 & -0.012259 & -0.023905 & 0.00020249 \end{bmatrix}.$$

Отже, функція Бесселя першого роду 0-го порядку апроксимується

виразом

$$J_0(z);\ \frac{679310+65794z}{179.26+16.793z+z^2}+\frac{-506130-79342z}{164.03+24.529z+z^2}+\frac{-154720+1310z}{220.11+0.24981z+z^2}.$$

Оцінка похибки наближення: $J_0(7)=0.30010;\overline{J}_0(7)=0.30010$.

За таким же алгоритмом одержано апроксимуючу функцію для функції Бесселя першого роду 1-го порядку

$$J_1(z);\ \frac{106790+16219z}{96.498+12.66z+z^2}+\frac{-78570-18245z}{88.504+18.057z+z^2}+\frac{-25687-296.6z}{117.88+1.0346z+z^2}.$$

$z=\sigma_j/2x^2$.

У згорнутому вигляді апроксимацію функцій Бесселя першого роду n-порядку можна записати у такому вигляді

$$J_n(x)=\sum_{k=1}^{3}\frac{c^n_{6k+3}+c^n_{6k+4}x^2}{c^n_{6k}+c^n_{6k+1}x^2+c^n_{6k+2}x^4}, \qquad (4.17)$$

m – кількість власних значень при вирішенні відповідної задачі Штурма–Ліувілля.

Із урахуванням виразу (4.17) можна записати добуток двох функцій Бесселя першого роду

$$J_n(\sigma_j x)J_l(\sigma_i x)=\sum_{k=1}^{3}\frac{c^n_{6k+3}+c^n_{6k+4}x^2}{c^n_{6k}+c^n_{6k+1}x^2+c^n_{6k+2}x^4}\sum_{k=1}^{3}\frac{c^l_{6k+3}+c^l_{6k+4}x^2}{c^l_{6k}+c^l_{6k+1}x^2+c^l_{6k+2}x^4} \qquad (4.18)$$

Для того, щоб привести процес інтегрування добутку дробово-раціональних функцій до обчислення інтегралів від ланцюгів другого порядку, що є сандартна процедура математичного аналізу, подамо вираз (4.18) у вигляді суми ланцюгів другого порядку:

$$I2_{n,l}(x)=\int_0^R J_n(x)J_l(x)xdx$$

$$= \int_0^R \sum_{k=1}^{3} \frac{d_{6k+3} + d_{6k+4}x^2}{[d_{6k} + d_{6k+1}x^2 + d_{6k+2}x^4]^2} xdx + \int_0^R \sum_{k=4}^{9} \frac{d_{6k+3} + d_{6k+4}x^2}{d_{6k} + d_{6k+1}x^2 + d_{6k+2}x^4} xdx. \quad (4.19)$$

Для добутку 3-х функцій Бесселя

$$I3_{n,l,m}(x) = \int_0^R J_n(x)J_l(x)J_m(x)xdx =$$

$$= \int_0^R \sum_{k=1}^{3} \left[\frac{d_{6k+3} + d_{6k+4}x^2}{[d_{6k} + d_{6k+1}x^2 + d_{6k+2}x^4]^3} + \sum_{k=4}^{12} \frac{d_{6k+3} + d_{6k+4}x^2}{d_{6k} + d_{6k+1}x^2 + d_{6k+2}x^4} \right] xdx. \quad (4.20)$$

У випадку, коли задача поставлена для полого циліндру, замість функцій Бесслея першого роду у якості власних функцій задіяні функції Бесселя другого роду – функції Неймана

$$Y_n(\sigma_j x) = \frac{2}{\pi}(C + \log\frac{x}{2})J_n(x) - \frac{1}{\pi}\sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!}\left(\frac{x}{2}\right)^{2k-n}$$

$$-\frac{1}{\pi}\sum_{k=0}^{\infty} \frac{(-1)^k}{k!(n+k)!}\left(\frac{x}{2}\right)^{2k+n}\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{k}+1+\frac{1}{2}+\cdots+\frac{1}{n+k}\right) \quad (4.21)$$

При $k = 0$ вираз в останній дужці дорівнює

$$1+\frac{1}{2}+\cdots+\frac{1}{n}.$$

Апроксимація функції Неймана ускладнюється наявністю $\log x/2$. Застосування до виразу (4.21) апарату ланцюгових дробів дає

$$Y_n(\sigma_j x) = \sum_{k=1}^{3} \frac{c^n_{6k+3} + c^n_{6k+4}x^2}{c^n_{6k} + c^n_{6k+1}x^2 + c^n_{6k+2}x^4} + \sum_{k=1}^{3} \frac{c^l_{6k+3} + c^l_{6k+4}x}{c^l_{6k} + c^l_{6k+1}x + c^l_{6k+2}x^2} \quad (4.22)$$

При обчисленні інтегралу виконується множення сум ланцюгів другого порядку відносно 4-го степеню по x та другого степеню по x за рахунок апроксимації $\log x/2$.

Отже, добуток двох функцій Неймана дає вираз вигляду

$$Y_n(\sigma_j x)Y_l(\sigma_i x) = \sum_{k=1}^{3} \frac{b_{6k+3}^n + b_{6k+4}^n x^2}{[b_{6k}^n + b_{6k+1}^n x^2 + b_{6k+2}^n x^4]^2} + \sum_{k=1}^{3} \frac{b_{6k+3}^l + b_{6k+4}^l x}{[b_{6k}^l + b_{6k+1}^l x + b_{6k+2}^l x^2]^2}$$

$$+ \sum_{k=4}^{6} \frac{b_{6k+3}^n + b_{6k+4}^n x^2}{b_{6k}^n + b_{6k+1}^n x^2 + b_{6k+2}^n x^4} + \sum_{k=4}^{6} \frac{b_{6k+3}^l + b_{6k+4}^l x}{b_{6k}^l + b_{6k+1}^l x + b_{6k+2}^l x^2}.$$

Для добутку трьох функцій Неймана матимемо вирази, аналогічні (4.19),(4.20), але кількість складових ланцюгів другого порядку буде вдвічі більшою.

Вирази (4.17) – (4.22) отримуються для кожного власного значення σ_j, σ_i

.

Алгоритми й відповідні програми реалізції наведені у додатку.

Інша група алгоритмів пов'язана із визначенням власних значень для власних функцій задач Штурма –Ліувілля при виконанні скінченних інтегральних перетворень за рештою просторових змінних. До них належать задачі визначення коренів трансцендентних рівнянь, обчислення інтегралів від добутків власних функцій тощо.

4.2.4 Конвективні складові

Конвективні складові у рівняннях конвективно-дифузійного перенесення субстанції відносно компонент швидкості $(u = v_1, v = v_2, w = v_3)$ після застосування інтегральних перетворень за просторовими змінними можна подати у такому вигляді

$$v_1 \frac{\partial f_j}{\partial x} + v_2 \frac{\partial f_j}{\partial y} + v_3 \frac{\partial f_j}{\partial z}, \quad j = 1,2,3,4. \tag{4.23}$$

Тут $f_j = [v_1, v_2, v_3, T]$, T – субстанція, що досліджується (температура, концентрація домішків, напруженість, деформація тощо). Координати x, y, z можуть бути як прямокутними декартовими, так і циліндричними або

сферичними. У даному випадку це не суттєво, оскільки до виразу (4.23) застосовано інтегральні перетворення за просторовими змінними. Тож замість (4.23) маємо вираз:

$$V_1(t)\cdot T_j(t)+V_2(t)\cdot T_j(t)+V_3(t)\cdot T_j(t),\quad j=1,2,3,4. \tag{4.24}$$

Функції $V_k(t)$, $T_j(t)$ – це функції, що отримані внаслідок інтегральних перетворень за просторовими змінними і є функції часу, які отримуються внаслідок розв'язання системи рівнянь на попередній ітерації. Зокрема, перша ітерація – це розв'язок системи рівнянь у лінійному наближенні (див. попередній розділ). Наприклад,

$$V_1(t)T_j(t)=\overline{\overline{\overline{\overline{v_1\frac{\partial T}{\partial x}}}}}=\iiint_{R_0} v_1(x,y,z,t)T(x,y,z,t)X(x)Y(y)Z(z)dzdydx.$$

Отже, для кожної конвективної складової у просторі зображень матимемо вираз вигляду

$$f_1(t)f_2(t)=[a_0+e^{-\alpha_{nkj}t}(a_1g_1(\omega_{nkj}t)+a_2g_2(\omega_{nkj}t)]\times[b_0+e^{-\alpha_{nkj}t}(b_1h_1(\omega_{nkj}t)+b_2h_2(\omega_{nkj}t)], \tag{3}$$

де g_1,h_1 – $\sin rt$ або $sh\,rt$, g_2,h_2 – $\cos rt$ або $ch\,rt$, залежно від значень ω_{nkj}.

Оскільки інтегральне перетворення за часом (перетворення Лапласу) є лінійне, треба виконати операцію добутку у часі для кожного із конвективних членів ($j=\overline{1,4}$) Після виконання операції добутку отримаємо вираз, до якого можна застосовувати перетворення Лапласа:

$$\overline{V_1(t)T_j(t)}=\sum_{k=1}^{4}\left[\frac{c_{6k+2}}{p}+\frac{c_{6k}+c_{6k+1}p}{c_{6k+3}+c_{6k+4}p+c_{6k+5}p^2}\right].$$

Після застосування алгоритму еквівалентного спрощення до всіх складових ($j=\overline{1,4}$) можна записати

$$\mathsf{L}_t[V_1(t)T_j(t) + V_2(t)T_j(t) + V_3(t)T_j(t)] \approx \frac{d_2}{p} + \frac{d_0 + d_1 p}{d_3 + d_4 p + d_5 p^2}. \qquad (4.26)$$

Похибки апроксимації, що виникають, компенсуються за рахунок додаткових ітерацій.

На рис. 4.1 наведено приклад апроксимації добутку двох функцій часу дробово-раціональним виразом (4.26).

На рис. 4.2 наведено приклад апроксимації суми 4-х функцій часу як результат наближеного обчислення конвективної складової рівняння.

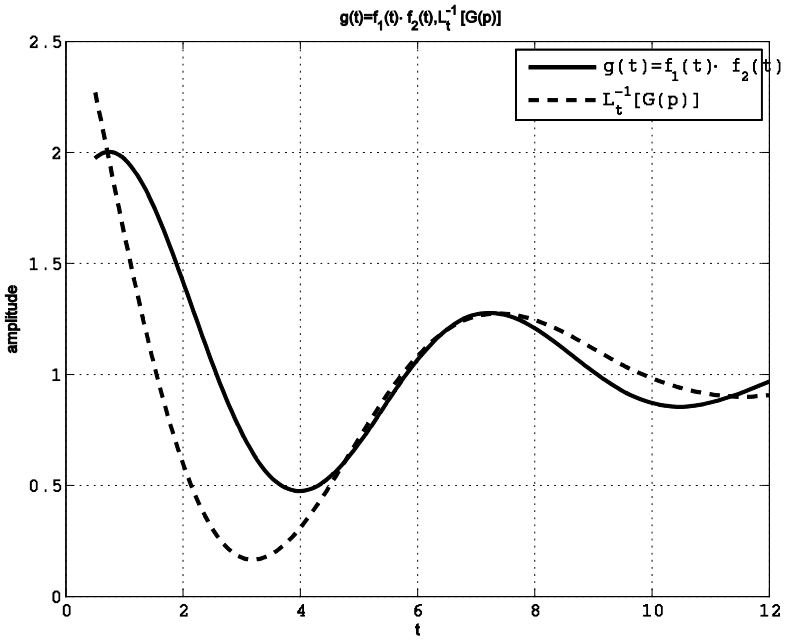


Рис. 4.1 Апроксимація $f_1(t) \cdot f_2(t)$

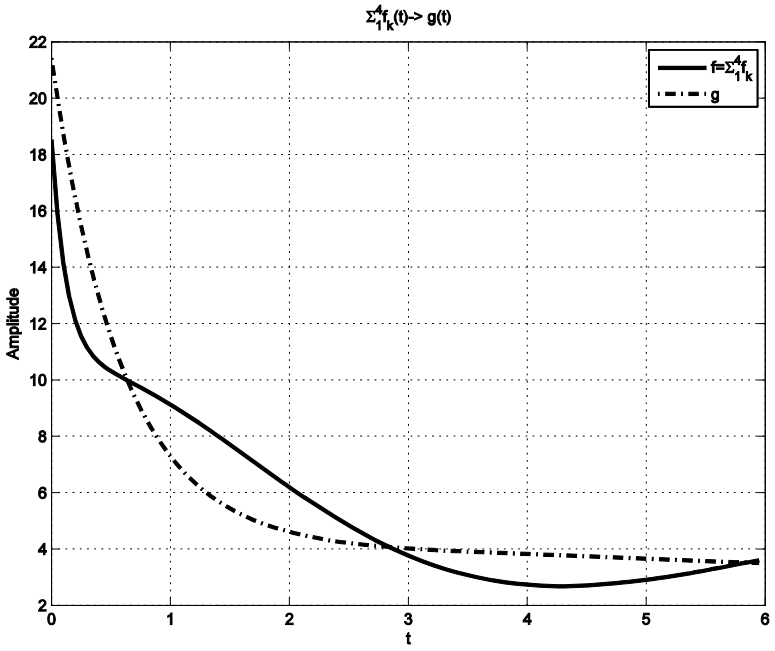


Рис. 4.2 Апроксимація $\Sigma_1^4 f_k(t)$

Алгоритм і програма його реалізації наведено у додатку.

4.3 Висновки по розділу 4

1. Запропоновано алгоритм спрощення дробово-раціональних функцій шляхом пониження їхніх степенів із використанням ланцюгових дробів. Застосування цього алгоритму надає можливість запобігти надмірності степенів рядів, за допомогою яких подаються розв'язання лінійних крайових задач математичної фізики.

2. Запропоновано алгоритм апроксимації циліндричних функцій дробово-раціональними виразами, що надає можливість автоматизувати обчислення інтегралів від довільного добутку циліндричних функцій, зокрема функцій Бесселя першого та другого типів.

3. Розроблено алгоритм перетворення конвективної складової рівнянь Нав'є –Стокса.

4. Реалізація наведених алгоритмів дозволяє створити алгоритмічне і

програмне забезпечення і, отже, автоматизувати процеси розв'язання нелінійних крайових задач.

Розділ 5. МОДЕЛЮВАННЯ ТЕМПЕРАТУРНОГО ПОЛЯ КОРПУСУ

5.1 Механізм індукційного нагріву [36]

Визначимо явний аналітичний вираз для густини внутрішніх джерел w нагріву. Розглянемо спрощений варіант вирішення цієї задачі, який полягає у тому, щоб наближено описати функцію густини внутрішніх джерел нагріву без розв'язання відповідної крайової задачі за певних спрощуючих припущень про взаємозв'язок електромагнітних процесів в індукторі і корпусі.

Будемо припускати, що потужність потоку пропорційна квадрату напруженості магнітного поля. При змінюванні току в індукторі напруженість поля змінюється одразу в усій поверхні корпусу. Відповідно, в усіх точках поверхні корпусу, що нагрівається, змінюється густина індукованих токів, тобто змінювання густини джерел тепла здійснюється всюди одночасно. Тоді функцію густини теплових джерел можна подати як

$$w(x,y,z,t) = q_0(t)w_0(x,y,z). \tag{5.1}$$

Тут $q_0(t)$ – питома поверхнева потужність.

Знайдемо залежність густини джерел тепла $w_0(x,y,z)$ від напруженості магнітного поля $H(x,y,z)$. Розглянемо плоску електромагнітну хвилю, яка поляризована у площині zOy, що виправдано, оскільки геометричні розміри корпусу, що нагрівається, такі, що його довжина перевищує ширину майже на порядок, а напруга змінюється за синусоїдальним законом.

У відповідності із рівняннями Максвела, із урахуванням наведених припущень, а також за умови сталості магнітної і діелектричної проникності рівняння електродинаміки набувають вигляду

$$-\frac{\partial H_z}{\partial x} = \gamma T_y + \frac{\partial D}{\partial t}; \tag{5.2}$$

$$\frac{\partial H_y}{\partial x} = -\frac{\partial B_z}{\partial t}. \tag{5.3}$$

Для синусоїдальної напруги

$$|H| = |H_{me}| e^{i(\theta_H + \omega t)}; \quad |E| = |E_{me}| e^{i(\theta_E + \omega t)}. \tag{5.4}$$

Звідси

$$\frac{d^2 |H_m|}{dx^2} = i\omega\mu_0\mu\gamma |H_m| - \omega^2\varepsilon_0\varepsilon\mu_0\mu |H_m|. \tag{5.5}$$

Позначимо $k^2 = \omega\mu_0\mu\gamma$. Тоді маємо

$$|H_m| = H_{me}e^{-xk} = H_{me}e^{-x/\Delta}. \tag{5.6}$$

Напруженість магнітного поля дорівнює

$$|E_m| = \rho J_m = E_{me}e^{-x/\Delta} = \sqrt{2}\frac{\rho}{\Delta}H_{me}e^{-x/\Delta}H_{me}e^{-xk} = H_{me}e^{-x/\Delta}, \tag{5.7}$$

де $\Delta = \frac{1}{k} = \sqrt{\frac{2\rho}{\omega\mu_0\mu}}$ — глибина проникнення струму у корпус.

Перетворимо цей вираз із урахуванням значення $\mu_0 = 4\pi 10^{-5}$, $\mu = 2\pi f$ у такий:

$$\Delta = \frac{1}{2\pi}\sqrt{\frac{\rho}{f_k \cdot 10^{-2}\mu}} = \frac{1}{20\pi}\sqrt{\frac{\rho'}{10 f_k}}. \tag{5.8}$$

Тут $\rho = \rho' \cdot 10^{-5}$ – питомий електричний опір, f_k – лінійна частота.

Питомий електричний опір змінюється із зростанням температури за законом

$$\rho_T = \rho_{20}[1 + \alpha(T - 20^\circ C)] = \rho_{20}[1 + \alpha(T + 253K)], \tag{5.9}$$

де $\alpha = \Delta\rho/(\rho\Delta T)$ – температурний коефіцієнт опору.

Для сталі $\rho_{20} = 0.55 \cdot 10^{-5}$, $\alpha_{20} = 0.41 \cdot 10^{-2}$.

Питома потужність на поверхні середовища, що проводить, дорівнює

$$p_0 = J_{me}^2 \rho \Delta / 4 = \frac{10^{-4}}{4\pi} \sqrt{10^{-1} f_k \rho'} H_{me}^2. \qquad (5.10)$$

Отже, можна записати

$$w_0 = p_0 e^{(b/2-\Delta)/\Delta} = \frac{10^{-4}}{4\pi} \sqrt{10^{-1} f_k \rho'} H_{me}^2 e^{(x-b/2)/\Delta}. \qquad (5.11)$$

Якщо густина джерела нагріву на поверхні корпусу задана, користуємося першою частиною формули (5.11), якщо відома напруженість магнітного поля на поверхні корпусу, користуємося другою частиною цієї формули.

Питома поверхнева потужність:

$$\Delta P = 10^{-4} H_0^2 \sqrt{\rho \mu f},$$

H_0 в А/см; ρ – в Ом см.

Напруженість магнітного поля H_0 можна подати у вигляді добутку току індуктора I і кількості витків ω_0 на 1 см його висоти:

$$H_0 = \sqrt{2} I \omega_0.$$

Тоді

$$\Delta P = 2 \cdot 10^{-4} (I \omega_0)^2 \sqrt{\rho \mu f}.$$

Частота $f = 5 \cdot 10^4 / x_k^2$, x_k – глибина проникнення.

Стосовно залежності від температури нагріву корпусу питомого електричного опору. Враховуючи явну залежність ρ від T (5.9), можна або апроксимувати цю залежність дробово-раціональною функцією, або на певному інтервалі температур вважати ρ сталим, а потім перераховувати його значення відповідно до формули (5.9).

Розглянемо спочатку процес індукційного нагріву корпусу екструдера до заданої температури. Задамо на верхній межі ($z = h$) умови теплообміну конвекцією і випромінюванням.

5.2 Температурне поле корпусу екструдера при індукційному нагріві

5.2.1 Математична модель нагрівання корпусу

Основним джерелом нагріву корпусу екструдера вважається індукційний нагрів.

Густина внутрішніх джерел тепла є електромагнітна енергія, що виділяється за одиницю часу в одиниці об'єму. У силу наявності поверхневого ефекту розподіл внутрішніх джерел тепла є суттєво неоднорідний і залежить від електрофізичних властивостей завантаження, які змінюються у процесі нагрівання.

При цьому весь процес нагрівання поділено на інтервали, у кожному із яких властивості завантаження приймаються незмінними.

$$L_i(m) = \{0,9;1,45;2,05;2,65\}.$$

На рис. 5.1 наведено графік розподілу питомої об'ємної потужності за довжиною циліндру.

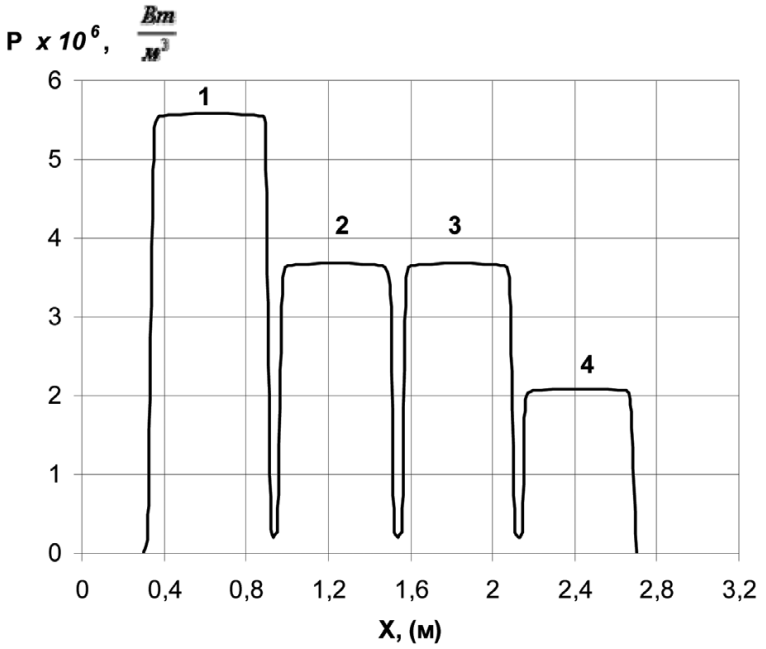


Рис. 5.1 Розподіл питомої об'ємної потужності за довжиною секцій 1—4 [36].

Із графіку видно, що на першій ділянці екструдера (перший індуктор) виділяється максимальна потужність, оскільки на цій ділянці необхідно нагріти полімер від початкової температури до температури плавлення, а на решті зон здійснюється підігрівання і підтримка заданої температури.

Розподіл температурного поля корпусу T_k описується рівнянням теплопровідності, яке у циліндричній системі координат має вигляд

$$\frac{\partial T_k}{\partial t} = a_k\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_k}{\partial r}\right)+\frac{\partial^2 T_k}{\partial z^2}\right)+\frac{a_k}{\lambda}g(r_4,z), \qquad (5.12)$$

де $a_k = \lambda_k/(c_k\rho_k)$ – коефіцієнт температуропровідності; λ_k, c_k, ρ_k – коефіцієнт теплопровідності (Вт/(м$^\circ C$), питома тепломісткість і густина матеріалу корпусу відповідно; $g(r,z)$ – функція розподілу густини внутрішніх джерел енергії у матеріалі, Вт/м^3. Із урахуванням того, що глибина проникнення електромагнітної енергії від індуктора є мала, $\Delta = 1$, будемо вважати, що вона діє на зовнішній поверхні корпусу $r = r_4$. Тоді початкові і межові умови для корпусу:

$$T_k(r,z,t)\,|_{t=0} = T_0; \quad T_k\,|_{r=0} = T_{0k}; \ \frac{\partial T_k}{\partial r}\,|_{r=r_3} = -\frac{q^k}{h_1}; \qquad (5.13)$$

$$\left[\frac{\partial T_k}{\partial z} - h_1 T_k\right]|_{z=0} = T_0, \ \left[\frac{\partial T_k}{\partial z} + h_1 T_k\right]|_{z=L} = 0. \qquad (5.14)$$

де $h_1 = \alpha_k/\lambda_k$; α_k – коефіцієнт тепловіддачі корпусу у навколишнє середовище. Для поверхні контакту сталевої труби із повітрям $\alpha_k = 9$ Вт/(м^2 $^\circ C$).

Межова умова на бічній поверхні циліндричної заготовки:

$$\left[\lambda_1(T_k)\frac{\partial T_k}{\partial r} + \alpha_1(T_k)T_k(r,z,t)\right]|_{r=r_4} = L_G + N_G(T_k); \qquad (5.15)$$

$$L_G = \alpha(T_k)T_0 + \varepsilon_1\left(\frac{T_0}{100}\right)^4; \ N_G = \varepsilon_1\left(\frac{T_k(r_4,z,t)}{100}\right)^4; \ \varepsilon_1 = 0{,}65. \qquad (5.16)$$

Функція $g(r_4, z)$ у даному випадку – це температура, з якою індуктор нагріває поверхню корпусу, тобто $g(r_4, z) = T^{\text{ind}}$.

Значення теплофізичних параметрів для корпусу (сталь) за температури $T = 300K$ дорівнюють: $\rho = 7845$ кг/м^3, $c_V = 0,461$ Квт/(кг$\cdot c \cdot K$), $\lambda = 58$, $\alpha = 10,51/K$.

5.2.2 Розв'язання задачі

Знайдемо температурне поле корпусу екструдера на першій ділянці нагріву (зоні завантаження та пластикації полімеру). Застосуємо до рівняння (5.12) інтегральне перетворення за змінною z.

$$\frac{d^2 Z(\delta_k)}{dz^2} + \delta_k^2 Z(\delta_k) = 0.$$

Власні функції перетворення:

$$Z(\delta_k z) = \frac{1}{\| Z_k \|^2}[\sin \delta_k z - \frac{\delta_k}{h_1} \cos \delta_k z]. \qquad (5.17)$$

Інтеграл від добутку власних функцій дорівнює:

$$\int_{k=1}^{M} \int_{j=1}^{M} Z_k Z_j dz = \sum_{\pm} \frac{\delta_j}{2 p_{kj}} \left[\left(\frac{\delta_k \delta_j}{h_1^2} \mp 1 \right)(1 - \cos p_{kj} L) + \frac{p_{kj}}{h_1} \sin p_{kj} L \right], \; p_{kj} = \delta_k \pm \delta_j;$$

$$\int_{k=1}^{M} \int_{j=1}^{M} Z_k(\delta_k z) \frac{dZ_j(\delta_j z)}{dz} dz = \sum_{\pm} \frac{\delta_j}{2 p_{kj}} \left[\left(1 \mp \frac{\delta_k \delta_j}{h_1^2} \right)(1 - \cos p_{kj} L) - \frac{p_{kj}}{h_1} \sin p_{kj} L \right];$$

зокрема, квадрат норми обчислюється за формулою

$$\| Z \|^2 = \int_0^L Z^2(\delta_k z) dz = \frac{1}{2} \left[\left(\frac{\delta_k^2}{h_1^2} + 1 \right) L + \left(\frac{\delta_k^2}{h_1^2} - 1 \right) \frac{1}{2\delta_k} \sin 2\delta_k L + \frac{1}{h_1}(1 - \cos 2\delta_k L) \right].$$

Для визначення власних значень цієї функції маємо рівняння (задовольняємо другу межову умову (5.14)) :

$$f(\delta) = 2\delta\cos\delta L + (\frac{\delta^2}{h_1} - h_1)\sin\delta L = \frac{2}{L}\sigma\cos\sigma + \left(\frac{\sigma^2}{h_1 L^2} - h_1\right)\sin\sigma = 0,\ \ \sigma = \delta L.$$

Власні значення відшукуються за рівнянням

$$\delta^i = \delta^{i-1} - \frac{f(\delta)}{df(\delta)/d\delta}.$$

$$\frac{df(\delta)}{d\delta} = g(y) = \left(\frac{2}{L} - h_1 + \frac{y^2}{h_1 \cdot L^2}\right)\cos y + \frac{2y}{L}\left(\frac{1}{h_1 \cdot L} - 1\right)\sin y,\ \ y = \delta L.$$

На рис. 5.2 нведено графіки власних функцій (5.17) для перших 6-ти власних значень.

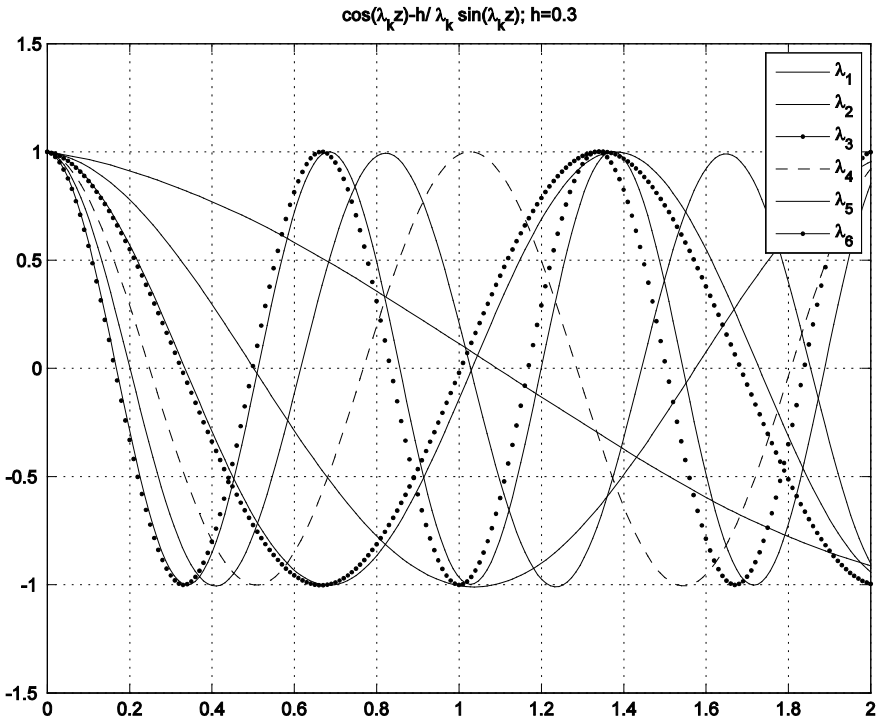


Рис. 5.2 Графіки власних функцій $Z(\lambda_k z)$

Застосування інтегрального перетворення за змінною z до рівняння (5.12) дає:

$$\frac{\partial\overline{T}}{\partial\tau} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\overline{T}}{\partial r}\right) - \delta_k^2\overline{T}_k(r,\delta_k,t) + \frac{a_k}{\lambda}z_k(\delta_k)w + Qz_k; \qquad (5.18)$$

$$Qz_k = \left[h_1 \left(\sin\delta_k L - \frac{\delta_k}{h_1}\cos\delta_k L \right) + (1-h_1)\frac{\delta_k}{h_1} \right] T_{0k}. \qquad (5.19)$$

Межові умови (5.13) набувають вигляду

$$\overline{T}\,|_{r=0} = 0;\ \frac{\partial \overline{T}_k}{\partial r}|_{r=r_3} = -z1_k \frac{q_k}{\lambda_k} = \overline{q}_k; \qquad (5.20)$$

$$\left[\lambda_1(T_k)\frac{\partial \overline{T}_k}{\partial r} + \alpha_1(T_k)_1 \overline{T}_k \right]|_{r=r_4} = z1_k L_G + \overline{L_G},\ z1_k = \int_0^L Z_k(\delta_k z)dz. \qquad (5.21)$$

$$\overline{T}_k(r,t,\delta_k) = \int_0^L Z(\delta_k z)T(r,t,z)dz,\ \ \overline{L_G} = \int_0^L Z_k(\delta_k z)N_G(T_k)dz. \qquad (5.22)$$

Відносно змінної r маємо рівняння Бесселя у межах $r \in [r_3, r_4]$.

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \overline{T}_k(r,\delta_k,t)}{\partial r} \right) + \delta_k^2 \overline{T}_k(r,\delta_k,t) = 0 \qquad (5.23)$$

із однорідними межовими умовами

$$\frac{d\overline{T}_k(r,\delta_k,t)}{dr}|_{r=r_3} = 0;\ \left[\lambda_1(T_k)\frac{d\overline{T}_k(r,\delta_k,t)}{dr} + \alpha_1(T_k)\overline{T}_k(r,\delta_k,t) \right]|_{r=r_4} = 0.$$

Власні функції рівняння (5.23) $R(r)$ можна записати у такому вигляді:

$$R(r) = AJ_0(\beta r) + BI_0(\beta r), \qquad (5.24)$$

де A, B – коефіцієнти, що визначаються із межових умов. β – вектор власних значень задачі. Із урахуванням того, що нагрівання корпусу екструдера (стальної труби) здійснюється до невисоких температур порівняно із температурою плавлення сталі, вважатимемо параметри λ_1 і α_1 незалежними від температури (тобто сталими). Позначимо $h_1 = \alpha_1/\lambda_1$.

Власні функції за змінною r набувають вигляд:

$$R_n(\beta r) = \frac{B_n(\beta_n r)}{\mathrm{P}B_n(\beta_n r)\mathrm{P}^2};\quad R_n(\beta_n r) = [I_0(\beta r) - D_n(\beta r_3)J_0(\beta r)]. \qquad (5.25)$$

Заміщення (5.23) і (5.15) при $r = r_4$ у другу межову умову (5.14) із урахуванням виразу для A призводить до характеристичного рівняння для відшукання власних значень β_n.

$$h_1[J_1(\beta r_3)I_0(\beta r_4) - I_1(\beta r_3)J_0(\beta r_4)] - \beta[J_1(\beta r_3)I_1(\beta r_4) - I_1(\beta r_3)J_1(\beta r_4)] = 0.$$

Позначимо $\gamma = \beta r_4$, $c = r_3/r_4$ і

$$B_m^l(\gamma) = J_m(\gamma c)I_l(\gamma) - I_m(\gamma c)J_l(\gamma);\ l,m = 1,2.$$

Тоді для обчислення власних значень β_n будемо мати такі функції:

$$q(\gamma) = \frac{\gamma}{r_4}B_1^1(\gamma) - h_1 B_1^0(\gamma) = 0,$$

$$w(\gamma) = \frac{c}{b}\gamma B_0^1(\gamma) - \left(\frac{1}{r_4} + h_1\right)B_1^1(\gamma) - \left(\frac{1}{r_4}\gamma + \frac{h_1}{\gamma}\right)B_1^0(\gamma) + h_1 c B_0^0(\gamma).$$

Обчислення власних значень β_n здійснюємо за ітераційною схемою:

$$\gamma_n^{(j)} = \gamma_n^{(j-1)} - \frac{f(\gamma)}{g(\gamma)},\ \gamma_1^0 = 1,\ n = \overline{1,N},\ \varepsilon = 0{,}0001;\ \beta_n = \gamma_n/r_4.$$

На рис. 5.3 наведено графіки власних функцій (5.25) для перших 6-ти власних значень.

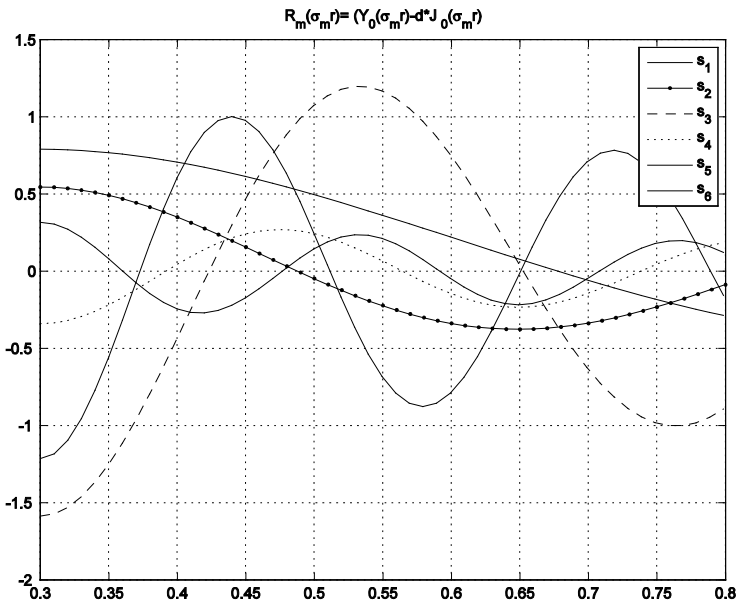


Рис.5.3 Графіки власних функцій $R(\beta_n r)$

Алгоритм розв'язання лінійної крайової задачі, який викладено у цьому розділі, використовується у наступних розділах, тому його викладено докладно, щоб не повторювати у цих розділах. Усі відмінності, що пов'язані із специфікою постановки відповідних крайових задач, враховуються у відповідних розділах.

Перейдемо до пошуку розв'язання нелінійної частини цієї задачі.

Інтегральне перетворення за змінною r до рівняння (5.12) та межових умов (5.13):

$$\frac{\partial \overline{\overline{T}}}{\partial t} + \eta_{nk} \overline{\overline{T}} = \overline{\overline{L}}_G - R_n(r_4)\overline{N}_G. \tag{5.26}$$

$$\overline{\overline{L}}_G = \overline{Z_k(0)T_0} + (R_n(r_4)\overline{L}_G + R_n(r_3)Z_R);\ Z_R = z_k q_k/\lambda_1.$$

У просторі зображень за Лапласом маємо

$$\overline{\overline{\overline{T}}}_k(\eta_{nk}, p) = \frac{\overline{\overline{L}}_G}{p(p+\eta_{nk})} - R_n(\beta_n r_4)\frac{1}{p+\eta_{nk}} \mathsf{L}\{\overline{N}_G[T_k^{(0)}(\beta_n, \delta_k, t)]\}. \tag{5.27}$$

Оскільки у правій частині цього виразу міститься нелінійна функція від T_k (умова Стефана—Больцмана) $N_G = \left[T_k(r_4, z, t)/100\right]^4$, розв'язок рівняння (5.26) будемо шукати за ітераційною схемою. На першій ітерації отримаємо розв'язок рівняння без урахування нелінійної функції N_G.

$$T_k^{(0)}(r, z, t) = \sum_{n=1}^{M}\sum_{k=1}^{M} R_n(\beta_n r) Z_k(\delta_k z) C_{nk}\left(1 - e^{-\eta_{nk} t}\right). \tag{5.28}$$

де $C_{nk} = \dfrac{\overline{\overline{L}}_G}{\eta_{nk}}$, $\eta_{nk} = \dfrac{\lambda_k}{c_v \rho}(\beta_n^2 + \delta_k^2)$.

Замістимо розв'язок (5.28) із урахуванням інтегральних перетворень за змінною z:

$$\overline{N}_G[T_k^{(0)}(\beta_n, \delta_k, t)] = \prod_{m=1}^{4}\sum_{n_m}^{M}\sum_{k_m}^{N} D_{n_m, k_m}\left(1 - e^{-\eta_{n_m, k_m} t}\right); \tag{5.29}$$

$$D_{n_j,k_j} = rn_{n_j} zk_{k_j} C_{n_j,k_j};\ m = [n,k;n_1,k_1;n_2,k_2;n_3,k_3;n_4,k_4]$$

$$rn_{n_j} = R_n(\beta_n r_4)R_{n_1}(\beta_{n_1} r_4)R_{n_2}(\beta_{n_2} r_4)R_{n_3}(\beta_{n_3} r_4)R_{n_4}(\beta_{n_4} r_4);$$

$$zk_{k_j} = \frac{1}{\mathrm{P}Z_k(\delta_k z)\mathrm{P}}\int_{k=1}^{N} Z_k(\beta_k z)Z_{k_1}(\beta_{k_1} z)Z_{k_2}(\beta_{k_2} z)Z_{k_3}(\beta_{k_3} z)Z_{k_4}(\beta_{k_4} z)dz.$$

Оскільки у цьому виразі температура $T_k^{(0)}(r,z,t)$ входить у 4-му степені ми не будемо виписувати цей добуток (обчислення відповідних перетворень реалізується за допомогою відповідної програми мовою C). Випишемо добуток для однієї складової, наприклад, для $n_1 = n_2 = n_3 = n_4 = 1$, $k_1 = k_2 = k_3 = k_4 = 1$. Позначимо $\overline{N}_{1,1,G}$. Тоді

$$\overline{N}_{1,1,G} = \frac{1}{100^4}[D_{1,1}^4\left(1 - e^{-\eta_{11}t}\right)]^4 = \sum_{l=0}^{4} d_l e^{-\sigma_l t}, \qquad (5.30)$$

де

$$d_l = D_{1,1}\cdot(-1)^l \cdot C_l^4;\ \sigma_1 = \eta_{1,1}, \sigma_2 = 2\eta_{1,1},\ \sigma_3 = 3\eta_{1,1},\ \sigma_4 = 4\eta_{1,1}.$$

Решта складових виразу (29) матиме аналогічний вигляд зі зміною індексів 1,1 на i, j.

Тепер застосуємо до (5.30), алгоритм еквівалентного спрощення [142], який перетворює вирази вигляду (5.30) у ланцюг другого порядку у просторі зображень за Лапласом:

$$\overline{\overline{\overline{T}}}_k(\eta_{nk}, p) = \frac{a_2^{n,k}}{p} + \frac{a_0^{n,k} + a_1^{n,k}}{a_3^{n,k} + a_4^{n,k} p + a_5^{n,k} p^2}. \qquad (5.31)$$

Застосування ітераційної процедури обчислення виразів вигляду (5.30) призводить до змінювання значень коефіцієнтів $a_j^{n,k}$, $j = \overline{0,5}$, але структура цього виразу залишається незмінною. Після реалізації ітераційної процедури за досягнення вимог точності (кількість ітерацій) отримаємо розв'язок задачі:

$$T_k(r,z,t)=\sum_{n=1}^{M}R_n(\beta_n r)\sum_{k=1}^{N}Z_k(\beta_k z)\left[\overline{b}_2^{n,k}+e^{-\alpha^{n,k}t}(\overline{b}_0^{n,k}f_1(\omega^{n,k}t)+\overline{b}_1^{n,k}f_2(\omega^{n,k}t))\right]. \quad (5.32)$$

Тут

$$f_1(\omega^{n,k}t)=\begin{cases}\sin(\omega^{n,k}t) & \omega^{n,k}>0,\\ \mathrm{s}\,h(\omega^{n,k}t) & \omega^{n,k}<0,\end{cases}; \quad f_2(\omega^{n,k}t)=\begin{cases}\cos(\omega^{n,k}t) & \omega^{n,k}>0,\\ ch(\omega^{n,k}t) & \omega^{n,k}<0,\end{cases}.$$

Отже, завдяки використанню алгоритмів еквівалентного спрощення нелінійних виразів у рівняннях отримуємо розв'язок нелінійної крайової задачі у класі лінійних функцій.

На рис. 5.4, 5.5 наведено результати моделювання температурного поля корпусу шнека.

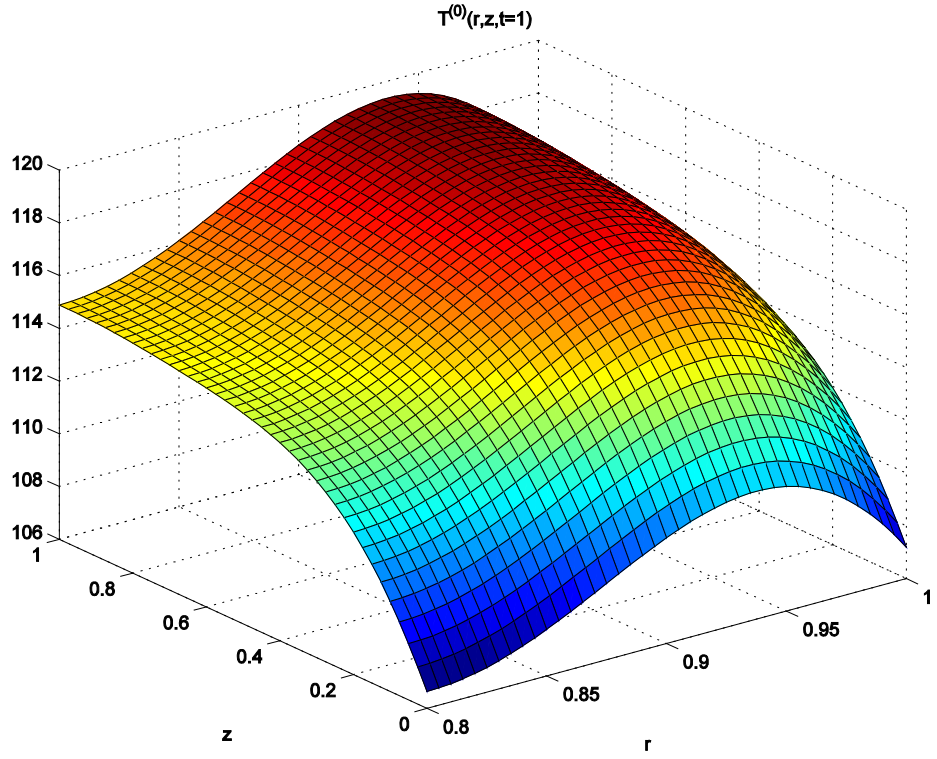


Рис. 5.4 Температурне поле корпусу шнека на 1-й ітерації

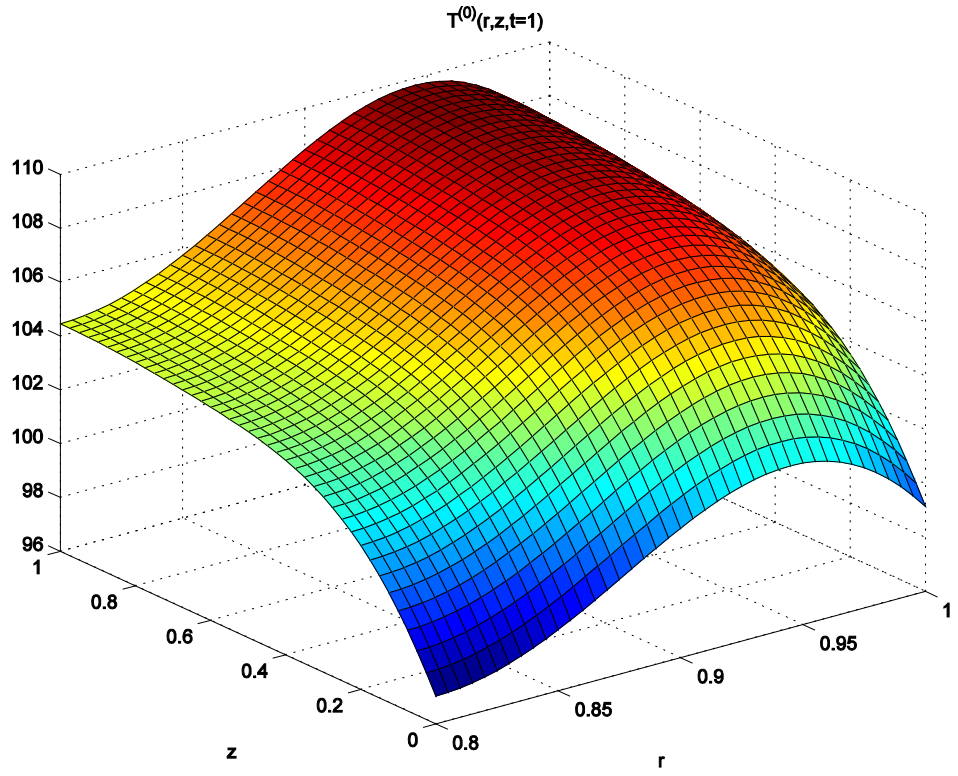


Рис.5.5 Температурне поле корпусу шнека 2-й ітерації

Основна мета математичного моделювання теплоперенесення у корпусі екструдера полягає у визначенні температурного поля на внутрішній поверхні корпусу на межі із шнеком, що визначає умови нагріву полімерної суміші у зоні завантаження. Виходячи з цього, вираз (5.32) набуває вигляду

$$T_k(r_3,z,t)=\sum_{n=1}^{M}R_n(\beta_n r_3)\sum_{k=1}^{N}Z_k(\beta_k z)\left[\overline{b}_2^{n,k}+e^{-\alpha^{n,k}t}(\overline{b}_0^{n,k}f_1(\omega^{n,k}t)+\overline{b}_1^{n,k}f_2(\omega^{n,k}t))\right], \quad (5.33)$$

Підсумовування по n за власними функціями $R_n(\beta_n r_3)$ у просторі зображень за Лапласом) на внутрішній поверхні циліндру дає

$$\tilde{T}_k(r_3,z,p)=\sum_{k=1}^{N}Z_k(\beta_k z)\left[\frac{c_2^k}{p}+\frac{c_0^k+c_1^k p}{c_3^k+c_4^k p+c_5^k p^2}\right]. \quad (5.34)$$

Оригінал цього виразу:

$$\tilde{T}_k(r_3,z,t)=\sum_{k=1}^{N}Z_k(\beta_k z)\left[\overline{c}_2^k+e^{-\xi^k t}(\overline{c}_0^k f_1(\chi^k t)+\overline{c}_1^k f_2(\chi^k t))\right], \quad (5.35)$$

На рис. 5.6, 5.7 наведені температурні поля на внутрішній поверхні корпусу екструдера для різних питомих потужностей індуктора.

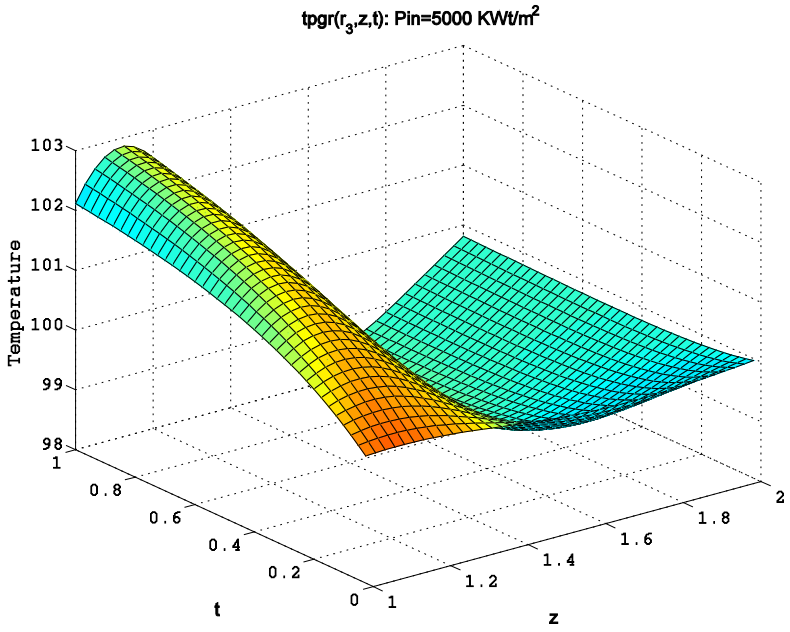


Рис. 5.6 Температурне поле внутрішньої поверхні корпусу

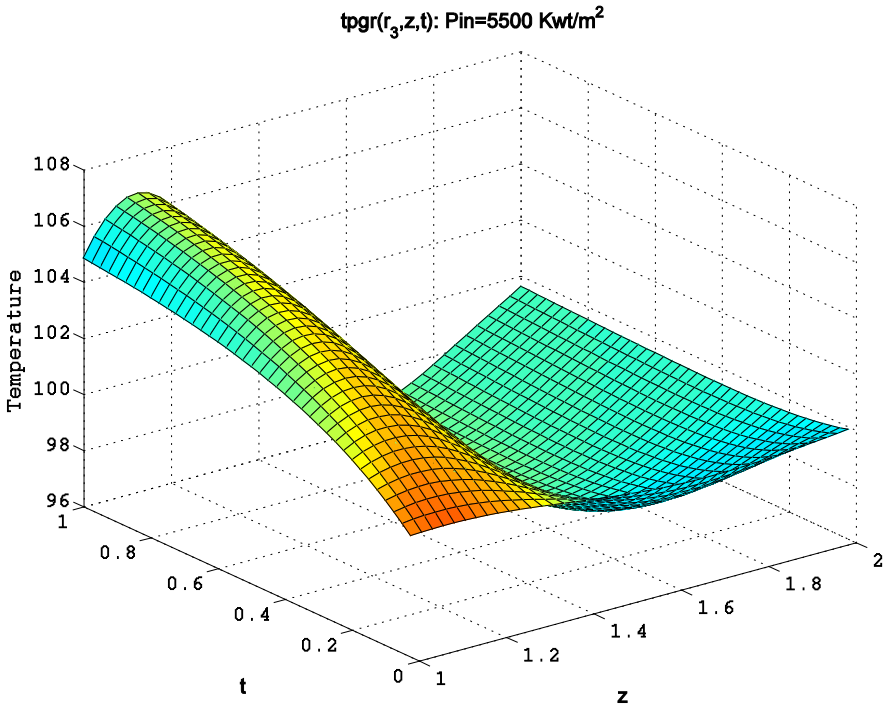


Рис. 5.7 Температурне поле внутрішньої поверхні корпусу

Вираз (35) слугує межовою умовою на зовнішній поверхні шнека крайової задачі теплоперенесення полімерній суміші у зоні її завантаження.

5.2.3 Власні значення відносно змінної z на інтервалі $z \in [L_1, L]$

Цей підрозділ наведено для випадку, коли одна із меж відносно змінної z може змінюватися у процесі дослідження задачі.

Власні функції:

$$Z_k(z) = e(\lambda_k L_1)\sin(\lambda_k z) + \cos(\lambda_k z).$$

$$e(\lambda L_1) = \frac{\lambda \sin(\lambda L_1) + h\cos(\lambda L_1)}{\lambda \cos(\lambda L_1) - h\sin(\lambda L_1)},\ e(\lambda \cdot 0) = \frac{h}{\lambda}.$$

Для відшукання λ_k, $k = 1,2,\ldots$, маємо рівняння

$$f(\lambda) = e(\lambda L_1)(\lambda\cos(\lambda L) - h\sin(\lambda L)) + h\cos(\lambda L) - \lambda\sin(\lambda L) = 0.$$

Згідно з формулою

$$\Delta\lambda_k = -\frac{f(\lambda)}{df(\lambda)/d\lambda}$$

отримаємо вираз для $df/d\lambda$.

$$\frac{df}{d\lambda} = g(\lambda) = \frac{de(\lambda L_1)}{d\lambda}(\lambda\cos(\lambda L) - h\sin(\lambda L))$$

$$+e(\lambda L_1)[(1 + h\cdot L)\cos(\lambda L) + L\lambda\sin(\lambda L)] - L[h\sin(\lambda L) + \lambda\cos(\lambda L)].$$

$$\frac{de(\lambda L_1)}{d\lambda} = 2\frac{h(hL_1 - 1) - hL_1\lambda\sin(2\lambda L_1) - h^2 L_1\cos(2\lambda L_1)}{\lambda^2 + h^2 + (\lambda^2 - h^2)\cos(2\lambda L_1) - \lambda h\sin(2\lambda L_1)},\ \frac{de(\lambda L_1)}{d\lambda}\Big|_{L_1=0} = \frac{-h}{\lambda^2}.$$

Після відшукання власних значень λ_k, $k = \overline{1, M}$, обчислюємо

$$\|Z_k(z)\|^2 = \int_0^L Z_k^2(z)dz = \frac{1}{\lambda_k^2}\left[(\lambda_k^2 + h^2)L - \frac{1}{2\lambda_k}(\lambda_k^2 - h^2)\sin(2\lambda_k^2 L) + h(1 - \cos(\lambda_k^2 L)\right].$$

$$zi_k = \frac{1}{\|Z_k(z)\|^2}\int_0^L Z_k(z)dz = \frac{1}{\|Z_k(z)\|^2}[\sin\lambda_k L/\lambda_k + h(\cos\lambda_k L - 1)].$$

Інтеграли від власних функцій відносно змінної z (для обчислення нелінійних складових розв'язку):

$$zi_{kk_1k_2} = \frac{1}{\|Z_k(z)\|^2}\int_0^L Z(\lambda_k z)Z(\lambda_{k_1} z)Z(\lambda_{k_2} z)dz =$$

$$\frac{1}{\|Z_k(z)\|^2}\left[\frac{1}{4(\lambda_k \pm \lambda_{k_1} \pm \lambda_{k_2})}\left[1 \pm c_{k_1}c_k \mp c_{k_2}(c_k \pm c_{k_1})\right]\sin(\lambda_k \pm \lambda_{k_1} \pm \lambda_{k_2})L\right.$$

$$\left.-\left[1 + \mp c_{k_1}c_k)c_{k_2} + (c_k \mp c_{k_1})\right]\cos(\lambda_k \pm \lambda_{k_1} \pm \lambda_{k_2})L\right].$$

5.3.1 Висновки по розділу 5

1. Сформульовано задачу про індукційне нагрівання корпусу екструдера яка описується лінійним рівнянням теплопровідності із нелінійним межовими умовами на межі індуктор – зовнішня поверхня корпусу.

2. Отримано розв'язок цієї крайової задачі із застосуванням скінченних інтегральних перетворень за просторовими змінними та інтегрального перетворення Лапласа за часовою змінною.

3. Показана доцільність урахування променистого випромінювання на визначення оптимального температурного поля корпусу екструдера.

4. Показано вплив змінювання питомої потужності індуктора на температуру зовнішньої поверхні корпусу екструдера.

Розділ 6. МОДЕЛЮВАННЯ ТЕМПЕРАТУРНОГО ПОЛЯ ПОЛІМЕРУ У ЗОНІ ЗАВАНТАЖЕННЯ

6.1 Постановка задачі

При формулюванні крайової задачі процесу нагрівання сухої поліетиленової суміші на ділянці завантаження необхідно враховувати наявність обертального руху шнеку зі сталою швидкістю $V_{øi}$ із нарізкою під кутом φ. Рівняння теплоперенесення набуває вигляду:

$$\rho_i c_{v_i}\left(V_{øi}^{r}\frac{\partial T}{\partial r}+V_{øi}^{z}\frac{\partial T}{\partial z}+\frac{\partial T_i}{\partial t}\right)=\lambda_i\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_i}{\partial r}\right)+\frac{\partial^2 T_i}{\partial z^2}\right). \qquad (6.1)$$

Для поліетилену c_{v_i} суттєво залежить від температури нагрівання [ ]. На рис. 6.1 наведено графік змінювання c_{v_i} залежно від температури. Коефіцієнт тепломісткості c_{v_i} апроксимується залежністю

$$c_{v_i}=c_0+c_1T+c_2T^2\approx 2{,}5-0{,}024\cdot T+5{,}7\cdot 10^{-4}\cdot T^2. \qquad (6.2)$$

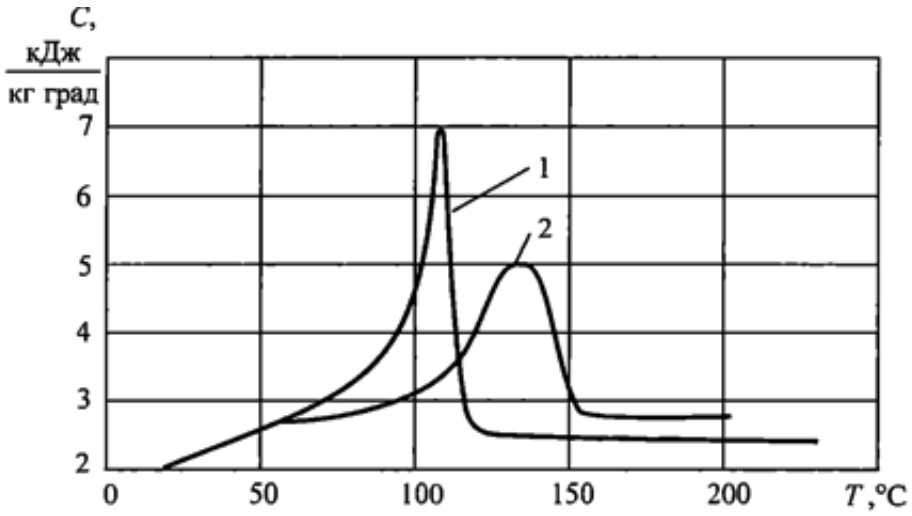


Рис. 6.1 Графік залежності C_{v_i} від температури полімеру [127]

На межі корпусу і шнеку має виконуватися умова рівності теплових потоків

$$\lambda_k \frac{\partial T_k}{\partial r}|_{r=r_3} = \lambda_{sn} \frac{\partial T_{sn}}{\partial r}|_{r=r_3}. \qquad (6.3)$$

6.2 Розв'язання задачі

Із урахуванням (6.3) маємо межову умову при $r = r_3$ для температури шнеку:

$$T_k(r,z,t)|_{r=r_3} = \sum_{n=1}^{M}\sum_{k=1}^{M} \frac{1}{\mathrm{P}R_n\,\mathrm{P}^2}[I_1(\beta_n r) - D_n J_1(\beta_n r)]|_{r=r_3}\, Z_k(\delta_k z)\frac{G_{nk}}{\eta_{nk}}\left(1 - e^{-\eta_{nk}t}\right). \qquad (6.4)$$

$$T_{\text{øí}}|_{z=0} = T_{sn}^0, \qquad (6.5)$$

$$\left[\frac{\partial T_{\text{øí}}}{\partial z} + h_2 T_{\text{øí}}\right]|_{z=L_1} = 0. \qquad (6.6)$$

Температурне поле полімерної суміші у зоні завантаження описується крайовою задачею (6.1)–(6.6). Із урахуванням залежності коефіцієнту тепломісткості полімеру від температури c_{vp} (6.2) рівняння теплопровідності для полімеру у зоні завантаження можна записати у вигляді:

$$\frac{\partial T_{\text{п}}}{\partial t} = a_{\text{п}}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_{\text{п}}}{\partial r}\right) + \frac{\partial^2 T_{\text{п}}}{\partial z^2}\right) + NT_{\text{п}}(r,z,t), \qquad (6.7)$$

де $a_{\text{п}} = \lambda_{\text{п}}/(\rho c_{01})$.

$$NT_{\text{п}}(r,z,t) = [c'_{11}T_{\text{п}} + c'_{21}T_{\text{п}}^2]\frac{\partial T_{\text{п}}}{\partial t}, \qquad (6.8)$$

де $c'_{11} = -c_{11}/c_{01}$, $c'_{21} = -c_{21}/c_{01}$. Оскільки рівняння (6.7) – нелінійне, будемо шукати його розв'язок за ітераційною схемою. Розв'язок лінійної частини рівняння

$$\frac{\partial T_{\text{п}}^{(0)}}{\partial t} = a_{\text{п}0}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_{\text{п}}^{(0)}}{\partial r}\right) + \frac{\partial^2 T_{\text{п}}^{(0)}}{\partial z^2}\right] \qquad (6.9)$$

$a_{п0} = \dfrac{\lambda_п}{\rho c_{01}}$ із межовими умовами (6.7), (6.6) має вигляд, аналогічний (5.28), відрізняється теплофізичними параметрами і межовими умовами.

Різниця полягає в умові на межі $r = r_3$, що дає вираз для температурного поля на цій межі (6.4). Під час застосування інтегральних перетворень за просторовими змінними до крайової задачі (6.7)–(6.9) із урахуванням (6.4) матимемо для цієї умови:

$$\overline{T^k}(r_3, \gamma_k, t) = \sum_{n_1=1}^{M} \sum_{k=1}^{M} \sum_{k_1=1}^{M} \frac{1}{\left\| R_{n_1} \right\|^2} \frac{\partial R_{n_1}}{\partial r} \overline{Z}_{k,k_1} \frac{G_{n_1,k_1}}{\eta_{n_1,k_1}} \left(1 - e^{-\eta_{n_1,k_1} t}\right). \tag{6.10}$$

$$UG_{n,n_1,k,k_1} = R^п(r_3)[I_1(\beta_{n_1} r_3) - D_{n_1}(r_2) J_1(\beta_{n_1} r_3)] \overline{Z}_{k,k_1} G_{n_1,k_1} \left(1 - e^{-\eta_{n_1,k_1} t}\right), \tag{6.11}$$

$$R^п(r_3) = [I_0(\beta_n r_3) - D_n(r_2) J_0(\beta_n r_3)].$$

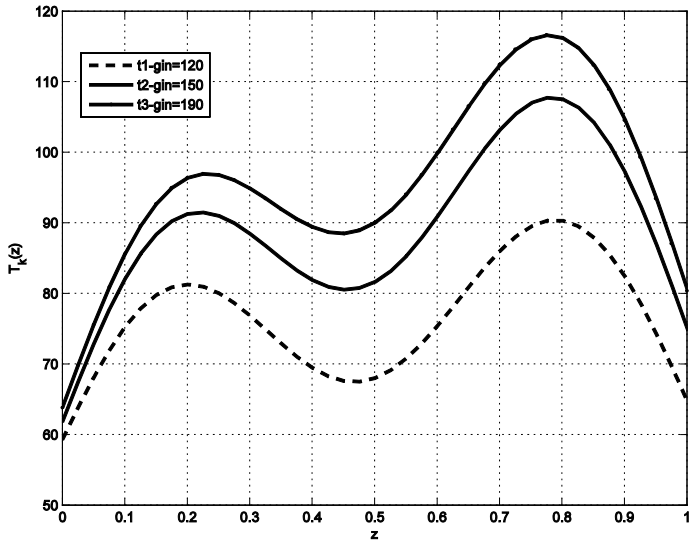


Рис. 6. Залежність температури на межі корпус – шнек від параметрів

На рис. 6. наведено залежність температури на межі корпус – шнек від параметрів. Використання цього виразу в якості межової умови при розв'язанні задачі теплоперенесення (6.7), (6.5), (6.6) та (6.10) призводить до оперування із $M \times N \times N$ складовими вигляду $UG_{n_1,k,k_1}\left(1 - e^{-\eta_{n_1,k_1} t}\right)$, що практично

унеможливлює їхнє подальше використання. Тому виконаємо спрощення цього виразу.

$$\overline{T^k}(r_3,\gamma_k,t) \approx \sum_{n_1}^{M}\sum_{k}^{N}\sum_{k_1}^{N} UG_{n_1,k,k_1}\left(1-e^{-\eta_{n_1,k_1}t}\right)$$

$$= Gr_{n,k}(t) = \sum_{n=1}^{N}\sum_{k=1}^{M}\left[gr_0^{n,k} + e^{-\alpha^{n,k}t}(gr_1^{n,k}\sin(\omega^{n,k}t) + gr_2^{n,k}\cos(\omega^{n,k}t))\right]. \qquad (6.12)$$

Алгоритм, за яким виконується таке спрощення, викладено у розділі 4. Похибка, що природно виникає за такого спрощення, компенсується за рахунок ітераційного процесу при вирішення нелінійної задачі.

Запишемо рівняння теплопереносу у шнеці (температурне поле суміші полімеру) після застосування перетворень за просторовими змінними z,r:

$$\overline{\overline{T0}}_{n,k}(t) + \eta_{n,k}\overline{\overline{T}}_{n,k}(t) = gz_{n,k} + Gr_{n,k}(t). \qquad (6.13)$$

У просторі зображень за часовою координатою t матимемо

$$\overline{\overline{\overline{T0}}}_{n,k}(p) = \frac{1}{p+\eta_{n,k}}\left[\frac{gz_{n,k}}{p} + \frac{(gr_0^{n,k}\omega^{n,k} - gr_1^{n,k}\alpha^{n,k}) + gr_1^{n,k}(p+\alpha^{n,k})}{(p+\alpha^{n,k})^2 + (\omega^{n,k})^2}\right]$$

$$= \frac{gz_{n,k}}{\eta_{n,k}}\frac{1}{p} + \frac{t0_0^{n,k} + t0_1^{n,k}p}{t0_3^{n,k} + t0_4^{n,k}p + p^2}.$$

У просторі оригіналів маємо

$$T^{(0)}(r,z,t) = \sum_{n=1}^{N}\sum_{k=1}^{M} R_n(r)Z_k(z)T0(t). \qquad (6.14)$$

$$T0(t) = gz1^{n,k} + e^{-\alpha^{n,k}t}\left[t1_0^{n,k}\sin(\omega^{n,k}t) + t1_1^{n,k}\cos(\omega^{n,k}t)\right]. \qquad (6.15)$$

У цьому виразі враховано наближену рівність

$$\frac{1}{p+\eta^{n,k}}\frac{t0_0^{n,k} + t0_1^{n,k}p}{t0_3^{n,k} + t0_4^{n,k}p + p^2} \text{ ґ } \quad e^{-\alpha^{n,k}t}\left[t1_0^{n,k}\sin(\omega^{n,k}t) + t1_1^{n,k}\cos(\omega^{n,k}t)\right]. \quad (6.16)$$

Після отримання розв'язку рівняння у вигляді (6.14) із урахуванням (6.15) розв'язок рівняння (7) шукатимемо за ітераційною схемою. Нагадаємо, що

$$NT_{\text{ї}}^{(1)}(r,z,t)=\left[c'_{11}T_{\text{ї}}^{(0)}+c'_{21}(TT^{(0)})_{\text{ї}}^{2}\right]\frac{\partial T_{\text{ї}}}{\partial t}, \tag{6.17}$$

Загальну схему розв'язання нелінійного рівняння запишемо у вигляді

$$T_{\text{п}}^{(m)}(r,z,t)=T_{\text{п}}^{(0)}(r,z,t)+NT_{\text{п}}^{(m-1)}(r,z,t),\ m=1,2,\ldots \tag{6.18}$$

Замістимо вираз (6.14) у (6.8).

$$T_{\text{ї}}^{(0)}\frac{\partial T_{\text{ї}}^{(0)}}{\partial t}=\sum_{n_1=1}^{M}\sum_{k_1=1}^{M}R_{n_1}^{p}(\beta_{n_1}^{p}r)Z_{k_1}((\delta^{p})_{k_1}z)\frac{G_{n_1k_1}^{p}}{\eta_{n_1k_1}^{p}}\left(1-e^{-\eta_{n_1k_1}^{p}t}\right)$$

$$\times\sum_{n_2=1}^{M}\sum_{k_2=1}^{M}R_{n_2}^{p}(\beta_{n_2}^{p}r)Z_{k_2}(\delta_{k_2}^{p}z)\frac{G_{n_2k_2}^{p}}{(\eta^{p})_{n_2k_2}^{2}}\left(\eta_{n_2k_2}^{p}e^{-\eta_{n_2k_2}^{p}t}\right); \tag{6.19}$$

$$\eta_{nk}^{p}=a_k[(\beta^{p})_n^2+(\delta^{p})_k^2]=a_k\gamma_{nk}^2,\ a_k=\lambda^{p}/(\rho^{p}c_{01}).$$

β^{p}, δ^{p} – власні значення для власних функцій за r та z-змінними.

$$(T^{(0)})_{\text{п}}^{2}\frac{\partial T_{\text{ї}}^{(0)}}{\partial t}=\sum_{n_1=1}^{M}\sum_{k_1=1}^{M}R_{n_1}^{p}(\beta_{n_1}^{p}r)Z_{k_1}((\delta^{p})_{k_1}z)\frac{G_{n_1k_1}^{p}}{\eta_{n_1k_1}^{p}}\left(1-e^{-\eta_{n_1k_1}^{p}t}\right)$$

$$\times\sum_{n_2=1}^{M}\sum_{k_2=1}^{M}R_{n_2}^{p}(\beta_{n_2}^{p}r)Z_{k_2}((\delta^{p})_{k_2}z)\frac{G_{n_2k_2}^{p}}{\eta_{n_2k_2}^{p}}\left(1-e^{-\eta_{n_2k_2}^{p}t}\right)$$

$$\times\sum_{n_3=1}^{M}\sum_{k_3=1}^{M}R_{n_2}^{p}(\beta_{n_3}^{p}r)Z_{k_3}(\delta_{k_3}^{p}z)\frac{G_{n_3k_3}^{p}}{\eta_{n_3k_3}^{p}}\left(\eta_{n_3k_3}^{p}e^{-\eta_{n_3k_3}^{p}t}\right). \tag{6.20}$$

Застосування інтегральних перетворень до (6.7) за просторовими змінними призводить до такого рівняння:

$$\overline{\overline{\overline{T^{(1)}}}}_{nk}(p)=\overline{\overline{\overline{T^{(0)}}}}_{nk}(p)+\overline{\overline{\overline{NT^{(0)}}}}_{\text{ї}}(r,z,t). \tag{6.21}$$

Далі, маючи на увазі температурне поле суміші полімеру, для спрощення викладок індекс p опустимо. Інтегральне перетворення за змінними z і r

(після заміщення (6.19), (6.20) у (6.8)) призводять до обчислення інтегралів:

$$nz_{kk_1k_2} = \frac{1}{\|Z_k\|^2}\int_{L_0}^{L_1} Z(\delta_k z)Z(\delta_{k_1} z)Z(\delta_{k_2} z)dz;$$

$$nz_{kk_1k_2k_3} = \frac{1}{\|Z_k\|^2}\int_{L_0}^{L_1} Z(\delta_k z)Z(\delta_{k_1} z)Z(\delta_{k_2} z)Z(\delta_{k_3} z)dz;$$

$$nr_{nn_1n_2} = \frac{1}{\|R(\beta_n r)\|^2}\int_{r_2}^{r_3} R(\beta_n r)R(\beta_{n_1} r)R(\beta_{n_2} r)rdr;$$

$$nr_{nn_1n_2n_3} = \frac{1}{\|R(\beta_n r)\|^2}\int_{r_2}^{r_3} R(\beta_n r)R(\beta_{n_1} r)R(\beta_{n_2} r)R(\beta_{n_3} r)rdr.$$

Отже, після застосування інтегральних перетворень за просторовими змінними до нелінійної частини рівняння (6.21) отримаємо:

$$\overline{\overline{\overline{T^{(1)}}}}_{nk}(p) = \overline{\overline{\overline{T^{(0)}}}}_{nk}(p) + \mathsf{L}_t\{c_{11}nr_{n,n_1n_2}nz_{k,k_1k_2}G_{n_1k_1}G_{n_2k_2}\left(1-e^{-\eta_{n_1k_1}t}\right)\eta_{n_2k_2}e^{-\eta_{n_2k_2}t}$$

$$+c_{21}nr_{n,n_1n_2n_3}nz_{k,k_1k_2k_3}G_{n_1k_1}G_{n_2k_2}G_{n_3k_3}\left(1-e^{-\eta_{n_1k_1}t}\right)\left(1-e^{-\eta_{n_2k_2}t}\right)\eta_{n_3k_3}e^{-\eta_{n_3k_3}t}\}.$$

(6.22)

Для наочності запровадимо позначення:

$$d^1 = c_{11}nr_{n,n_1n_2}nz_{k,k_1k_2}G_{n_1k_1}G_{n_2k_2}\eta_{n_2k_2};$$

$$d^2 = c_{21}nr_{n,n_1n_2n_3}nz_{k,k_1k_2k_3}G_{n_1k_1}G_{n_2k_2}G_{n_3k_3}\eta_{n_3k_3};$$

$$s^1 = \eta_{n_2k_2};\ s^2 = \eta_{n_1k_1} + \eta_{n_2k_2};\ s^3 = \eta_{n_1k_1} + \eta_{n_3k_3};\ s^4 = \eta_{n_2k_2} + \eta_{n_3k_3};$$

$$s^5 = \eta_{n_1k_1} + \eta_{n_2k_2} + \eta_{n_3k_3}.$$

$$\overline{\overline{\overline{T^{(1)}}}}_{nk}(p) = \overline{\overline{\overline{T^{(0)}}}}_{nk}(p) + \mathsf{L}_t\{ \sum_{\substack{n_1=1 \\ n_2=1}}^{N} \sum_{\substack{k_1=1 \\ k_2=1}}^{M} \left[d^1 \left(e^{-s^1 t} - e^{-s^2 t} \right) \right]$$

$$+ \sum_{n_1, n_2, n_3 = 1}^{N} \sum_{k_1, k_2, k_3 = 1}^{M} \left[d^2 \left(e^{-s^3 t} - e^{-s^4 t} + e^{-s^5 t} \right) \right] \} \qquad (6.23)$$

Нагадаємо, що у силу запроваджених позначень коефіцієнти d_1, d_2, s_i, $i = \overline{1,5}$, залежать від власних значень за просторових змінних. Застосуємо до рівняння перетворення Лапласа за часом (початкова умова дорівнює нулю). Маємо:

$$\overline{\overline{\overline{T^{(1)}}}}_{nk}(p) = \overline{\overline{\overline{T^{(0)}}}}_{nk}(p) + \frac{1}{p + \eta_{nk}} \left[d^1 \left(\frac{1}{p + s^1} - \frac{1}{p + s^2} \right) + d^2 \left(\frac{1}{p + s^3} - \frac{1}{p + s^4} + \frac{1}{p + s^5} \right) \right]$$

(6.24)

У виразі у квадратних дужках рівняння (6.24) опущені (для його наочності) суми за індексами n_1, n_2, n_3 та k_1, k_2, k_3. Для однієї ``точки" зворотне перетворення за Лапласом у (6.24) дає вираз вигляду:

$$\overline{\overline{T^{(1)}}}_{nk}(t) = \overline{\overline{T^{(0)}}}_{nk}(t) + E_0 e^{-\eta_{nk} t} + \sum_{i=1}^{5} E_i e^{-s^i t}, \qquad (6.25)$$

Підсумовування за вказаними індексами правої частини рівняння (6.24) призведе до такого виразу відносно оператора p, що унеможливлює його практичну реалізацію засобами обчислювальної техніки. Для розв'язку у другому наближенні ми маємо заміщувати у (6.22) призведе до добутку виразів вигляду (6.24) тощо, тобто до збільшення складових відносно елементарних ланцюгів у геометричній прогресії.

Це спонукає до розробки алгоритмів, що надають можливість запобігти цьому зростанню. Похибки, що при цьому накопичуються, можна компенсувати додатковими ітераціями у відповідному алгоритмі.

Отже, повернемося до рівняння (6.24) і розглянемо вираз у квадратних дужках із урахуванням підсумовування за індексом k_2.

$$\sum_{k_2}^{M} d_{k_2}^{1}\left(\frac{1}{p+s_{k_2}^{1}}-\frac{1}{p+s_{k_2}^{2}}\right)+\sum_{k_2}^{M}\sum_{k_3}^{M} d_{k_2k_3}^{2}\left(\frac{1}{p+s_{k_2k_3}^{3}}-\frac{1}{p+s_{k_2k_3}^{4}}+\frac{1}{p+s_{k_2k_3}^{5}}\right)$$

$$\approx \frac{b_3^{k_1}+b_4^{k_1}p}{b_0^{k_1}+b_1^{k_1}p+b_2^{k_1}p^2}.$$

Своєю чергою, вираз, що отримано, можна за тією самою схемою апроксимувати виразом

$$\sum_{k_1=1}^{M}\frac{b_3^{k_1}+b_4^{k_1}p}{b_0^{k_1}+b_1^{k_1}p+b_2^{k_1}p^2}\approx\frac{c_3^{n_2}+c_4^{n_2}p}{c_0^{n_2}+c_1^{n_2}p+c_2^{n_2}p^2}.$$

Якщо врахувати, що власні значення η зростають досить швидко зі збільшенням N і M, можна обмежитися $N=M=4$. Алгоритм такого спрощення дробово-раціональними виразами наведено у розділі , внаслідок якого отримуються вирази

$$D_{nk}^{(0)}(p)=\sum_{nk}^{M}\frac{d_3^{nk}+d_4^{nk}p}{d_0^{nk}+d_1^{nk}p+d_2^{nk}p^2}.$$

Цьому виразу відповідає оригінал (у просторі інтегральних перетворень за просторовими змінними)

$$D_{nk}^{(0)}(t)=\sum_{n,k=1}^{M}e^{-\alpha_{nk}t}\left(\begin{cases} e1_{nk}\sin\omega_{nk}t+e2_{nk}\cos\omega_{nk}t, & \omega_{nk}<0,\\ e1_{nk}sh\omega_{nk}t+e2_{nk}ch\omega_{nk}t, & \omega_{nk}>0\end{cases}\right). \qquad (6.26)$$

Тоді розв'язок у першому наближенні набуде вигляду

$$T^{(1)}(r,z,t)=\sum_{n=1}^{M}\sum_{k=1}^{M}R(\delta_n r)Z(\beta_k z)\left[\frac{G_{nk}}{\eta_{nk}}(1-e^{-\eta_{nk}t})+D_{nk}^{(0)}(t)\right]. \qquad (6.27)$$

Подальші наближення реалізуються за аналогічною схемою:

$$T^{(m)}(r,z,t)=\sum_{n=1}^{M}\sum_{k=1}^{M}R(\delta_n r)Z(\beta_k z)\left[\frac{G_{nk}}{\eta_{nk}}(1-e^{-\eta_{nk}t})+D_{nk}^{(m-1)}(t)\right]. \qquad (6.28)$$

На рис. 6.2—6.4 наведено температурні поля у зоні завантаження для перших трьох ітерацій.

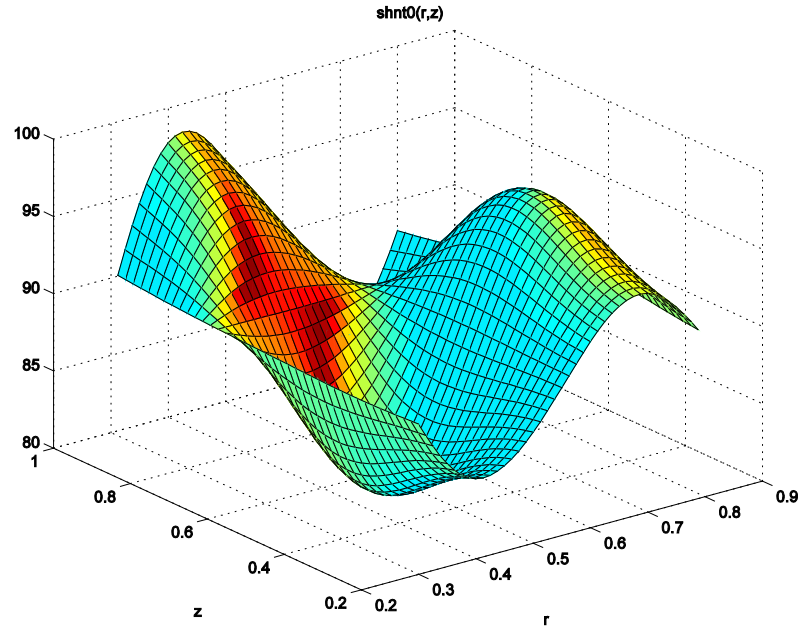


Рис. 6.2 Температурне поле суміші у нульовому наближенні

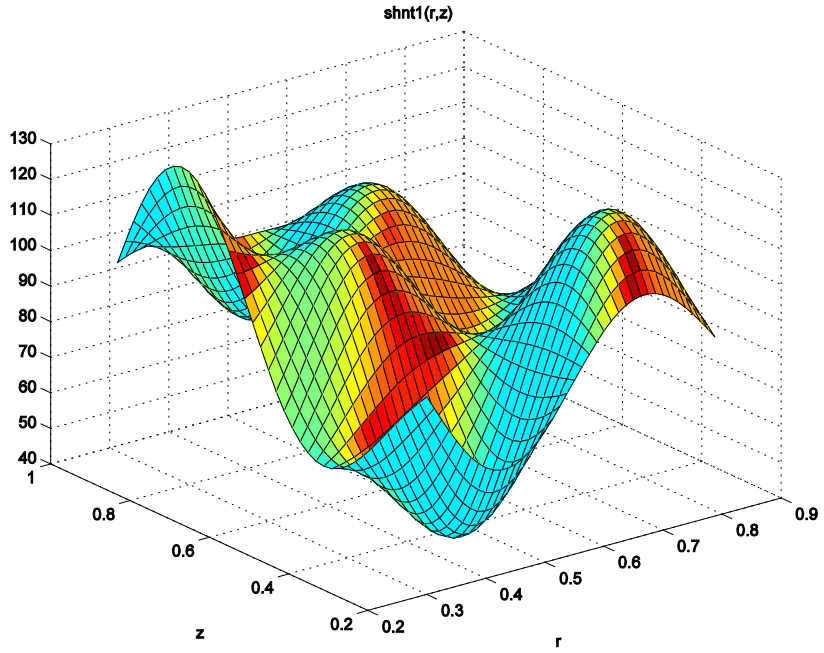


Рис. 6.3 Температурне поле суміші у першому наближенні

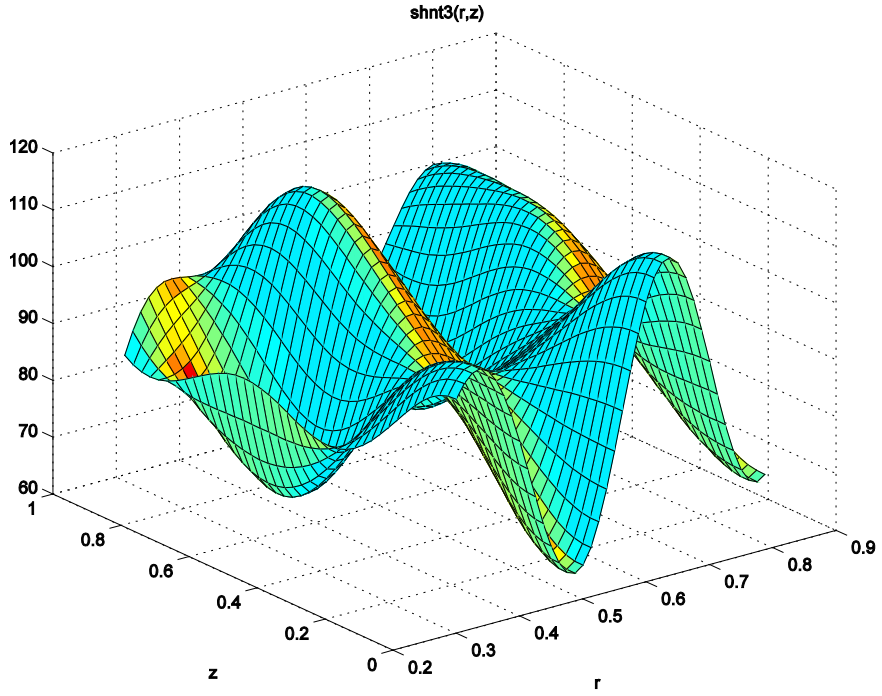


Рис. 6.4 Температурне поле суміші у третьому наближенні

6.2.1 Оцінка похибки ітерацій

Інтегральна оцінка похибки:

$$\Delta^{(m)} = \int_{r_a}^{r_b}\int_{z_a}^{z_b}\int_{T_0}^{T_k}\left[\sum_{n,k} R_n(r)Z_k(z)D_{n,k}^{(m)}(t) - \sum_{n,k} R_n(r)Z_k(z)D_{n,k}^{(m-1)}(t)\right]^2 drdzdt$$

$$= \int_{T_0}^{T_k}\sum_{n,k}[D_{n,k}^{(m)}(t)D_{n,k}^{(m)}(t) - 2D_{n,k}^{(m)}(t)D_{n,k}^{(m-1)}(t) + D_{n,k}^{(m-1)}(t)D_{n,k}^{(m-1)}(t)]dt \qquad (6.29)$$

6.3 Процеси у зоні затримки

Після нагріву поліетиленової суміші у зоні завантаження до температури

плавлення у зоні затримки формується примежовий шар розплаву полімеру.

У цій зоні здійснюється перехід суміші від твердого стану до рідинного у примежовій області з утворенням тонкої плівки розплавленого полімеру.

Ця задача математично описується двохфазними рівняннями теплопровідності у твердій і рідинній фазах, що відома як двохфазна задача типу Стефана. Особливість цієї задачі полягає у стрибкоподібному змінюванні коефіцієнту тепломісткості на межі переходу суміші з твердого стану до рідинного.

Розв'язанню задач типу Стефана присвячена безліч праць. Всі вони грунтуються на використанні різницевих схем математичної фізики і, як правило, обмежуються розв'язанням одновимірних відносно просторових змінних випадком.

Розглянемо задачу переходу суміші із твердого стану у рідинний із утворенням тонкої плівки розплаву у примежовій зоні шнек -- корпус.

Постановка задачі про фазовий перехід тверде тіло – розплав сформульовано у розділі 3.6.

Суттєва відмінність задачі стосовно плавлення полімерних матеріалів в екструдерах полягає у тому, що полімерна суміш є рухома, отже межа фазового переходу має два ступеня свободи -- рух межі відносно радіальної складової і рух ``пробки" відносно аксіальної складової.

6.3.1 Алгоритм побудови ітераційної процедури

Сформулюємо задачу про рух розплаву полімеру у примежовій області шнек – корпус шнека із урахуванням конвективної складової. Оскільки плівка розплаву є тонка, досить враховувати розподіл швидкості руху рідини вздовж осьової координати z. Тоді матимемо:

$$\rho\left(\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial z}+\mu\frac{\partial^2 v}{\partial z^2} \tag{6.30}$$

$$c_{ef}=c+\frac{1}{\rho}L\delta(T-T^*).$$

$$[\rho c0_V+L(T-T^*)]\left(\frac{\partial T}{\partial t}+v\text{grad}T\right)=\text{div}(k\text{grad}T)+q_V. \tag{6.31}$$

Початкові умови:

$$v(z,0)=V_z(z),\ T(z,0)=T_p(z,0). \tag{6.32}$$

Межові умови:

$$v(t)_{z=z1}=V_z(t),\ T(t)_{z=z1}=T_p(t),\ T(r=r_3,z,t)=T_{\text{ml},z,t}. \tag{6.33}$$

Ітераційна процедура розв'язання наведеної системи рівнянь полягає у такому.

1. Спочатку отримуємо розв'язок лінеаризованого рівняння

$$\rho\frac{\partial v^{(0)}}{\partial t}=\nu\frac{\partial^2 v^{(0)}}{\partial z^2}. \tag{6.34}$$

$\nu=\mu/\rho$. Початкова умова $v(z)_{t=0}=v_0$.

Межові умови

$$\left[\frac{\partial v(z,t)}{\partial z}-h_v v(z,t)\right]|_{z=z0}=0.;\ \left[\frac{\partial v(z,t)}{\partial z}+h_v v(z,t)\right]|_{z=z0}=0. \tag{6.35}$$

Із урахуванням отриманих власних функцій $Z_k(\beta_k z)=\frac{h_v}{\beta_k}\sin(\beta_k z)+\cos(\beta_k z)$ Розв'язок рівняння отримуємо у вигляді

$$v^{(0)}(z,t)=\frac{1}{\text{P}Z_k\text{P}^2}\sum_1^N Z_k(\beta_k z)v_k e^{-\eta_k t}. \tag{6.36}$$

2. Отриманий розв'язок для швидкості $v^{(0)}(z,t)$ використовуємо для розв'язання лінійної частини рівняння відносно температури

$$[\rho c0_V - LT^*]\left(\frac{\partial T^{(0)}}{\partial t} + v^{(0)}\text{grad}T^{(0)}\right) = \text{div}(k\text{grad}T^{(0)}) + q_V. \qquad (6.37)$$

3. Відшукуємо розв'язок рівняння із урахуванням залежності коефіцієнта тепломісткості від температури розплаву $c0_V + L(T - T^*)\text{c} - \ T^{(1)}(z,t)$.

4. Визначаємо вираз для тиску із використанням формули $p = \rho RT^{(1)}(z,t)$.

5. Отримуємо розв'язок рівняння (6.30) у першому наближенні, тобто розв'язуємо рівняння

$$\rho\frac{\partial v^{(1)}}{\partial t} = -\frac{\partial p}{\partial z} + \mu\frac{\partial^2 v}{\partial z^2} - N_v(v^{(0)}, T^{(1)}) \qquad (6.38)$$

$$N_v(v^{(0)}, T^{(1)}) = v^{(0)}(z,t)\frac{\partial v^{(0)}}{\partial z} + R\rho\frac{\partial T^{(1)}}{\partial z}.$$

5. Пункти 2 – 5 повторюємо доти, доки $\left|T^{(m+1)}(z,t) - T^{(m)}(z,t)\right| > \Delta$ із заміною у викладених пунктах індексів 0 і 1 на m і $m+1$.

Оскільки ітераційна процедура розв'язання нелінійних рівнянь викладена у попередніх розділах, ми у цій частині її не наводимо.

6.4 Висновки по розділу 6

1. Сформульовано математичну модель нагрівання полімерної суміші у зоні її завантаження із урахуванням нелінійної залежності коефіцієнту тепломісткості від температури корпусу.

2. Розроблено метод розв'язання нелінійної крайової задачі теплоперенесення під час нагрівання суміші у зоні завантаження, що надало можливість отримати реальний розподіл температурного поля полімерної

суміші у зоні завантаження і визначити оптимальну довжину цієї зони.

3. Запропоновано алгоритм моделювання процесів тепломасоперенесення у зоні затримки процесу плавлення полімерної суміші як двохфазної задачі теплоперенесення тверда пробка – рідина на внутрішньому боці корпусу шнека (формування тонкої плівки розплаву полімеру).

Розділ 7. МОДЕЛЮВАННЯ ПРОЦЕСІВ ТЕПЛОМАСОПЕРЕНЕСЕННЯ У ЗОНІ ПЛАВЛЕННЯ

7.1 Формулювання задачі

Процеси у зоні плавлення полімерної суміші є визначальні з точки зору забезпечення якості ізоляційного покриття кабелів на надвисокі напруги.

Нагрів корпусу екструдера здійснюється із меншою потужністю джерела внутрішньої енергії порівняно із зоною завантаження для підтримки у зоні плавлення температури полімерної суміші (``пробки"), близької до температури плавлення полімеру. Алгоритм моделювання процесів нагріву корпусу та полімерної суміші у зоні плавлення є той самий, що й у зоні завантаження.

Задача моделювання процесів плавлення полімерної суміші полягає у вирішенні задачі у двохфазній області – тверда фаза – рідинна фаза, що відома як задача типу Стефана. Тому і формулювання задачі плавлення полімеру має бути як двохфазна задача.

Опис постановки задач типу Стефана та методам їх вирішення присвячена безліч праць, наприклад, [20,56]. У переважній більшості розглядаються одно- і двохфазні задачі для одновимірних моделей. Постановка задачі опимасна у розділі 3.6.

7.2 Огляд математичних моделей

Рівняння нагріву твердої суміші (пробки) є аналогічне рівнянню теплопровідності у зоні завантаження.

Розв'язок відповідної крайової задачі, що описує нагрів пробки до температури плавлення є аналогічний розв'язку відповідної крайової задачі у зоні завантаження. Відмінність полягає у тому, що за двохфазної зони нагріву--

плавлення умови на межі поділу фаз є функції часу (рух у часі межі поділу тверда фаза – рідинна фаза).

Із урахуванням циліндричної геометрії екструдера рівняння руху розплаву полімеру і температурного поля розплаву можна описати такою системою рівнянь:

рівняння нерозривності:

$$\frac{\partial(\rho v_z)}{\partial z}+\frac{1}{r}\frac{\partial(\rho v_\theta)}{\partial r}+\frac{1}{r}\frac{\partial(r\rho v_r)}{\partial r}=0, \tag{7.1}$$

рівняння руху:

$$\frac{\partial v_r}{\partial t}+v_r\frac{\partial v_r}{\partial r}+v_z\frac{\partial v_r}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial r}+\mu_e\left[\Delta v_r+\frac{\partial^2 v_z}{\partial r\partial z}\right]+g_r; \tag{7.2}$$

$$\frac{\partial v_z}{\partial t}+v_r\frac{\partial v_z}{\partial r}+v_z\frac{\partial v_z}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial z}+\mu_e\left[\Delta v_z+\frac{1}{r}\frac{\partial^2 v_r}{\partial r\partial z}+\frac{1}{r}\frac{\partial}{\partial z}\left(\frac{\partial v_r}{\partial r}\right)\right]+g_z; \tag{7.3}$$

рівняння енергії:

$$\frac{\partial T}{\partial t}+v_r\frac{\partial T}{\partial r}+v_z\frac{\partial T}{\partial z}=k\left(\frac{\partial^2 T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial z^2}\right)+\frac{A}{\rho c_v}q_T; \tag{7.4}$$

$$q_T=-T\frac{\partial p}{\partial T}|_\rho\left[\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{\partial v_z}{\partial z}\right]+2\mu_e[\left(\frac{\partial v_r}{\partial r}\right)^2$$

$$+\left(\frac{\partial v_z}{\partial z}\right)^2+\left(\frac{\partial v_z}{\partial r}+\frac{\partial v_r}{\partial z}\right)^2+\left(\frac{\partial v_\theta}{\partial z}\right)^2]. \tag{7.5}$$

A – термічний еквівалент роботи; c_v – питома тепломісткість рідини за сталого об'єму. μ_e – ефективна в'язкість при зсувній течії, ν – кінематична в'язкість.

$$\mu_e=\mu_0(I_2/2)^{\frac{n-1}{2}}e^{-\beta(T-T_0)}, \quad n=2\rightarrow\ \mu_e=\mu_0\sqrt{I_2/2}e^{-\beta(T-T_0)}, \tag{7.6}$$

$$(\nabla\cdot v)=\frac{1}{r}\frac{\partial(rv_r)}{\partial r}+\frac{\partial v_z}{\partial z}.$$

де μ_0 – початкова в'язкість; I_2 – другий інваріант тензору швидкостей

деформації; n – показник аномалії в'язкості; v_r, v_θ, v_z – компоненти вектору швидкості; τ_{ij} $(i, j = r, \theta, z)$– компоненти тензору напружень; T – температура, β – температурний коефіцієнт; T_0 – початкова температура.

Вираз $(\partial p/\partial T)|_\rho$ визначається із рівняння стану. У разі сталого значення ρ $p = \rho RT$ він дорівнює $R\rho$.

Рівняння (7.4) описує температурне поле у середовищі із урахуванням конвективної складової – вплив вектору швидкості на розподіл температурного поля.

Початкові умови для рівнянь (7.2)–(7.4) отримаємо, враховуючи розв'язок задачі тепломасоперенесення на виході із зони затримки плавлення:

$$v_r(r, z, t_0) = vz_r(r, z). \tag{7.7}$$

$$v_z(r, z, t_0) = vz_z(r, z). \tag{7.8}$$

7.3 Розв'язання задачі

Задача визначення швидкості руху межі поділу тверда фаза -- рідинна фаза складається із таких кроків.

7.3.1 Теплоперенесення у ``пробці''

Крайова задача про нагрівання твердої суміші (``пробки") у зоні плавлення є аналогічна задачі про нагрівання ``пробки" у зоні завантаження. Відмінність полягає у тому, що змінюються умови на межі Г фазового переходу тверда фаза – рідинна фаза у тому сенсі, що ця межа є рухома. Ітераційна процедура аналогічна викладеній у попередньому розділі із урахуванням необхідності обчислення інтегральних перетворень у змінних

межах $r \in [r_a, r_b' = r_b - \xi(z)] \subset \Omega_t$.

$$\rho_{\text{ї}} c_{v_{\text{ї}}}(T_{\text{ї}}) \left(V_{\text{øí}}^r \frac{\partial T_{\text{ї}}}{\partial r} + V_{\text{øí}}^z \frac{\partial T_{\text{ї}}}{\partial z} + \frac{\partial T_{\text{ї}}}{\partial t} \right) = \lambda_{\text{ї}} \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T_{\text{ї}}}{\partial r} \right) + \frac{\partial^2 T_{\text{ї}}}{\partial z^2} \right). \qquad (7.9)$$

Отже, алгоритм розв'язання рівняння (7.4) відрізняється від наведеного у попередньому розділі у частині обчислення інтегралів від функцій Бесселя другого роду з огляду на змінні межі. Інтегральні перетворення нелінійної складової рівняння теплопровідності (7.9):

$$nr_{m,n,l} = \frac{1}{\mathrm{P} B_m(\beta_m r_b') \mathrm{P}} \int_{r_a}^{r_b - \xi(z)} B(\beta_m r) B(\beta_n r) B(\beta_l r) r dr. \qquad (7.10)$$

Природно, що верхня межа інтегралів по r від аналогічних до (7.10) виразів по мірі просування ``пробки'' вздовж осі z зменшуватиметься і прямуватиме до нижньої межі r_a. Точка L, в якій $r_b' \approx r_a$ визначатиме довжину зони плавлення і перехід до зони гомогенізації та кристалізації розплаву полімеру.

7.3.2 Розв'язання задачі тепломасоперенесення у рідинній фазі

Оскільки рівняння відносно швидкості руху розчину полімеру містить градієнти тиску у розчині, почнемо розв'язання задачі із розв'язання крайової задачі відносно температури. Із урахуванням температурної залежності коефіцієнту тепломісткості c_v, а також наявності конвективної складової і дисипативної складової q_T маємо

$$\rho[c0_V + L\delta(T - T^*)] \left(\frac{\partial T}{\partial t} + v_r \frac{\partial T}{\partial r} + v_z \frac{\partial T}{\partial z} \right) = \lambda^+ \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2} \right) + q_T. \qquad (7.11)$$

Відокремимо у цьому рівнянні лінійну частину

$$\frac{\partial T}{\partial t}=\frac{\lambda^{+}}{c0_{V}\rho}\left(\frac{\partial^{2}T}{\partial r^{2}}+\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^{2}T}{\partial z^{2}}\right)-C_{T}-\frac{1}{\rho}[L\delta(T-T^{*})]\frac{\partial T}{\partial t}+q_{T}, \qquad (7.12)$$

де λ^{+} – стрибкоподібне змінювання коефіцієнту теплопровідності на межі поділу тверда фаза – рідинна фаза, через C_T позначено конвективну складову у рівнянні теплоперенесення

$$C_{T}=v_{r}\frac{\partial T}{\partial r}+v_{z}\frac{\partial T}{\partial z}. \qquad (7.13)$$

За аналогією запровадимо позначення для конвективних складових компонент швидкості руху розплаву в екструдері.

$$C_{v_r}=v_{r}\frac{\partial v_{r}}{\partial r}+v_{z}\frac{\partial v_{r}}{\partial z}; \qquad (7.14)$$

$$C_{v_z}=v_{r}\frac{\partial v_{z}}{\partial r}+v_{z}\frac{\partial v_{z}}{\partial z}. \qquad (7.15)$$

Визначення температури рідинної фази у лінійному наближенні. Ця задача вирішена у попередньому розділі, оскільки у лінійному наближенні (без урахування конвективних складових і дисипативної складової у правій частині рівняння (7.12) воно збігається із рівнянням теплоперенесення у пробці):

$$T^{(0)}(r,z,t)=\sum_{n,k}^{M,N}R_{m}(r)Z_{k}(z)[T0_{m,k}+T1_{m,k}e^{-\gamma_{m,k}t}]. \qquad (7.16)$$

Наявність градієнту тиску у рівняннях відносно швидкості руху розплаву врахуємо відповідно до рівіняння $p=\rho RT$, тобто $\partial p/\partial r=R\rho\partial T/\partial r$, $\partial p/\partial r=R\rho\partial T/\partial z$ із урахуванням (7.16).

Тоді рівняння руху можна записати у такому вигляді.

$$\frac{\partial v_{r}}{\partial t}=-R\frac{\partial T^{(0)}}{\partial r}+\mu_{e}\left[\Delta v_{r}+\frac{\partial^{2}v_{z}}{\partial r\partial z}\right]-C_{r}; \qquad (7.17)$$

$$\frac{\partial v_z}{\partial t} = -R\frac{\partial T^{(0)}}{\partial z} + \mu_e\left[\Delta v_z + \frac{1}{r}\frac{\partial^2 v_r}{\partial r \partial z} + \frac{1}{r}\frac{\partial}{\partial z}\left(\frac{\partial v_r}{\partial r}\right)\right] - C_z; \qquad (7.18)$$

Позначимо, як і раніше, лінійну частину для компонент швидкості руху розплаву через $v_r^{(0)}(r,z,t)$, $v_z^{(0)}(r,z,t)$. застосування рівнянь (7.17), (7.18) інтегральних перетворень за просторовими змінними дає для $m = 1,2,\ldots$:

$$\frac{d\overline{\overline{v_r^{(m)}}}}{dt} + \gamma_{n,k}^{v_r}\overline{\overline{v_r^{(m)}}} = -R\overline{\overline{T^{(m-1)}}}_r(t) = F_r^{(m-1)}(t)_{n,k} + \overline{\overline{C_r}}\left(v_r^{(m-1)}(r,z,t), v_z^{(m-1)}(r,z,t)\right);$$

$$\frac{d\overline{\overline{v_z^{(m)}}}}{dt} + \gamma_{n,k}^{v_z}\overline{\overline{v_z^{(m)}}} = -R\overline{\overline{T^{(m-1)}}}_z(t) = F_z(t)_{n,k} + \overline{\overline{C_r}}\left(v_r^{(m-1)}(r,z,t), v_z^{(m-1)}(r,z,t)\right);$$

Із урахуванням виразу (7.16) після застосування інтегральних перетворень за просторовими змінними у ``лінійному наближенні'' матимемо

$$(V^{(0)})_r^{m,k}(p) = \frac{1}{p + \gamma_{m,k}^{v_r}}\left[vr_{m,k} - \sum_{m_1,k_1}^{M,K} r2_{m,m_1} zd_{k,k_1}\left(\frac{T0_{m_1,k_1}}{p} + \frac{T1_{m_1,k_1}}{p + \gamma_{m_1,k_1}^{T}}\right)\right].$$

$$(V^{(0)})_z^{m,k}(p) = r_m z_k \frac{1}{p + \gamma_{m,k}^{v_z}}\left[vz_{m,k} - \sum_{m_1,k_1}^{M,K} r2_{m,m_1} zd_{k,k_1}\left(\frac{T0_{m_1,k_1}}{p} + \frac{T1_{m_1,k_1}}{p + \gamma_{m_1,k_1}^{T}}\right)\right],$$

де $r2_{m,m_1}$, zd_{k,k_1} – коефіцієнти інтегральних перетворень за змінними r і z відповідно.

Підсумовування у цих виразах із застосуванням алгоритму еквівалентного спрощення дає:

$$(V^{(0)})_r^{m,k}(p) \approx \frac{vr_2}{p} + \frac{vr_0 + vr_1 p}{vr_3 + vr_4 p + vr_5 p^2};$$

$$(V^{(0)})_z^{m,k}(p) \approx \frac{vz_2}{p} + \frac{vz_0 + vz_1 p}{vz_3 + vz_4 p + vz_5 p^2}.$$

У цих виразах позначено:

$$vr_{m,k} = r_m z_k vr0,\ vz_{m,k} = r_m z_k vz0;$$

$$rd_{m,m_1} = \int_{r_a}^{r_b} R_m(\beta_m r)\left(\frac{\partial R_{m_1}}{\partial r}(\beta_{m_1} r)\right)dr;$$

$$zd_{k,k_1} = \int_{z_0}^{z_l} Z_k(\alpha_k z)\left(\frac{\partial Z_{k_1}}{\partial z}(\alpha_{k_1} z)\right)dz,$$

де $vr0, vz0$ – початкові значення швидкості – рух "пробки" у поздовжньому і радіальному напрямках.

Оригінали від цих виразів є:

$$vr^{(0)}(r,z,t) == \sum_{m=1}^{M} R_m(\beta_m r)\sum_{k=1}^{N} Z_k(\alpha z)\mathrm{Vrt}_{m,k}^{(0)}(t); \tag{7.19}$$

$$\mathrm{Vrt}_{m,k}^{(0)}(t) = \mathrm{vrt}_2 + e^{-\gamma_{m,k}^{v_r} t}\left(\mathrm{vrt}_1 \sin(\omega_{v_r} t) + \mathrm{vrt}_1 \cos(\omega_{v_r} t)\right); \tag{7.20}$$

$$vz^{(0)}(r,z,t) = \sum_{m=1}^{M} R_m(\beta_m r)\sum_{k=1}^{N} Z_k(\alpha z)\mathrm{Vzt}_{m,k}^{(0)}(t) \tag{7.21}$$

$$\mathrm{Vzt}_{m,k}^{(0)}(t) = \mathrm{vzt}_2 + e^{-\gamma_{m,k}^{v_z} t}\left(\mathrm{vzt}_1 \sin(\omega_{v_z} t) + \mathrm{vzt}_1 \cos(\omega_{v_z} t)\right); \tag{7.22}$$

За наявності виразів для компонент швидкості руху розплаву у лінійному наближенні можна визначити конвективні складові у рівняннях для швидкостей і температури. Замістимо (7.16), (7.19),(7.20) у (7.13), (7.14),(7.15). Наведемо вираз для C_{v_r}. Вираз для C_{v_z} – аналогічний.

$$C_{v_r}^{(0)} = \sum_{m_1,m_2}^{M}\sum_{k_1,k_2}^{K} R_{m_1}(\beta_{m_1} r)Z_{k_1}(\alpha_{k_1} z)\mathrm{Vrt}_{m_1,k_1}^{(0)}(t)\frac{dR_{m_2}(\beta_{m_2} r)}{dr}Z_{k_2}(\alpha_{k_2} z)\mathrm{Vrt}_{m_2,k_2}^{(0)}(t)$$

$$+\sum_{m_1,m_2}^{M}\sum_{k_1,k_2}^{K} R_{m_1}(\beta_{m_1} r)Z_{k_1}(\alpha_{k_1} z)\mathrm{Vzt}_{m_1,k_1}^{(0)}(t)R_{m_2}(\beta_{m_2} r)\frac{dZ_{k_2}(\alpha_{k_2} z)}{dz}\mathrm{Vrt}_{m_2,k_2}^{(0)}(t).$$

Застосування до цього виразу інтегральних перетворень за просторовими змінними призводить до виразів вигляду

$$\overline{\overline{C_{v_r}^{(0)}}}_{m,k} = \sum_{m_1,m_2}^{M} \sum_{k_1,k_2}^{K} F_{m1,m_2,k_1,k_2}^{m,k}(t);$$

$$F_{m_1,m_2,k_1,k_2}^{m,k}(t) = F1_{m_1,m_2,k_1,k_2}^{m,k} \mathrm{Vrt}_{m_1,k_1}^{(0)}(t) \mathrm{Vrt}_{m_2,k_2}^{(0)}(t) + F2_{m_1,m_2,k_1,k_2}^{m,k} \mathrm{Vzt}_{m_1,k_1}^{(0)}(t) \mathrm{Vrt}_{m_2,k_2}^{(0)}(t).$$

Підсумовування за індексами m_1, m_2, k_1, k_2 із застосуванням алгоритму еквівалентного спрощення (розділ 4) дає:

$$C_r^{m,k}(p) \approx \frac{vr_2^{m,k}}{p} + \frac{vr_0^{m,k} + vr_1^{m,k} p}{vr_3^{m,k} + vr_4^{m,k} p + vr_5^{m,k} p^2}; \tag{7.23}$$

За аналогічним алгоритмом отримуємо вираз для

$$C_z^{m,k}(p) \approx \frac{vz_2^{m,k}}{p} + \frac{vz_0^{m,k} + vz_1^{m,k} p}{vz_3^{m,k} + vz_4^{m,k} p + vz_5^{m,k} p^2}; \tag{7.24}$$

$$C_T^{m,k}(p) \approx \frac{t_2^{m,k}}{p} + \frac{t_0^{m,k} + t_1^{m,k} p}{t_3^{m,k} + t_4^{m,k} p + t_5^{m,k} p^2}. \tag{7.25}$$

Алгоритм обчислення коефіцієнтів $vr_j^{m,k}$, $vz_j^{m,k}$, $t_j^{m,k}$ у конвективних складових викладений у розділі 4.

Оскільки дисипативна складова у рівнянні для температури може бути записана у вигляді

$$q_T = \frac{2A\mu_e}{\rho c 0_V}\left(\frac{\partial v_r}{\partial r} + \frac{\partial v_z}{\partial z}\right)^2,$$

із урахуванням (7.23), (7.24) матимемо

$$Q_T^{m,k}(p) \approx \frac{2A\mu_e}{\rho c 0_V}\left[\frac{Tq_2}{p} + \frac{Tq_0 + Tq_1 p}{Tq_3 + Tq_4 p + Tq_5 p^2}\right]. \tag{7.26}$$

Тоді у першому наближенні для температурного поля у просторі зображень матимемо:

$$\overline{\overline{T^{(1)}}}_{m,k}(p) = \overline{\overline{T^{(0)}}}_{m,k}(p) - C_T^{m,k}(p) + \overline{\overline{q_T}}_{m,k}(p) \approx \frac{t1_{m,k}^0 + t1_{m,k}^1 p}{t1_{m,k}^3 + t1_{m,k}^4 p + t1_{m,k}^5 p^2}.$$

Або у просторі оригіналів

$$T^{(1)}(r,z,t) = \sum_{m=1}^{M}\sum_{k=1}^{N} R_m(r) Z_k(z) TT_{n,k}^{(1)}(t) \tag{7.27}$$

$$TT_{n,k}^{(1)}(t) = t1_{m,k}^{(0)} - e^{-t1_{m,k}^{(4)} t}\,(t1_{m,k}^{(1)} f_1(t1_{m,k}^{(5)} t) + t1_{m,k}^{(2)} f_2(t1_{m,k}^{(5)} t), \tag{7.28}$$

де $f_1(at), f_2(at)$ – оригінали від ланцюгів другого порядку.

Обчислення цих інтегралів за алгоритмами, що наведені у розділі , виконується один раз для всіх ітерацій. У результаті розв'язки для компонент швидкості руху розплаву і температурного поля набувають такого вигляду:

$$v_r^{(m)}(r,z,t) = \sum_{n=1}^{M} R_n(r) \sum_{k=1}^{N} Z_k(z) vr_{n,k}^{(m)}(t); \tag{7.29}$$

$$v_r^{(m)}(r,z,t) = \sum_{n=1}^{M} R_n(r) \sum_{k=1}^{N} Z_k(z) vz_{n,k}^{(m)}(t). \tag{7.30}$$

На рис. 7.1–7.2 наведено графіки компонент швидкості $v_r^{(m)}(r,z,t)$:

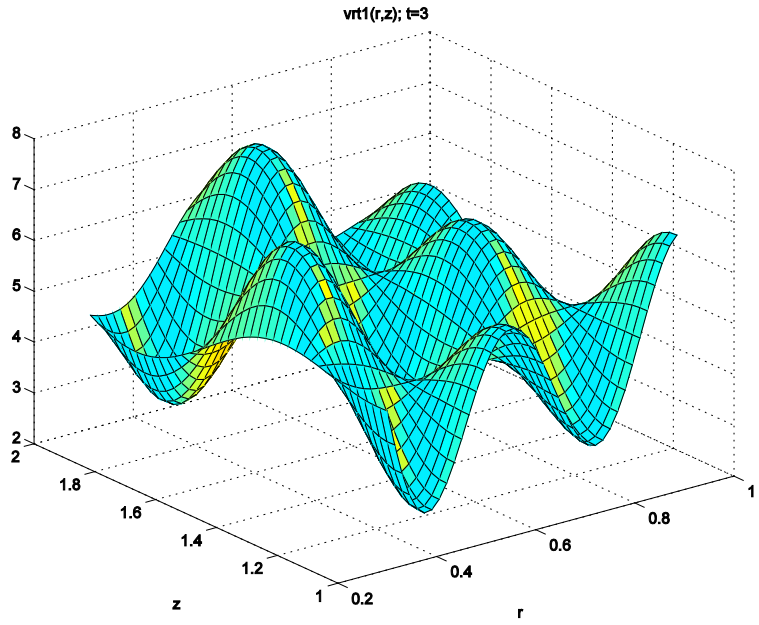


Рис.7.1 Поле радіальної швидкості у першому наближенні

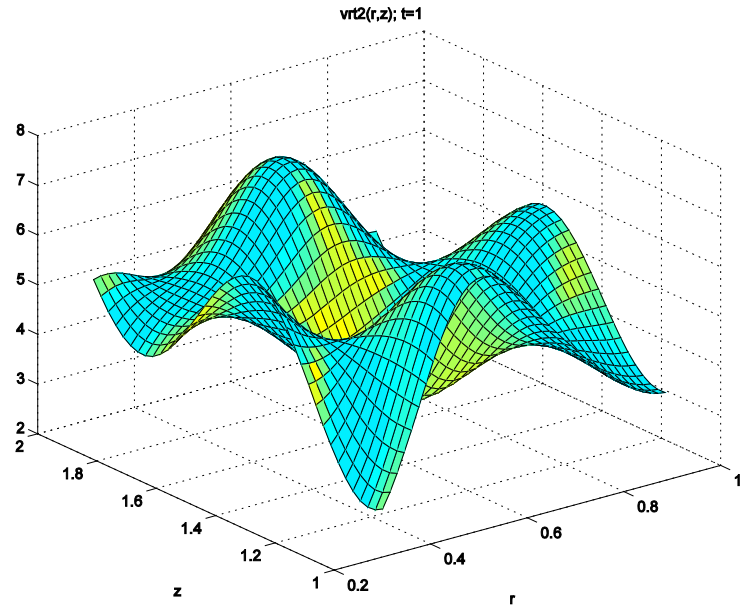


Рис.7.2 Поле радіальної швидкості у другому наближенні

На рис. 7.3–7.4 наведено графіки компонент швидкості $v_z^{(m)}(r,z,t)$:

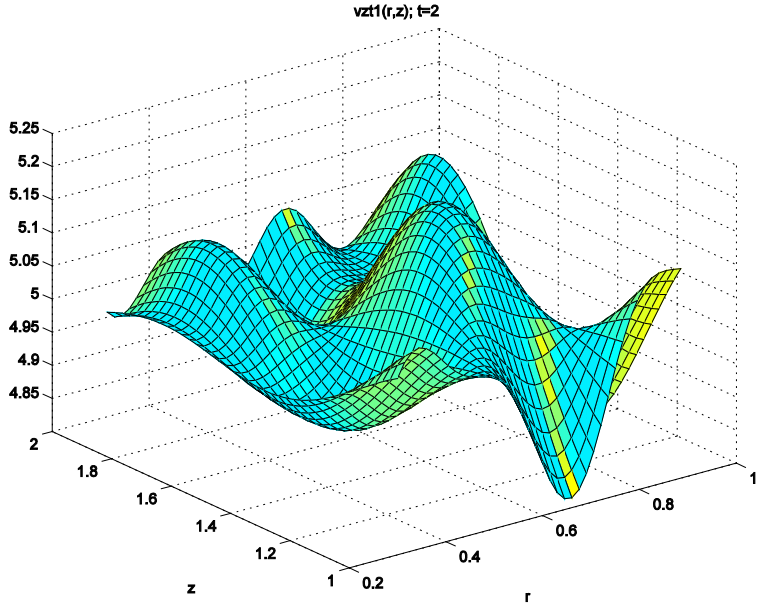


Рис.7.3 Поле осьової швидкості у першому наближенні

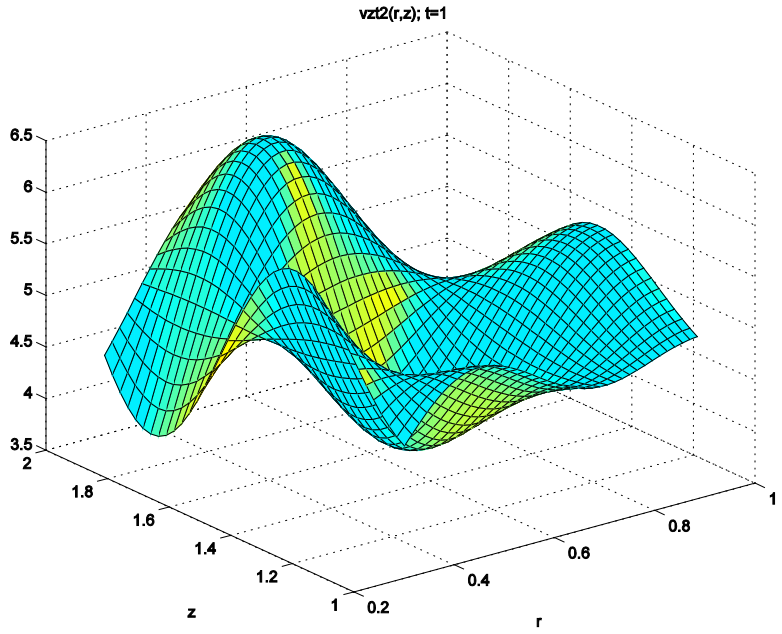


Рис.7.4 Поле осьової швидкості у другому наближенні

Застосування ітераційної процедури розв'язання нелінійного рівняння теплопровідності можна записати для m-го наближення:

$$T^{(m)}(r,z,t)=\sum_{m=1}^{M}\sum_{k=1}^{N}R_m(r)Z_k(z)TT_{n,k}^{(m)}(t) \qquad (7.31)$$

$$TT_{n,k}^{(m)}(t)=t1_{m,k}^{(0,m-1)}-e^{-t1_{m,k}^{(4,m-1)}t}(t1_{m,k}^{(1,m-1)}f_1(t1_{m,k}^{(5,m-1)}t)+t1_{m,k}^{(2,m-1)}f_2(t1_{m,k}^{(5,m-1)}t)), \qquad (7.32)$$

Наведемо графіки температурних полів, отриманих на перших 3-х ітераціях.

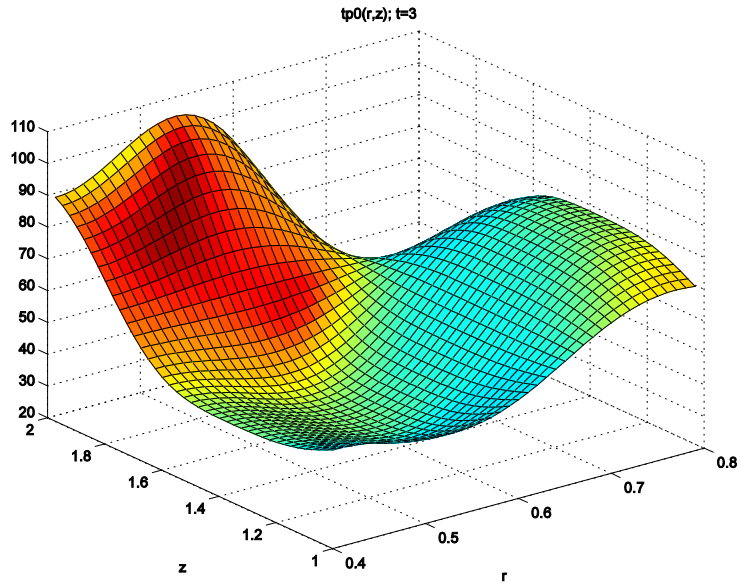


Рис.7.5 Температурне поле у лінійному наближенні

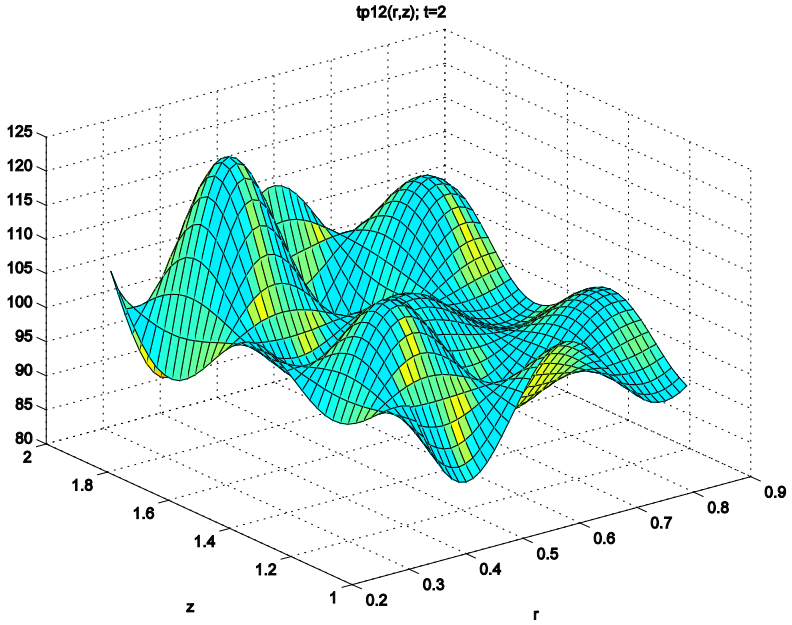


Рис.7.6 Температурне поле у першому наближенні

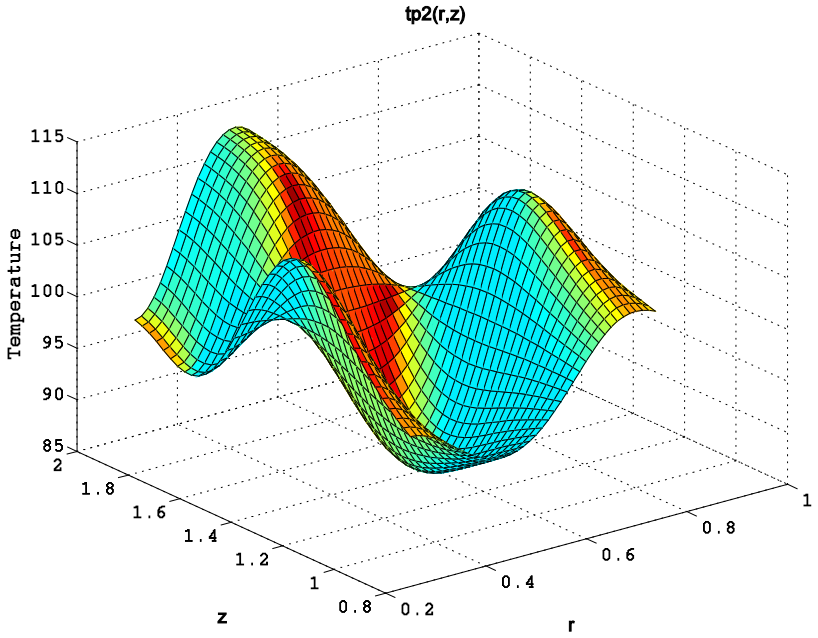


Рис.7.7 Температурне поле у другому наближенні

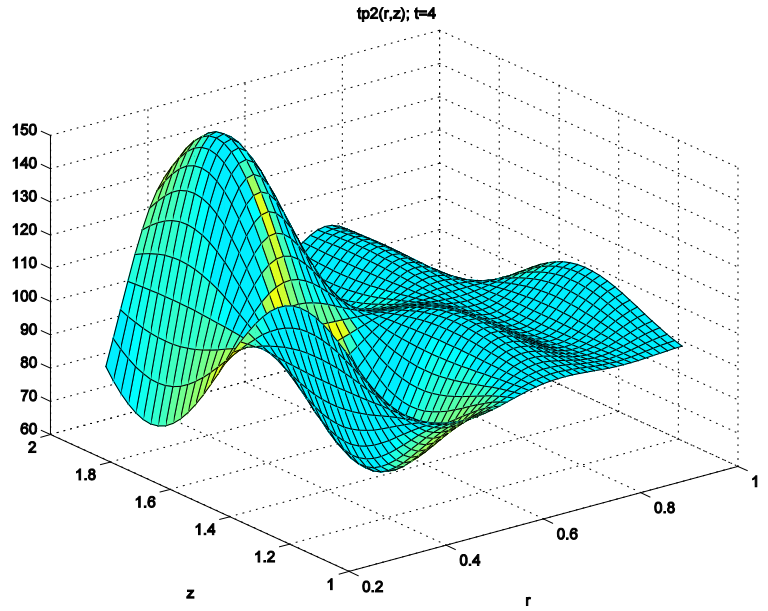


Рис.7.8 Температурне поле у третьому наближенні

За наявним наближеним розв'язком (7.31) для температурного поля розплаву полімеру у першому наближенні можемо виписати вираз для поля тиску

$$p^{(m)}(r,z,t) = \rho R T^{(m)}(r,z,t) = \sum_{n=1}^{M}\sum_{k=1}^{N} R_n(r) Z_k(z) TT_{n,k}^{(m)}(t) \qquad (7.33)$$

із урахуванням виразу (7.32) для $TT_{n,k}^{(m)}(t)$.

Оскільки нас цікавить значення тиску на виході із зони плавлення полімеру, слушно визначати значення тиску на межі зони плавлення L і зони дозування:

$$p^{(m)}(r,L,t) = \rho R T^{(m)}(r,L,t) = \sum_{n=1}^{M}\sum_{k=1}^{N} R_n(r) Z_k(L) TT_{n,k}^{(m)}(t) \qquad (7.34)$$

Отриманий розв'язок задачі про нагрівання ``пробки" – твердої полімерної суміші та процесу утворення тонкої плівки розплаву у кінці зони затримки плавлення можна розглядати як перший крок у визначенні межі фазового переходу тверда фаза – рідинна фаза.

Утворення рідинної фази супроводжується, природно, зменшенням частки твердої фази (``пробки"). Процес плавлення призводить до утворення межі ($r \in [r_a, r_b - \xi]$) для твердої фази і $r \in (r_b, r_a + \xi)$) для рідинної фази.

Суттєва відмінність задачі про фазовий перехід у процесі плавлення полімеру на відміну від класичних задач типу Стефана полягає у тому, що:

1) у процесі плавлення полімеру в екструдері тверда суміш (``пробка") рухається у напрямку осі z зі швидкістю V_z;

2) це призводить до залежності змінної межі від поздовжньої координати z, тобто $\xi = \xi(z)$.

7.3.3 Визначення межі фазового переходу

На межі фазового переходу мають виконуватися умови збереження тепла на вільній межі, що поділяє тверду фазу і розчинну:

$$L_V V_n\,|_\Gamma = \frac{\lambda_s}{\rho_s}\frac{\partial T_s}{\partial n}|_{\Omega_s} - \frac{\lambda_l}{\rho_l}\frac{\partial T_l}{\partial n}|_{\Omega_l}, \qquad (7.35)$$

де $V_n = d\xi/dz$ – швидкість руху межі фазового переходу, L_V – теплота фазового переходу (визначається експериментально).

У розділі 5 отримано вираз для температурного поля полімерної суміші у вигляді:

$$D_{nk}^{(0)}(t) = \sum_{n,k=1}^{M} e^{-\alpha_{nk}t}\left(\begin{cases} e1_{nk}\sin\omega_{nk}t + e2_{nk}\cos\omega_{nk}t, & \omega_{nk} < 0, \\ e1_{nk}\,\omega_{nk}t + e2_{nk}\,\omega_{nk}t, & \omega_{nk} > 0 \end{cases}\right). \qquad (7.36)$$

Тоді розв'язок у першому наближенні набуде вигляду

$$T^{(1)}(r,z,t) = \sum_{n=1}^{M}\sum_{k=1}^{M} R(\delta_n r)Z(\beta_k z)\left[\frac{G_{nk}}{\eta_{nk}}(1-e^{-\eta_{nk}t}) + D_{nk}^{(0)}(t)\right]. \qquad (7.37)$$

Подальші наближення реалізуються за аналогічною схемою:

$$T^{(m)}(r,z,t) = \sum_{n=1}^{M}\sum_{k=1}^{M} R(\delta_n r)Z(\beta_k z)\left[\frac{G_{nk}}{\eta_{nk}}(1-e^{-\eta_{nk}t}) + D_{nk}^{(m-1)}(t)\right]. \qquad (7.38)$$

З огляду на геометрію шнеку ($R_{\text{øí}} = L$) можна вважати, що

$$\frac{\partial T}{\partial n} \approx \frac{\partial T}{\partial r} = \sum_{m=1}^{M}\frac{\partial B(\beta_m r)}{\partial r}|_{r=r_b'}\sum_{k=1}^{K} Z(\alpha_k z)tt_{m,k}^{(pl)}(t).$$

Тоді для твердої і рідинної фаз можна записати:

$$\frac{\partial T_s}{\partial n} \approx \frac{\partial T_s}{\partial r} = \sum_{m=1}^{M}\frac{\partial B(\beta_m r)}{\partial r}|_{r=r_b'}\sum_{k=1}^{K} Z(\alpha_k z)tt_{m,k}^{(sl)}(t).$$

$$\frac{\partial T_l}{\partial n} \approx \frac{\partial T_l}{\partial r} = \sum_{m=1}^{M}\frac{\partial B(\beta_m r)}{\partial r}|_{r=r_a'}\sum_{k=1}^{K} Z(\alpha_k z)tt_{m,k}^{(pl)}(t).$$

Із урахуванням того, що густина матеріалу полімеру майже не змінюється із переходом із твердого у рідинний стан, маємо таке рівняння для визначення швидкості руху межі розділу фаз:

$$\rho L_V V_n = \frac{1}{\rho_{pl}}\sum_{m=1}^{M}\frac{\partial B(\beta_m r)}{\partial r}|_{r=r'}\sum_{k=1}^{K} Z(\alpha_k z)\left[\lambda_s tt(sl)_{m,k}(t) - \lambda_l tt_{m,k}^{(pl)}(t)\right].$$

Далі, для того, щоб отримати вираз для межі поділу фаз, зробимо припущення, що процес плавлення полімеру в екструдері із утворенням межі фазового переходу можна вважати квазістаціонарним. Тоді можна записати рівняння для визначення межі фазового переходу у такому вигляді:

$$\rho L_V V_n = \frac{1}{\rho_{pl}} \sum_{m=1}^{M} \frac{\partial B(\beta_m r)}{\partial r} |_{r=r'} \sum_{k=1}^{K} Z(\alpha_k z) \left[\lambda_s tt_{m,k}^{(sl)}(0) - \lambda_l tt_{m,k}^{(pl)}(0) \right].$$

$$V_n = \frac{d\xi}{dz} = \frac{1}{\rho L_V} \sum_{k=1}^{K} Z(\alpha_k z) C_k.$$

$$C_k = \frac{1}{\rho_{pl}} \sum_{m=1}^{M} \frac{\partial B(\beta_m r)}{\partial r} |_{r=r'} \left[\lambda_s tt_{m,k}^{(sl)}(0) - \lambda_l tt_{m,k}^{(pl)}(0) \right].$$

Початкова умова для цього рівняння:

$$\xi(0) = \frac{1}{\rho L_V} \sum_{k=1}^{K} Z(\alpha_k 0) C_k.$$

7.4 Висновки по розділу 7

1. Сформульовано задачу моделювання процесів плавлення полімерів в екструдері. Математична модель процесу являє собою систему диференціальних рівнянь, що враховує конвективне перенесення рідини, нелінійну залежність від температури теплофізичних параметрів полімеру та дисипативну складову.

2. Отримано розв'язок цієї системи рівнянь із використанням ітераційного числово- аналітичного методу розв'язання нелінійних крайових задач.

3. Сформульована математична модель тепломасоперенесення як задача зі змінними межами – задача типу Стефана. Запропоновано метод розв'язання задачі про визначення рухомої межі поділу фаз тверда фаза – рідинна фаза.

4. Отриманий вираз для рухомої межі може використовуватися для постановки задачі оптимального управління процесом плавлення полімерів.

Розділ 8. МОДЕЛЮВАННЯ ПРОЦЕСІВ КРИСТАЛІЗАЦІЇ ПОЛІМЕРНИХ РОЗЧИНІВ В ЕКСТРУДЕРІ

На рис. 8.1 [43] представлена модель охолодження шнеку: 1 – шнек; 2 – область охолодження шнеку; 3 – трубка для подачі рідини, що охолоджує.

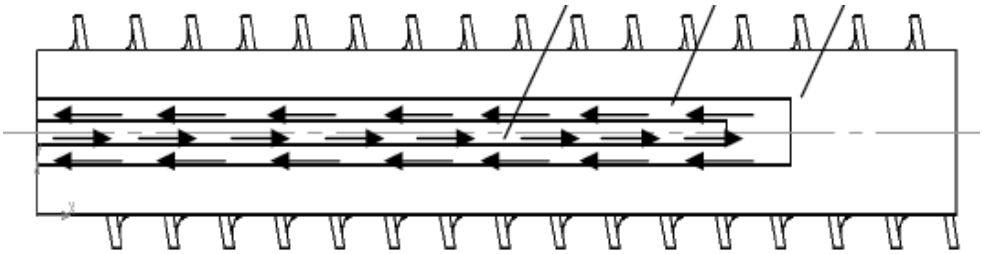


Рис. 8.1 Модель охолодження шнеку

Охолодження ізоляції супроводжується кристалізацією полімерного матеріалу.

Перехід полімерів із рідинного стану у твердий супроводжується складними хіміко- фізичними процесами – багатостадійними реакціями полімеризації мономеру, переходом аморфної фази полімеру у кристалічну. Аналіз цього переходу починають із процесу полімеризації, який визначає просторово-часовий розподіл температури для процесу кристалізації. Нестаціонарні, неізотермічні і просторово розподілені явища полімеризації і кристалізації визначають напружений і деформований стан полімеру.

8.1 Огляд підходів до побудови моделей

Побудові математичних моделей кристалізації різного призначення присвячено значну кількість наукових і прикладних праць [6,43,102,105,116,121,143,162,173,176, 189—195,198,214,215—220,229,298,311].

Ми зупонимося на математичному опису процесу кристалізації [215], як показовому підході до вирішення цієї проблеми, де розглядається задача кристалізації виробу циліндричної форми із зовнішнім охолодженням (тобто, охолодженням з боку внутрішньої поверхні шнеку), (позначення авторські).

Модель складається з двох нелінійних диференціальних рівнянь для ступеню кристалічності $y(r,t)$, визначеного як середня об'ємна частка простору, займаного кристалами, і температурного поля $T(r,t)$, поєднане за допомогою норми функції зародження і зростання $b_N(T)$ і $b_G(T)$, функція запуску нуклеації $k(y)=1-y^2$ і функція агрегації та насичення ядер $\beta(y)=y(1-y)$:

$$\frac{\partial y}{\partial t}=\beta(y(r,t))b_G(T(r,t))+v_0k(y(r,t))b_N(T(r,t)), \tag{8.1}$$

$$\frac{\partial T}{\partial t}=\sigma\left(\frac{\partial^2T}{\partial r^2}+\frac{1}{r}\frac{\partial T}{\partial r}\right)+a_G\beta(y(r,t))b_G(T(r,t)), \tag{8.2}$$

для $(r,t)\in[r_c,r_a]\times(0,\tau)$, τ – час, на який зупиняється процес охолодження. Просторові довжини визначаються як $l=r_a-r_c$, l_z – висота зразка вздовж осі z.

Охолодження здійснюється за температури $u_c(t)\in[0,T_f]$ (відповідно $u_a(t)\in[0,T_f]$) до відповідної сторони r_c (r_a) впродовж загального часу охолодження $\tau>0$.

Межові умови рівнянь (8.1)– (8.2):

$$T(r_c,t)=u_c(t),\ \frac{\partial T(r_a,t)}{\partial r}=0,\ t\in(0,\tau), \tag{8.3}$$

і початкові умови:

$$y(r,0)=0,\ T(r,0)=T_0,\ r\in(r_c,r_a). \tag{8.4}$$

Співвідношення між функцією зародження і темпами зростання вважається сталим.

Охолодження у процесі кристалізації описує така однофазна проблема Стефана:

$$\frac{\partial T}{\partial t} = \sigma\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right),\ r \in (r_c, R_c(t)),\ t > 0, \tag{8.5}$$

$$T(r,t) = T_f,\ r \in [R_c(t), \infty), t > 0, \tag{8.6}$$

$$T(r_c,t) = u_c(t),\ t > 0, \tag{8.7}$$

$$\frac{L_\delta}{c}\frac{dR_c}{dt} = \sigma\frac{\partial T}{\partial r}(R_c(t)^-,t),\ t > 0. \tag{8.8}$$

(8.8) – умова Стефана. Тут L_δ – приховане тепло, а c –- питоме тепло. В [217] показано, що $L/c = a_G K_\delta$ – функція $\delta = v_0 N/G$, безрозмірне співвідношення між коефіцієнтами зародження та зростання (та початковою масою):

$$K_\delta = \frac{1 + \delta(\ln\delta - 1)}{(1-\delta)^2}.$$

Вираз прихованої теплоти L_δ дає аналітичний вираз числа Стефана Ste_c, безрозмірне число визначається як відношення чутливого тепла $c\Delta T = c\max_t\{T_f - u_c(t)\}$ до L_δ:

$$\mathrm{Ste}_c = \frac{c\Delta T}{L_\delta}.$$

Число Стефана Ste_c характеризує процес зміни фази. Коли Ste_c = 1 (тобто, коли прихована теплота велика, що є у нашому випадку), можна припустити псевдостабільний стан динамічної системи (8.5)–(8.8), [217]. Це означає, що рівняння (8.8) можна замінити на $T_t = 0$, так що

$$\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} = 0.$$

У наведеній математичній моделі кристалізації розчину полімеру відсутні зв'язки визначальних складових b_G, b_N у рівнянні (1) від температури розчину.

У [11] розглядається математична модель, в якій залежність коефіцієнту кристалізації β від температури розплаву задається в явному вигляді.

Запроваджуються два макрокінетичні параметри α і β, які визначають

питомий внесок відповідно полімерної і кристалічної фаз. Ступінь полімеризації $\alpha(x,t)$ визначає ступінь завершеності процесу полімеризації і може приймати значення від 0 (вміст полімеру – 0) до 1 (весь мономер перейшов у полімер). Ступінь кристалізації $\beta(x,t)$ визначає ступінь завершеності процесу кристалізації і може приймати значення від 0 (вміст кристалічної фази – 0) до 1 (уся аморфна фаза полімеру перейшла у кристалічну).

Для кількісної характеристики ступеню завершеності фазового перетворення використовують параметр $\beta \in [0,1]$, що має назву відносного ступеню кристалічності.

Визначення полів температур $T(r,z)$ і ступеню кристалічності $\beta(r,z)$ в ізоляції полягає у сумісному розв'язанні рівняння теплопровідності і кінетичного рівняння [109]:

$$V\frac{\partial T}{\partial z} = \frac{1}{c\rho}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)\right]+Q_v,\ r\in(R_g,R_{iz}),\ z\in(0,L), \quad (8.9)$$

$$\frac{\partial\beta}{\partial z} = \frac{1}{V}\left[K_0\exp\left(-\frac{U}{RT}-\frac{\psi T_{iz}}{T(T_{iz}-T)}\right)\right](1+C_0\beta)[\beta_p(T)-\beta], \quad (8.10)$$

$$r\in(R_g,R_{iz}),\ z\in(0,L),\quad Q_v=\frac{VQ_m}{c}\frac{\partial\beta}{\partial z},$$

де $T_{pl}=415K$ – рівноважна температура плавлення ПЕНГ; R – універсальна газова стала; Q_v – швидкість питомого тепловиділення при кристалізації; $Q_m=377$ кДж/кг – експериментально визначене максимальне тепловиділення при кристалізації ПЕНГ; β_p – рівноважний відносний ступінь кристалічності ПЕНГ (максимально можливий відносний ступінь кристалічності при визначеній температурі); $C_0=1{,}8$, $K_0=0{,}041/c$; $U=-1000$ Дж/моль; $\psi=216K$ – експериментально отримані параметри кінетичного рівняння.

Межові умови:

$$T|_{z=0} = T_g,\ r \in (0, R_g),\ T|_{z=0} = T_{iz}, \tag{8.11}$$

$$\frac{\partial T}{\partial z}|_{z=L} = 0,\ \beta|_{z=0} = 0,\ r \in [R_g, R_{iz}], \tag{8.12}$$

$$\frac{\partial T}{\partial r}|_{r=0} = 0,\ \lambda_g \frac{\partial T}{\partial r}|_{r=R_g-0} = \lambda_g \frac{\partial T}{\partial r}|_{r=R_g+0}, \tag{8.13}$$

$$T|_{r=R_g-0} = T|_{r=R_g+0},\ z \in (0, L), \tag{8.14}$$

$$\frac{\partial T}{\partial r}|_{r=R_{iz}} = -\frac{\alpha_n}{\lambda_{iz}}(T - T_{sr})|_{r=R_{iz}},\ z \in (0, L). \tag{8.15}$$

α_n – коефіцієнт тепловіддачі на поверхні жили; T_{sr} – температура середовища, що охолоджує; T_g, T_{iz} – початкова температура жили та ізоляції.

Зазначимо, що рівняння (8.9)– (8.10) записані відносно процесів в ізоляції, а межові умови (8.11)– (8.15) містять умови для струмопровідної жили.

Далі автори досліджують задачу про напружено- деформований стан струмопровідної жили і термомеханічну поведінку полімеру, що кристалізується.

Наведена математична модель кристалізації полімерів стосується, за думкою авторів, процесам накладання полімерної ізоляції на струмопровідну жилу. Але ж процес кристалізації полімерного розчину здійснюється ще до накладання ізоляції на жилу, тобто, у зоні дозування екструдера. Тому такий підхід до дослідження процесу кристалізації полімерних розчинів не можна вважати таким, що відповідає суті справи.

8.2 Постановка задачі

У [162] наведено формулювання математичної моделі як моделі, що враховує процеси полімеризації та кристалізації із якною заліжністю цих характеристик від температури розчину полімеру.

За припущення, що технологічні напруження не впливають на температуру і протікання процесу кристалізації, треба вирішувати такі задачі:

1) задачу визначення температурних полів і ступеню кристалізації, або теплокінетичну задачу;

2) крайову задачу визначення напружено-деформованого стану (НДС) системи, що твердіє.

Задача про визначення температурних полів і ступеню полімеризації та кристалізації описується такою крайовою задачею:

$$c(T)\rho(T)\frac{\partial T(x,t)}{\partial t} = \mathrm{div}(\lambda(T)\mathrm{grad}T(x,t)) + \rho(T)\left(Q_\alpha \frac{d\alpha}{dt} + Q_\beta \alpha \frac{d\beta}{dt}\right);\ x \in V; \quad (8.16)$$

кінетичне рівняння полімеризації

$$\frac{d\alpha(x,t)}{dt} = K_\alpha(1-\alpha(x,t))(1+c_0\alpha(x,t))(1+n\alpha(x,t)) \quad (8.17)$$

кінетичне рівняння кристалізації

$$\frac{d\beta(x,t)}{dt} = K_\beta(T)(1+A_0\beta(x,t))(\beta_p(T)-\beta(x,t)),\ x \in V. \quad (8.18)$$

$$K_\alpha = k_\alpha \exp\left(-\frac{U}{RT}\right);\ K_\beta = k_\beta \exp\left(-\frac{E}{RT} - \frac{\Psi}{T_p - T}\right). \quad (8.19)$$

Початкові та межові умови:

$$\beta(x,0) = 0,\ T(x,0) = T_0; \quad (8.20)$$

$$\lambda(T)n\mathrm{grad}T(x,t) = h(T(x,t) - T_{av});\ n \cdot \mathrm{grad}T(x,t) = 0,\ x \in S_2; \quad (8.21)$$

$$T(x,t) = T^*(x,t);\ x \in S_1. \quad (8.22)$$

де c – питома теплоємність; ρ – густина; λ – коефіцієнт теплопровідності; Q_α, Q_β – інтенсивності теплових джерел, що обумовлені полімеризацією і кристалізацією відповідно; R – універсальна газова стала; $U, E, k_\alpha, k_\beta, c_0, c_1, n$ – кінетичні сталі, що визначаються експериментально із калориметричних вимірювань; T_p – температура плавлення; β_p – рівноважний ступінь

кристалізації.

Теплофізичні властивості полімеру низької густини (ПЕНГ):

c – 2,0–3,5 кДж/кг К; λ – 0,29–0,42 Вт/м К; T_{pl} – 376–388 К; T_c – 138 К; теплота плавлення – 7 кДж/моль.

Результат розв'язання системи рівнянь (8.16)–(8.22) є визначення температурного поля і ступеню кристалізації, із урахуванням яких вирішується задача напружено- деформованого стану (НДС) виробу.

Слід зауважити, що рівняння (8.16) не відповідає суті задачі, оскільки це рівняння має описувати процеси руху розплаву полімеру, що у зоні дозування полімеризується і кристалізується і воно має враховувати перенесення маси.

Конкретизуємо рівняння (8.16) у циліндричній системі координат:

$$c(T)\rho(T)\frac{\partial T}{\partial t}+v_r\frac{\partial v_r}{\partial r}+v_z\frac{\partial v_r}{\partial z}$$

$$=\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(T)\frac{\partial T}{\partial r}\right)+\frac{\partial}{\partial z}\left(\lambda(T)\frac{\partial T}{\partial z}\right)+\rho(T)\left(Q_\alpha\frac{d\alpha}{dt}+Q_\beta\alpha\frac{d\beta(t)}{dt}\right). \quad (8.23)$$

Початкові і межові умови:

$$\alpha(r,t)|_{t=0}=0,\ \beta(r,t)|_{t=0}=0;\ T(r,t)|_{t=0}=T_0;$$

$$\left[\frac{\partial T}{\partial r}-h_0(T-T_0)\right]|_{R_1}=0;\ \left[\frac{\partial T}{\partial r}+h_1(T-T_0)\right]|_{R_2}=0.$$

$Q_m=377$ кДж/кг – експериментально визначене максимальне тепловиділення при кристалізації ПЕНГ.

Канал охолодження:

$$v_z|_{z=0}=f_2(r);\ \frac{\partial v_r}{\partial r}|_{r=0}=0,\ \frac{\partial v_z}{\partial z}|_{r=0}=0.$$

Межові умовина твердих поверхнях для компонентів агенту, що охолоджує, відповідають умовам прилипання та непроникнення. У центрі каналу охолодження

$$\lambda_{\text{îõë}} \frac{\partial T}{\partial r}|_{r=R_k} = \lambda_{\text{íåò}} \frac{\partial T}{\partial r}|_{r=R_k};$$

$$T^+|_{r=R_k} = T^-|_{r=R_k},\ T|_{z=0} = T_{\text{âõ}},\ \frac{\partial v}{\partial z}|_{z=L_{\text{øí}}} = 0.$$

Моделювання процесів полімеризації і кристалізації виконується за таких значень параметрів для ПЕНГ:

Для ПЕНГ вихідні дані: $K_1 = 2{,}33 \cdot 10^4\ 1/c$, $U_1 = 30200$ Дж/моль,, $\Psi = 182\ K$, $T_p = 415\ K$, $A_0 = 82$, $Q_k = 164000\ Дж/кг;$, $k = 0{,}1$, $\mu_a = 4{,}19 \cdot 10^6\ Pa$, $B_{cr} = 1{,}78 \cdot 10^8\ Pa$, $a_a = 2{,}8 \cdot 10^{-5}\ 1/K$, $a_{cr} = 0{,}9 \cdot 10^{-5}\ 1/K$, $B_a^{\infty} = 1{,}08 \cdot 10^8\ Pa$. $R = 8{,}314$ Дж/(моль К); 1 Дж =4,189 кал.

Швидкість охолодження зовнішньої поверхні циліндру 0,2 К/с.

B_{cr} – модуль об'ємного стискання кристалічної фази.

У рівняннях позначено: λ_0 – коефіцієнт теплопровідності; Q_p – тепловий ефект полімеризації; Q_{cr} – тепловий ефект кристалізації (швидкість питомого тепловиділення при кристалізації ПЕНГ); k_α, k_β – сталі швидкостей полімеризації і кристалізації; U – енергія активації процесу полімеризації; E^a – енергія активації процесу кристалізації; $\varepsilon_1, \varepsilon_2$ – критерії автокаталітичності процесів полімеризації і кристалізації; Ψ – характерна температура полімеру; R – універсальна газова стала; h_0, h_1 – коефіцієнти теплообміну із навколишнім середовищем; $T_p = 415$ К – рівноважна температура плавлення; β_b – рівноважний ступінь кристалічності:

$$\beta_b = 0{,}52\sqrt{1 - (T/T_p)^4}.$$

Розрахунки процесу кристалізації полімерних розчинів виконуються із застосуванням різницевих схем, але результати розрахунків наводяться для моделей в одновимірному поданні, що не відповідає суті справи.

8.3 Розв'язання системи рівнянь

Розглянемо рівняння відносно ступеню полімеризації α та ступеню кристалізації β. При цьому на першому етапі вважатимемо $K_\alpha = k_\alpha$, $K_\beta = k_\beta$.

$$(1-\alpha)(1+c_0\alpha)(1+n\alpha) = (1-\alpha)[1+(c_0+n)\alpha + c_0 n\alpha^2]$$

$$= 1+(c_0+n-1)\alpha + (c_0 n - (c_0+n))\alpha^2 - c_0 n\alpha^3 = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3.$$

$$a_0 = 1;\ a_1 = c_0;\ a_2 = -1;\ a_3 = -c_0,\ (n=1).$$

Маємо рівняння

$$\frac{d\alpha}{dt} = K_\alpha[1 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3]. \tag{8.24}$$

$$K_\alpha = k_\alpha \exp\left(-\frac{U}{RT}\right). \tag{8.25}$$

Оскільки це рівняння містить нелінійну складову відносно $T(r,t)$, треба апроксимувати експоненту, наприклад, дробово-раціональним виразом типу

$$e^{\mp x} \approx \frac{12 \mp 6x + x^2}{12 \pm 6x + x^2},$$

який за $x < 5$ забезпечує [158] гарне наближення до експоненти.

Позначимо $C = U/R$. Тоді матимемо

$$K_\alpha(T) = k_\alpha \frac{C^2/12 - C/2T + T^2}{C^2/12 + C/2T + T^2}.$$

Заміщення цього виразу у рівняння (8.17) дає:

$$(C^2/12 + C/2T + T^2)\frac{d\alpha(x,t)}{dt} = (C^2/12 - C/2T + T^2)K_\alpha(1 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3)$$

Відокремимо у цьому рівнянні складові відносно α та T,α:

$$\frac{d\alpha(x,t)}{dt} = (1 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3) + N_\alpha(T,\alpha); \tag{8.26}$$

N_α – нелінійна складова рівняння:

$$N_\alpha = T[(T - C/2)K_\alpha(1 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3) - (C/2 + T)\frac{d\alpha(x,t)}{dt}] \qquad (8.27)$$

Отримаємо спочатку розв'язок рівняння (8.24) за $N_\alpha = 0$.

$$\frac{d\alpha^{(0)}}{dt} = k_\alpha(1 + a_1\alpha),\ \alpha(0) = 0;$$

$$\overline{\alpha}^{(0)}(p) = \frac{k_\alpha}{p(p - a_1)} \to \alpha^{(0)}(t) = \frac{K_\alpha}{a_1}\left(e^{a_1 t} - 1\right)$$

$$\overline{\alpha}^{(1)}(p) = \frac{k_\alpha}{p(p - a_1)} + \mathsf{L}_t\{a_2[\alpha^{(0)}(t)]^2 + a_3[\alpha^{(0)}(t)]^3\}$$

$$= \frac{k_\alpha}{p(p - a_1)} + \mathsf{L}_t\left\{\frac{a_2}{a_1^2}\left(e^{a_1 t} - 1\right)^2 + \frac{a_3}{a_1^3}\left(e^{a_1 t} - 1\right)^3\right\};$$

Позначимо $c0 = K_\alpha / a_0$.

$$\overline{\alpha}^{(1)}(p) = \frac{1}{c0^2}\frac{-2p^3 + c_0(5 + c_0)p^2 - (12 + 5c_0)p + 6c_0^3(1 + c_0)}{p(p^3 - 6c_0p^2 + 11c_0^2p - 6c_0^3)}$$

$$= \frac{a_0}{p} + \frac{a_1p^2 + a_2p + a_3}{p^3 - 6c_0p^2 + 11c_0^2p - 6c_0^3} \approx \frac{A_2}{p} + \frac{A_0 + A_1p}{A_3 + A_4p + A_5p^2};$$

$a_0 = -(1 + c_0), a_1 = c_0 - 1, a_2 = c_0(11 + 7c_0), a_3 = -5c_0 - 12 - 11c_0^2(1 + c_0), a_k = a_k / c_0^2;$

або у просторі оригіналів за Лапласом:

$$\alpha^{(1)}(t) \approx A_2 + e^{-\gamma_\alpha t}[\overline{A}_0(\omega_\alpha t) + \overline{A}_1(\omega_\alpha t)]. \qquad (8.28)$$

Повернимося до виразу (8.27). У ньому наявний вираз для температури розплаву полімеру $\overline{\overline{T^{pl}}}_{n,k}(t)$.

Цей вираз отримано у попередньому розділі. Застосування інтегральних перетворень за просторовими змінними дає:

$$\overline{\overline{N_\alpha^{(1)}}}(t) = \int_{r_2}^{r_2+\alpha} R_n(\beta r) R_{n_1}(\beta r) dr \int_{z_0}^{L} Z_k(\delta_k z) Z_k(\delta_{k_1} z) dz N_\alpha(r,z,t).$$

Заміщення виразу для температури розплаву у просторі зображень за просторовими змінними $\overline{\overline{tt^{(3)}}}_{n,k}(t)$ у $N_\alpha(r,z,t)$ дає

$$\overline{\overline{N_\alpha^{(1)}}}(t) = \sum_{n_1,k_1} \overline{\overline{tt}}_{n_1,k_1}(t) \left[(\overline{\overline{tt}}_{n_1,k_1}(t) - C/2)\alpha^{(1)} - (\overline{\overline{tt}}_{n_1,k_1}(t) - C/2) \frac{d(\alpha^{(1)}}{dt} \right].$$

Перетворення за Лапласом цього виразу після виконання алгоритмів спрощення дає

$$\overline{\overline{\overline{N_\alpha^{(1)}}}}(p) \approx \frac{N_2}{p} + \frac{N_0 + N_1 p}{N_3 + N_4 p + p^2} \to N_\alpha^{(1)}(t) = na_0 + e^{-na_4 t}(na_1 \sin(na_5 t) + na_2 \cos(na_5 t).$$

Із урахуванням отриманого виразу маємо

$$\alpha^{(2)}(t) = \alpha^{(1)}(t) + N_\alpha^{(1)}(t) = A_0 + e^{-A_4 t}(A_1 \sin(A_5 t) + A_2 \cos(A_5 t).$$

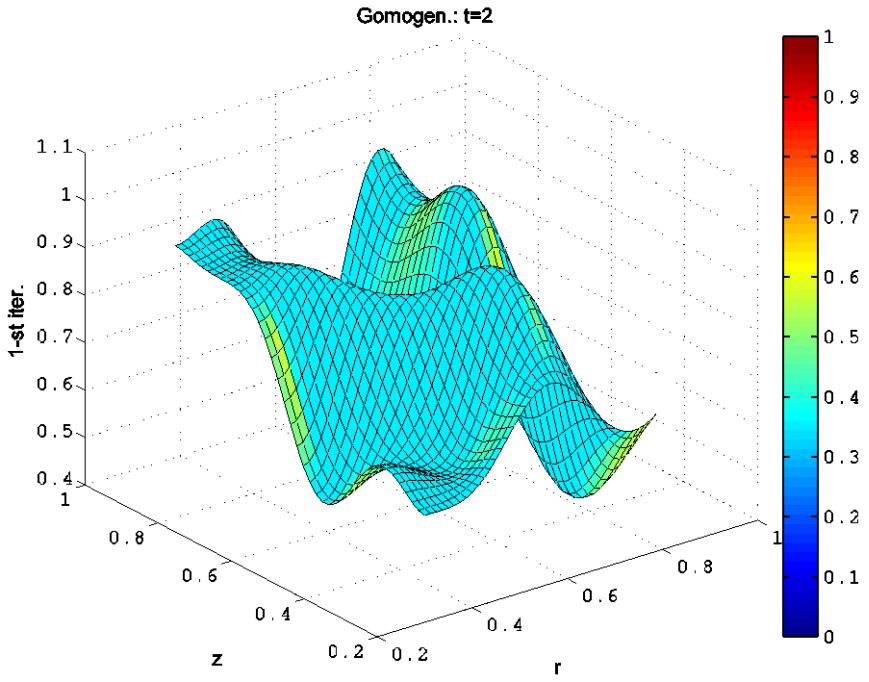


Рис. 8.2 Графік гомогенізації розплаву полімеру на 1-й ітерації

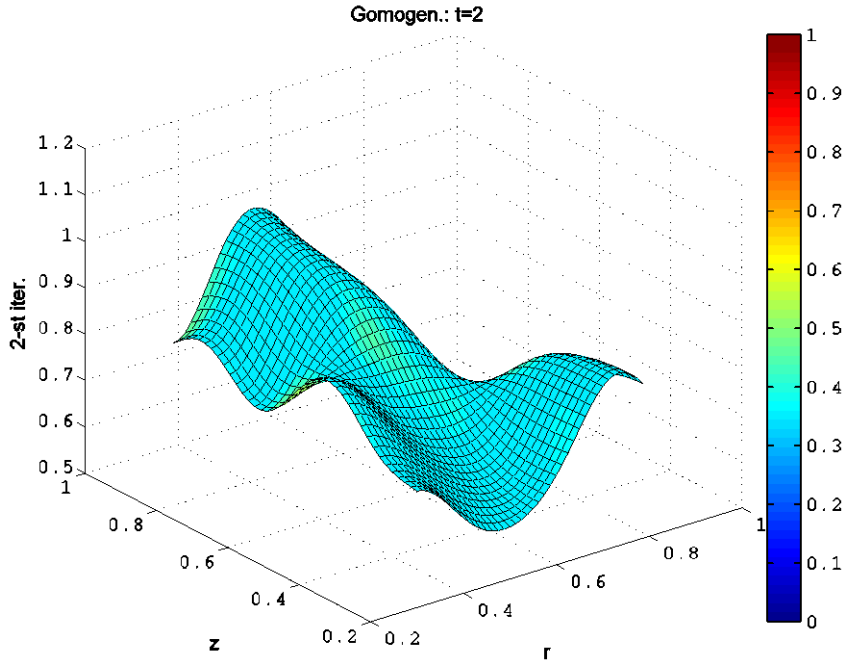


Рис. 8.3 Графік гомогенізації розплаву полімеру на 2-й ітерації

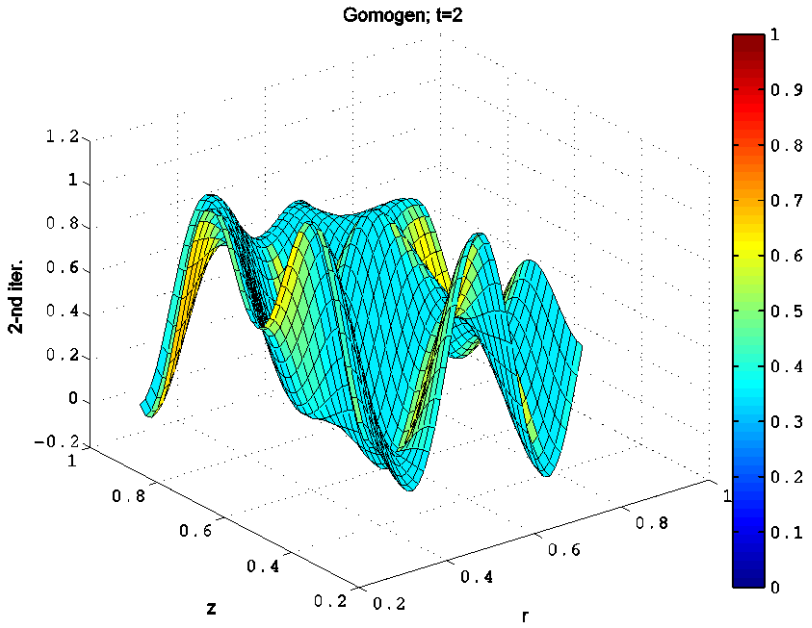


Рис. 8.4 Графік гомогенізації розплаву полімеру на 3-й ітерації

На рис. 8.2–8.4 наведено графіки розподілу частки гомогенізації розплаву за фіксованих значень часу. За аналогією перетворимо функцію K_β у (8.22).

$$K_\beta \approx \frac{d_0 + d_1T + d_2T^2 + d_3T^3 + d_4T^4}{c_0 + c_1T + c_2T^2 + c_3T^3 + c_4T^4};\ \ A = E/R,\ \ r_1 = AT_p,\ \ r_2 = \Psi - A.$$

Коефіцієнти d_i, c_i, $i = \overline{0,4}$, мають вигляд

$$d_0 = c_0 = r_1^2;\ d_1 = 2r_1(r_2 - 3T_p),\ \ d_2 = 12T_p^2 + r_2^2 - 6(r_2T_p - r_1),$$

$$d_3 = -6(4T_p - r_2),\ \ d_4 = c_4 = 1,\ c_1 = 2r_1(r_2 + 3T_p),$$

$$c_2 = 12T_p^2 + r_2^2 + 6(r_2T_p - r_1),\ \ c_3 = -6(4T_p + r_2);\ d_k = d_k/12,\ c_k = c_k/12,\ k = \overline{0,4}.$$

Рівняння (18) набуває вигляду:

$$(c_0 + c_1T + c_2T^2 + c_3T^3 + c_4T^4)\frac{d\beta(x,t)}{dt}$$

$$= (d_0 + d_1T + d_2T^2 + d_3T^3 + d_4T^4)(e_0 + e_1\beta + e_2\beta^2). \qquad (8.29)$$

де

$$e_0 = \beta_p,\ \ e_1 = c_1\beta_p - 1,\ \ e_2 = -c_1.$$

Отже, маємо систему диференційних рівнянь (8.16), (8.24), (8.29) відносно температури розплаву полімеру та ступеню гомогенізації α і кристалізації β із відповідними початковими і межовими умовами.

Ця система рівнянь є нелінійна, її розв'язок шукатимемо за ітераційною схемою за аналогією із попередніми розділами. Маємо:

$$c_0\frac{d\beta^{(m+1)}(x,t)}{dt} = d_0(e_0 + e_1\beta^{(m)} + e_2(\beta^{(m)})^2) + N_\beta(T, \beta^{(m)}); \qquad (8.30)$$

$$N_\beta(T, \beta^{(m)}) = T(d_1 + d_2T + d_3T^2 + d_4T^3)(e_0 + e_1\beta^{(m)} + e_2(\beta^{(m)})^2)$$

$$-T(c_1 + c_2T + c_3T^2 + c_4T^3)\frac{d\beta^{(m)}(x,t)}{dt}. \qquad (8.31)$$

$$\frac{d\beta^{m)}(x,t)}{dt} = (e_0 + e_1\beta^{(m)} + e_2(\beta^{(m)})^2).$$

За аналогією маємо

$$\overline{\beta}^{(0)}(p)=\frac{e_0}{p(p-e_1)}\to\beta^{(0}(t)=\frac{e_0}{e_1}(e^{e_1t}-1);\ e_0'=e_0/d_0,\ e_1'=e_1/d_0.$$

$$\beta^{(1)}(t)\approx B_2+e^{-\gamma_\beta t}[B_0(\omega_\beta t)+B_1(\omega_\beta t)]. \qquad (8.32)$$

На наступному кроці приєднаємо до отриманого розв'язку залежність α і β від температури розплаву $T(x,t)$.

Розв'язок рівняння теплопровідності (8.16) у ``нульовому наближенні", тобто без урахування похідних від ступеню полімеризації α і ступеню кристалізації β ($Q_k=0$) отримано вище.

$$T^{(m)}(r,z,t)=\sum_{n=1}^{M}\sum_{k=1}^{M}R(\delta_n r)Z(\beta_k z)\left[\frac{G_{nk}}{\eta_{nk}}(1-e^{-\eta_{nk}t})+D_{nk}^{(m-1)}(t)\right]. \qquad (8.33)$$

Оскільки вирази для $\alpha^{(1)}(t)$, $\beta^{(1)}(t)$ не задежать від просторових координат, треба знайти вираз для складової рівняння (16) $Q_p\frac{\partial\alpha}{\partial t}+Q_{cr}\frac{\partial\beta}{\partial t}$, тобто знайти вирази для похідних від $\alpha^{(1)}(t)$, $\beta^{(1)}(t)$.

$$L_T=Q_p\frac{d\alpha(t)}{dt}+Q_{cr}\frac{d\beta(t)}{dt}$$

$$=Q_pe^{-\gamma_\alpha t}\left[-(\gamma_\alpha A_0+\omega_\alpha A_1)(\omega_\alpha t)+(\omega_\alpha A_0-\gamma_\alpha A_1)(\omega_\alpha t)\right]$$

$$+Q_{cr}e^{-\gamma_\beta t}[-(\gamma_\beta B_0+\omega_\beta B_1)(\omega_\beta t)]+(\omega_\beta B_0-\gamma_\beta B_1)(\omega_\beta t)].$$

Цей вираз є лінійна функція від t, перетворення Лапласа від нього має вигляд

$$\overline{L}_T=Q_p\frac{C_1\omega_\alpha+C_2(p+\gamma_\alpha)}{(p+\gamma_\alpha)^2-\omega_\alpha^2}+Q_{cr}\frac{C_3\omega_\beta+C_4(p+\gamma_\beta)}{(p+\gamma_\beta)^2-\omega_\beta^2}\approx\frac{D_0+D_1p}{D_3+D_4p+p^2}.$$

Згідно загальній схемі застосування перетворення Лапласа маємо

$$\frac{1}{p+\eta_{n,k}}\mathsf{L}_t\{L_T\}=\frac{1}{p+\eta_{n,k}}\frac{D_0+D_1p}{D_3+D_4p+p^2}\approx\frac{E_0+E_1p}{E_3+E_4p+p^2}.$$

Об'єднаємо цей вираз із $D^{(m-1)}(t)$. Маємо

$$F_{n,k}^T(t)=D^{(m-1)}(t)+r1_n z1_k\mathsf{L}_{-1}\left\{\frac{E_0+E_1p}{E_3+E_4p+p^2}\right\}. \qquad (8.34)$$

де $r1_n, z1_k$ – інтеграли від власних функцій за змінними r,z. У просторі оригіналів маємо

$$F_{n,k}^T(t)=f_2^{n,k}+e^{-\gamma_f^{n,k}t}[f_0^{n,k}(\omega_f^{n,k}t)+f_1^{n,k}(\omega_f^{n,k}t)]. \qquad (8.35)$$

Отже, замість (8.33) маємо

$$T^{(m)}(r,z,t)=T_{\text{pol}}(r,z,t)\approx\sum_{n=1}^{M}\sum_{k=1}^{M}R(\delta_n r)Z(\beta_k z)F_{n,k}^T(t). \qquad (4)$$

Коефіцієнти у виразах (8.34), (8.36) обчислюються за допомогою відповідної програми мовою *C*.

Наступний крок полягає у виконанні відповідних перетворень в (8.26), (8.31) з метою обчислення нелінійних складових у рівняннях (8.24), (8.30).

Розглянемо нелінійну складову (8.27) у рівнянні (8.24).

Із урахуванням виразу для $\alpha^{(1)}(t)$ (8.25) апроксимуємо добуток

$$\begin{aligned}\alpha^{(m)}(t)&=1+a_1\alpha+a_2\alpha^2+a_3\alpha^3=1+a_1[A_2+e^{-\gamma_\alpha t}[\overline{A}_0(\omega_\alpha T)+\overline{A}_1(\omega_\alpha t)]\\&+a_2\left\{A_2+e^{-\gamma_\alpha t}[\overline{A}_0(\omega_\alpha T)+\overline{A}_1(\omega_\alpha t)]\right\}^2\\&+a_3\left\{A_2+e^{-\gamma_\alpha t}[\overline{A}_0(\omega_\alpha T)+\overline{A}_1(\omega_\alpha t)]\right\}^3\\&\approx A_2^\alpha+e^{-\gamma_\alpha t}[A_0^\alpha(\omega_\alpha t)+A_1^\alpha(\omega_\alpha t)]. \qquad (8.37)\end{aligned}$$

У правій частині цього виразу ми зберегли позначення для сталих затримки γ_α та частоти ω_α коливань, щоб не запроваджувати додаткові

позначення.

Обчислення коефіцієнтів виразу (8.37) здійснюється за розробленим алгоритмом, який тут не наводиться, засобами відповідної програми, що реалізує цей алгоритм. Загальний вигляд цього алгоритму викладено у розділі 4. Наприклад, $A_2^{\alpha} = 1 + a_1 A_2 + a_2 A_2^2 + a_3 A_2^3$, тощо.

Розглянемо добуток $T[T \mp C/2]$ у (8.27).

$$T[(T \mp C/2)] = \sum_{n,n_1=1}^{M} \sum_{k,k_1=1}^{M} R(\delta_n r) Z(\beta_k z) R(\delta_{n_1} r) Z(\beta_{k_1} z) F_{n,k}^{T}(t) \overline{F}^{T}_{n_1,k_1}(t). \qquad (5)$$

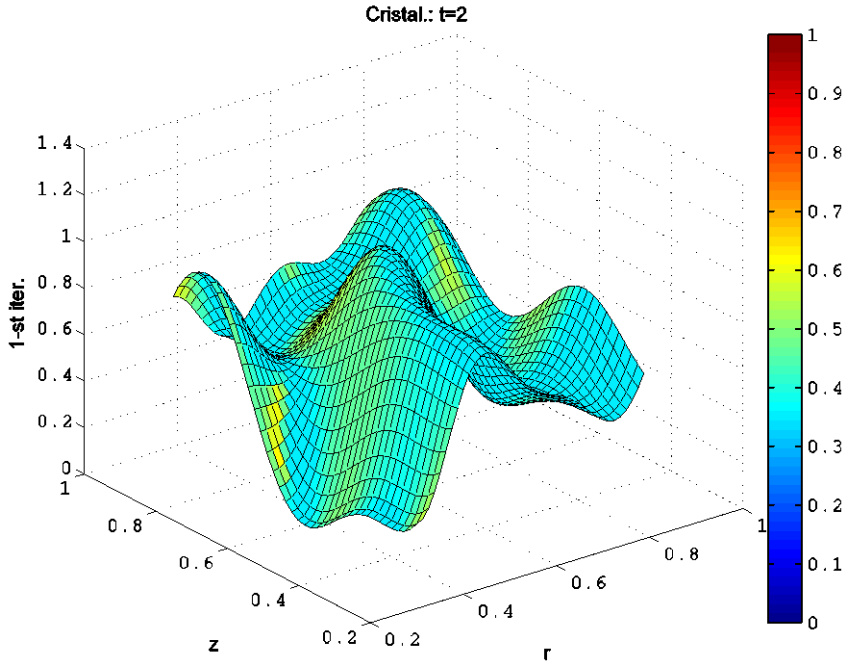
$\overline{F} = F \mp C/2$, тобто $\overline{f}_2^{n_1 k_1} = f_2^{n_1 k_1} \mp C/2$.

Оскільки вирази (8.37), (8.32) не залежать явно від просторових координат, для визначення значень добутку (8.38) на (8.37) та його похідну досить виконати операції добутку у часовій області та знайти їх зображення. Після виконання алгебраїчних операцій із цими зображеннями та подальшого переходу до оригіналів можна записати:

$$G^{n,k}(t) = F_{n,k}^{T}(t) \overline{F}^{T}_{n_1,k_1}(t) \alpha^{(m)}(t) \to g_0 + e^{-\alpha_g t}[g_1 \sin(\omega_g t) + g_2 \cos(\omega_g t)]. \quad (8.39)$$

Знову таки, спрощення розв'язку відносно часової змінної до виразу вигляду (8.39) компенсується збільшенням кількості ітерацій під час розв'язання системи рівнянь (у даному випадку для досягнення точності 5% знадобилося чотири ітерації).

Залежність ступеню полімеризації $\alpha(x,t)$ та ступеню кристалізації $\beta(x,t)$ від просторових координат x є неявна, оскільки рівняння для цих параметрів не залежать від цих координат у явному вигляді. Ця залежність виявляється після урахування коефіцієнтів K_α, K_β як функцій температури розплаву полімеру. Це означає, що ці параметри визначаються у фіксованих точках x як функції часу.



Графік кристалізації розплаву полімеру на 1-й ітерації

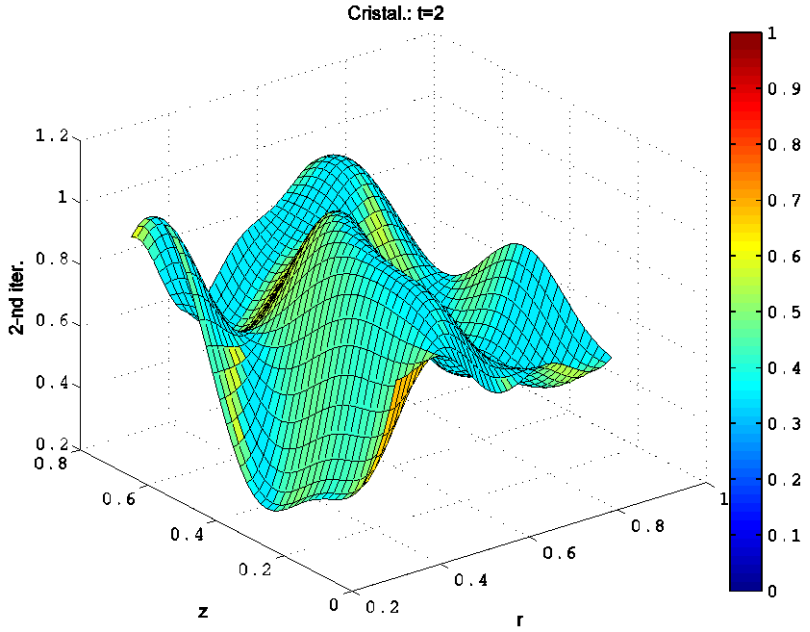


Рис. 8.5 Графік кристалізації розплаву полімеру на 3-й ітерації

На рис. 8.4—8.5 наведено графіки розподілу частки кристалізації за фіксованих значень часу.

8.4 Висновки по розділу 8

1. Мета математичного моделювання процесів полімеризації та кристалізації полягала у визначенні рівноважного стану цих процесів та визначенні довжини зони дозування.

2. Сформульовано задачу про визначення ступеню полімеризації і кристалізації розплаву полімеру у зоні дозування екструдера у вигляді системи нелінійних диференційних рівнянь відносно температури розплаву (рівняння масо і теплопереносу) із урахуванням межових умов охолодження та нелінійних кінетичних рівнянь ступеню полімеризації і кристалізації розплаву полімеру.

3. Розв'язання системи рівнянь руху розплаву полімеру (теплопровідності) та кінетичних рівнянь ступеню полімеризації та кристалізації здійснюється числово- аналітичним ітераційним методом, що надало можливість отримати розв'язок у квадратурах.

4. Отримано розподіл температури розплаву, що кристалізується у зоні дозування, ступеню полімеризації та кристалізації полімеру.

ЗАГАЛЬНІ ВИСНОВКИ

Основний результат дисертації – вирішення важливої науково-технічної проблеми по створенню наукових засад удосконалення і створення нового обладнання для комплексного екструзійного перероблення термопластичних матеріалів, які забезпечують визначення оптимальних конструктивно-технологічних параметрів екструзійного обладнання на базі одношнекових екструдерів, що сприятиме заощадженню енергетичних і матеріальних ресурсів, які необхідні для виконання численних експериментальних досліджень, та отриманню продукції високої якості.

1. Сформульовано математичні процесів нагрівання корпусу екструдера, нагрівання полімерної суміші у зоні завантаження та плавлення, що коректно враховують вплив нелінійних фізико-технічних параметрів полімерного матеріалу на процеси масо- і теплоперенесення у відповідних зонах.

2. Запропоновано ітераційний числово-аналітичний метод розв'язання крайових задач для систем нелінійних диференційних рівнянь із частинними похідними математичної фізики. Застосування цього методу надає можливість отримувати розв'язки таких крайових задач у квадратурах, що є суттєвим при розробці та реалізації систем автоматичного управління відповідними технологічними процесами.

3. Розроблено алгоритмічне забезпечення методів розв'язання крайових задач для систем нелінійних диференційних рівнянь як із звичайними похідними, так й з частинними похідними, що надало можливість автоматизувати процедуру комп'ютерного моделювання пошуку розв'язків відповідних крайових задач.

4. Розроблено математичну модель процесу у зоні плавлення екструдера, що враховує фазовий перехід тверда суміш (``пробка'') -- рідинна фаза в'язкопружних полімерних матеріалів із урахуванням дисипативних явищ у

розплаві, що надає можливість визначити швидкість руху межі фазового переходу і визначити, таким чином, оптимальний режим нагріву корпусу екструдера для визначення конструктивних значень довжини зон завантаження і плавлення.

5. Виконано математичне моделювання процесу індукційного нагріву корпусу екструдера із урахуванням променистого випромінювання на межі із зовнішньою поверхнею корпусу, що надало можливість удосконалити процес теплопередачі від індуктора до корпусу екструдера.

6. Отримано розв'язок крайової задачі, що описує процеси нагрівання полімерної суміші у зоні завантаження із урахуванням залежності коефіцієнту теплоємності від температури, що надало можливість отримати оптимальні значення температурного поля суміші і визначити відповідні значення внутрішньої теплової енергії індуктора нагріву корпусу у зоні завантаження.

7. Отримані значення температурного поля суміші за значень, близьких до температури плавлення у зоні завантаження є критерій, що визначає довжину цієї зони, де починається зона плавлення.

8. Виконано математичне і комп'ютерне моделювання процесів у зоні плавлення полімеру із урахуванням фазового переходу тверда суміш -- розплав, що надало можливість визначити швидкість руху межі фазового переходу і порівняти її значення із швидкістю обертального руху черв'яка шнека з метою визначення оптимальних значень внутрішньої теплової енергії індуктора нагріву корпусу у зоні плавлення.

9. Отримано розв'язок комбінованої задачі теплоперенесення у розплаві полімеру і кінетичної задачі гомогенізації і кристалізації розплаву у зоні дозування екструдера, що надало можливість визначати оптимальні значення геометричних розмірів цієї зони.

10. На грунті виконаних досліджень у зонах завантаження, плавлення та дозування запропоновано алгоритм управління потужністю індуктора, що

нагріває корпус екструдера у зазначених зонах.

11. Точність результатів, що отримано за допомогою запропонованих алгоритмів вирішення нелінійних крайових задач для рівнянь масо- і теплоперенесення оцінюється значеннями норми у гільбертовому просторі на послідовних ітераціях реалізації цих алгоритмів.

12. Вирішення комплексів задач, що розглянуто у дисертаційній роботі надає можливість оптимізувати існуюче екструзійне обладнання і проектувати екструзійні лінії для нових виробів із полімерних матеріалів (виготовлення плівок тощо).

13. Результати роботи впроваджено у практику експлуатації та проектування екструзійного перероблення термопластичних полімерів. На заводі "Південкабель" у м. Харкові впроваджено методики розрахунків геометричних параметрів екструзійного обладнання, алгоритмічне забезпечення розв'язання крайових задач для систем нелінійних рівнянь типу рівнянь Нав'є--Стокса. На ТОВ "Прикарпатгаз" (м. вано-Франківськ), а також на ТОВ "Укрекококосалт" (м. Київ) впроваджено алгоритмічне забезпечення методики розрахунків концентрації забруднюючих речовин у повітряних потоках. Результати досліджень використовуються у навчальному процесі кафедри біомедичної кібернетики факультету біомедичної інженерії та кафедри теоретичної електротехніки факультету електро-енерготехніки та автоматики НТУУ ``Київський політехнічний інститут імені Ігоря Сікорського".

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андерсон Д., Таннехилл Дж., Флетчер Р. Вычислительная гидромеханика и теплообмен: В 2-х томах: Т. 2: Пер. с англ. Москва : Мир, 1990. 392 с.

2. Арнольд В. И. Цепные дроби. Москва, 2001. 40 с.

3. Базаров А.А., Данилушкин А.И., Никитина Е.А. Моделирование и расчет внутренних источников тепла в трехфазном индукторе с вращающимся магнитным полем *Вестник Самарского государственного технического университета. Сер. Технические науки*. 2009. Вып. 2(24). С. 120–127.

4. Басов Н. И., Казанков Ю. В., Любартович В. А. Расчет и конструирование оборудования для производства и переработки пластмасс. Москва : Химия, 1986. 488 с.

5. Бачурина М. В., Казаков А.В., Труфанова Н.М. Математическое моделирование процесса стратифицированного течения расплавов полимеров в осесимметричной постановке. *Вестник ПНИПУ*, 2014, № 2. С. 102--124.

6. Бачурина М.В., Щербинин А.Г. Моделирование процессов течения и теплообмена расплава полимера в зоне дозирования одношнекового екструдера. *Пермский национальный исследовательский политехнический университет*. 2012. 6 с.

7. Бейкер Дж., Грейвс-Моррис П. Аппроксимация Паде. Москва, 1986. 502 с.

8. Белоносов С. М., Овсиенко В. Г., Карачун В. Я. Применение интегральных представлений к решениям задач теплопроводности и динамики вязкой жидкости. Киев, 1989. 163 с.

9. Бенькович Е. С., Колесов Ю. Б., Сениченков Ю. Б. Практическое моделирование сложных динамических систем. Санкт-Петербург, 2001. 401 с

10. Бессонова М. П. Расчет течения степенной жидкости в одношнековом

экструдере/М.П. Бессонова, М. А. Пономарева, В.А. Якутенок. *Вестник Томского гос. ун-та, математика и механика*. 2017. №49. С. 81--93.

11. Беляева Н.А. Математическое моделирование деформирования вязкоупругих структурированных полимерных (композиционных) систем. Автор. докт. дисс. Челябинск, 2008. 35 с.

12. Бідюк П.І., Меняйленко О. С., Половцев О. В. Методи прогнозування. Луганськ : «Альма– матер», 2008. 305 с.

13. Бовсуновська К.С., Зеленський К.Х., Прийомов С.Г. Математичне моделювання закручених потоків у циклонних камерах. *Вісник Університету ``Україна", серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2011, №2. С. 43--51.

14. Бовсуновська К.С., Зеленський К.Х. Математичне моделювання аеродинамічних процесів у циклонних камерах. Доповіді міжнародної науково-практичної конференції ISDMCI, м. Євпаторія, 2011.

15. Бовсуновська К.С., Зеленський К.Х. Комп'ютерне моделювання руху твердих домішків у циклонних камерах. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. Луцьк, 2013, №13, с. 71--78

16. Бойков И. В., Кривулин Н. П. Аналитические и численные методы идентификации динамических систем. Пенза, 2016. 396 с.

17. Болховітін В.М., Зеленський К.Х. Моделювання двохфазних течій у турбулентному потоці. *Матеріали X Всеукраїнської науково-практичної конференції ``Комп'ютерні технології: наука і освіта"*, Україна, Київ, 2017. С. 75--78.

18. Болховітін В.М., Зеленський К.Х. Математичне моделювання температурних режимів полімерного покриття кабельних виробів. *Вісник Університету ``Україна". серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2017, №1 (19). С. 50--53.

19. Булавацький В. М., Кривонос Ю. Г., Скопецький В. В. Некласичні

математичні моделі процесів тепло- та масопереносу. Київ, 2005. 284 с.

20. Бутаков Г.О., Зеленський К.Х., Ігнатенко В.М. Побудова моделі формування геометрії шва при зварюванні виробів неплавким електродом. *Міжв. наук.-техн. зб. ``Адаптивні системи автоматичного управління''*. 1998, №1 (21). С. 69--76.

21. Бутковский А. Г. Методы управления системами с распределенными параметрами. Москва, 1975. 568 с.

22. Бутковский А. Г. Структурная теория распределенных систем. Москва, 1977. 320 с.

23. Васидзу К. Вариационные методы в теории упругости и пластичности. Москва : Мир, 1987. 468 с.

24. Васильєв В. В., Сімак Л. О., Зеленков О. А. Аналіз та математичне моделювання динамічних систем на базі некласичних операційних числень. Київ, 2006. 184 с.

25. Васильев И.В. Совершенствование индукционного нагревательного комплекса для термообработки вязких жидкостей. Дис...канд. техн. наук, 2018, Самара. 138 с.

26. Верлань А. Ф., Дячук А. А., Палагин В. В. Методы математической редукции моделей динамических систем. Киев. Наукова думка, 2019. 311 с.

27. Верлань А. А., Іванюк В. А. Спрощення математичних моделей об’єктів з розподіленими параметрами на основі методу розщеплення. *Інформатика та математичні методи в моделюванні*. 2017. Т. 7, № 4. С. 285–290.

28. Верлань А. Ф., Федорчук В. А., Іванюк В. А. Інтегральні моделі нестаціонарних задач теплопровідності на основі методу теплових потенціалів. *Математичне та комп’ютерне моделювання. Серія: технічні науки*: зб. наук. праць. Кам’янець-Подільський, 2019. Вип. 19. С. 24–30.

29. Верлань А. Ф., Федорчук В. А., Іванюк В. А. Комп’ютерне

моделювання в задачах динаміки електромеханічних систем. Кам'янець-Подільський, 2010. 204 с.

30. Виноградов Г. В., Малкин А. Я. Реология полимеров. Москва : Химия, 1997. 438 с.

31. Волин Ю.М., Островский Г.М. Многокритериальная оптимизация технологических процессов в условиях неопределенности. *Автоматика и телемеханика*. 2007. № 3. С. 165–180.

32. Волков Н. В. Функциональные ряды в задачах динамики автоматизированных систем. Москва, 2001. 100 с.

33. Галай В.М., Зеленський К.Х., Сільвестров А.М. Теорія оптимальних систем автоматичного керування технологічними процесами. Навч. пос. Полтава, 2009. 152 с.

34. Гаращенко Ф. Г., Волошин О. Ф., Кириченко М. Ф. та ін. Розвиток методів і технологій моделювання та оптимізації складних систем: монографія. Київ, 2009. 667 с.

35. Гребенщиков Е.А., Рябов Ю.А. Конструктивные методы анализа нелинейных систем. Москва, Наука, 1979. 430 с.

36. Данилушкин А.И., Данилушкин В.А., Васильев И.В. Экономичная система электронагрева экструдера в линии производства пенополистирольных плит. *Градостроительство и архитектура*, 2017, Т.7, №2. С. 125--133.

37. Дейнека В. С., Сергиенко И. В. Анализ многокомпонентных распределенных систем и оптимальное управление. Киев: Наук. думка, 2007. 701 с.

38. Демиденко Н. Д., Потапов В. И., Шокин Ю. И. Моделирование и оптимизация систем с распределенными параметрами. Новосибирск, 2006. 551 с.

39. Демидович В.Б. Применение индукционного нагрева в металлургической промышленности. СПб., 2003.

40. Дёч Г. Руководство к практическому применению преобразования Лапласа и Z-преобразования. Москва, 1971. 288 с.

41. Дикусар В.В., Вуйтович М. Оптимальное управление процессом электронагрева. **Вестник РУДН, серия: Математика, информатика, физика**, №2, 2010. С. 120--123.

42. Евдокимов В. В. Оборудование и механизация производства полимерных пленочных материалов и искусственных кож. М.: Наука, 1992. 310 с.

43. Ершов Н.М., Труфанова Н.М., Субботин Е.В. Исследование влияния охлаждения шнека на процесс экструзии. 2012,

44. Жученко О. А., Цапар В. С. Метод спрощення математичних моделей об’єктів керування із розподіленими параметрами. Автоматизація технологічних і бізнес-процесів. Херсон, 2015. Вип. 7. 2015. С. 15–25.

45. Забара С.С., Філімонова Н.Б., Зеленський К.Х. Метод виділення інваріантних ознак сигналів. it Доповіді АН України, К.: Наукова думка, 2009, №2, С. 49--55.

46. Зеленська Н.К., Зеленський К.Х. Апроксимація циліндричних функцій дробово-раціональними виразами. *Вісник Університету ``Україна''. серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2016, №2(18). С. 17--23.

47. Зеленский К.Х. Разработка и исследование методов и средств эквивалентного упрощения математического описания сложных динамических объектов: автореф. дис. ... канд. техн. наук; 05.13.02, 1975. 28 с.

48. Зеленский К.Х. Математическое обеспечение задач структурной идентификации ОРП. *Вестник КПИ, серия автоматики и приборостроения*, Киев, 1975, №12. С. 148--150.

49. Зеленський К.Х. Числово-аналітичний метод розв'язання нелінійних крайових задач математичної фізики. Матеріали IV міжнар. наук. конф.

ISDMCI' 2009. Євпаторія. 2009.

50. Зеленський К.Х. Числово-аналітичний метод розв'язання просторово-часових задач із рухомими межами. *Міжв. наук.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2009, №14 (34). С. 107--117.

51. Зеленський К.Х. Вища математика, ч.2, Навч. пос. К.: ВДК ``Україна", 2007. 25 авт.арк.

52. Зеленський К.Х. Математичне програмування. Навч. пос. К.: ВДК ``Україна", 2007. 20 авт.арк.

53. Зеленський К.Х. Ітераційний метод розв'язання нелінійних крайових задач. *Наукові нотатки Луцького національного університету*, вип. 26, т, 2, 2010. С. 49—55.

54. Зеленский К.Х., Аленкин-Плешко В.М. Об идентификации нелинейных динамических моделей с помощью упрощенных обобщенных моделей. *Міжв. наук.-техн. зб. ``Адаптивные системы автоматического управления''*. 1975, №3. С.3--8.

55. Зеленский К.Х., Краскевич В.Е., Гречко В.И. Численные методы в инженерных исследованиях. Навч. пос. К.: Вища школа, 1986. 240 с.

56. Зеленский К.Х. Оценка геометрии сварочной ванны при сварке концентрированными потоками энергии. Доклады 2-й межд. конф. по ЭЛТ. Варна, 1988, С.105--112.

57. Зеленский К.Х., Мелехин Ю.А. Математическое моделирование магнитного поля в линейном индукторном двигателе. Всес. конф. ``Математическое и имитационное моделирование в системах проектирования и управленияє". Тез. докл. Чернигов, 1990, С.250--253.

58. Зеленский К.Х. Управление плотностью тока в фокальном пятне при ЭЛС. Доклады 2-й межд. конф. по ЭЛ. Варна, 1988, С. 97--104.

59. Зеленский К.Х. Адаптивное управление процессами сварки неплавящимся электродом. Труды X межд. симпозиума ``Welding-90", 1990.

Брно. С. 37--42.

60. Зеленський К.Х. Математическое моделирование температурного поля сварочной ванны. Матеріали V Міжнародної науково-технічної конференції. Т.2. Аерокосмічні системи моніторингу та керування. Київ, 2003. С. 24.48--24.57.

61. Зеленский К.Х., Бутаков Г.А. Управление сваркой плавящимся электродом. Матеріали V Міжнародної науково-технічної конференції. Т.2. Аерокосмічні системи моніторингу та керування. Київ, 2003. С. 24.171--24.175.

62. Зеленський К.Х., Кеменяш Ю.М. Комп'ютерне моделювання процесів демпфування рідини у рухомих ємностях. *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2005, №09(29). С. 73—79.

63. Зеленський К.Х., Семанишин Л.М. Математичне моделювання вмісту розчиненого кисню та біохімічного вживання кисню у поверхневих водах. *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2007, №11(31). С. 112--117.

64. Зеленський К.Х., Ліщина В.О. Моделювання динаміки обмеженого обсягу рідини із вільною поверхнею. *ІПМС* Зб. наукових праць, вип.38, 2007. С. 135-141.

65. Зеленський К.Х., Ігнатенко В.М., Коц О.П. Компьютерные методы прикладной математики. Реализация. К.: Академперіодика, 2001. 280 с.

66. Зеленський К.Х., Ігнатенко В.М., Коц О.П. Комп'ютерні методи прикладної математики. К.: Академперіодика, 2002. 480 с.

67. Зеленський К.Х., Ігнатенко В.М. Оптимальне керування системами із запізненнями. *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2008, №12(32). С. 93—97.

68. Зеленський К.Х. Визначення геометрії зварного шва при зварюванні КДЕ. *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2009, №13(33). С. 118--125.

69. Зеленський К.Х. Математичне моделювання теплових процесів при вирощуванні монокристалів. *Матеріали 4-ї Всеукраїнської науково-практичної конференції Комп'ютерні технології: наука і освіта*. Луцьк, 2009. с. 53--55.

70. Зеленський К.Х., Кіт Г.В., Філімонова Н.В. Побудова повної множини інваріантних ознак сигналів. *Восточно-Европейский журнал передовых технологий*, 2007, №6. С. 29--35.

71. Зеленський К.Х., Семанишин Л.М. Оцінка якості поверхневих вод. *Восточно - Европейский журнал передовых технологий*, 2008, №2/3. С. 56-62.

72. Зеленский К.Х., Ліщина В.О. Математичне моделювання аеродинаміки верхових лісових пожеж. *Наукові нотатки. Міжвузовський збірник*, Луцьк, 2010. Вип. 27. С. 110--115.

73. Зеленський К.Х., Ліщина В.О., Ваврук Є. Математичне моделювання низинних лісових пожеж. *Вісник ЛПУ, Комп'ютерні науки та інформаційні технології*, №638, 2009. С. 95--99.

74. Зеленський К.Х., Ігнатенко В.М., Бовсуновська К.С. Комп'ютерне моделювання динаіки повітряних потоків у циклонних камерах *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління''*. 2012, №21(41). С. 132--145.

75. Зеленський К.Х., Павлов В.А, Бовсуновська К.С. Математичні методи оптимізації і прийняття рішень. Навч. пос. К.: ВДК ``Україна", 2013. 31 авт.арк.

76. Зеленський К.Х., Ігнатенко В.М., Поліновський В.В. Комп'ютерні методи обробки сигналів та зображень. Навч. пос. К.: ВДК ``Україна", 2013. 31 авт.арк.

77. Зеленський К.Х. , Кіт Г.В., Чумаченко О.І. Комп'ютерне моделювання систем. К.: ВДК ``Україна", 2014. 27,5 авт.арк.

78. Зеленский К.Х., Бовсуновская К.С. Математическое моделирование конвективно- диффузионных процессов в циклнных камерах. *Актуальные проблемы гуманитарных и естественных наук, журнал научных публикаций*,

Москва, 2014. С. 44--49.

79. Зеленский К.Х., Настенко Е.А. Математическое моделирование динамики левого желудочка. *Вісник Університету ``Україна". серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2016, №1 (18). С. 27--38.

80. Зеленський К.Х., Болховітін В.М. До визначення концентрації домішків у двофазних циклонних пристроях. *Вісник Університету ``Україна". серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2016, №2(18). С. 32--38.

81. Зеленський К.Х., Ігнатенко В.М., Стєнін О.А. Структурна властивість оптимальних за витратами палива процесів управління у динамічних системах. *Міжв. науч.-техн. зб. ``Адаптивні системи автоматичного управління"*. 2017, Вип. 1(42), Дніпро. С. 97--103.

82. Зеленский К.Х., Бурлаков М.В. Моделювання процесів плавлення полімерів у гвинтовому каналі шнека. *Вісник Університету ``Україна". серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2018, №1 (19). С. 166--174

83. Зеленский К.Х., Болховітін В.М. Математичне моделювання процесу охолодження розплавів полімерів. Міжнародна науково- практична конференція «Інформаційні технології в освіті, науці і виробництві, Луцьк, 23-25 травня 2019 С. 44—-47.

84. Зеленский К.Х., Бовсуновская К.С., Болховітін В. М. Алгоритмічне забезпечення розв'язання нелінійних крайових задач тепломасопереносу. *Modern engineering and innovative technologies*. 2021, Karlsruhe, Germany, Issue 15/Part 1, February. P. 6--12. DOI 10.30890/2567-5273.

85. Зинатуллин Р.Р., Труфанова Н.М.. Численное моделирование технологических напряжений при изготовлении пластмассовой изоляции провода. *Вычислительная механика сплошных сред*. 2009. Т.2, №1. С.38--53.

86. Іванюк В. А. Ланцюгово-дробова апроксимація ірраціональних та трансцендентних передатних функцій об’єктів з розподіленими параметрами. *Математичне та комп'ютерне моделювання. Серія: Технічні науки*: зб. наук. праць. Кам’янець-Подільський, 2008. Вип. 1. С. 75–85.

87. Іванюк В. А. Метод відновлення сигналів на вході нелінійних динамічних об’єктів з розподіленими параметрами. *Моделювання*. Київ, 2018. С. 154–157.

88. Іванюк В. А., Федорчук В. А. Адаптивний метод ідентифікації моделей нелінійних динамічних систем інтегральними рядами Вольтерри. *Електронне моделювання*, 2019. Т. 41, № 3. С. 33–42.

89. Іванюк В. А., Костьян Н. Л. Інтегральний метод розв’язування диференціальних рівнянь при моделюванні об’єктів із розподіленими параметрами. *Математичне та комп’ютерне моделювання. Серія: Фізико-математичні науки : зб. наук. праць*. Кам’янець-Подільський, 2018. Вип. 18. С. 78–85.

90. Іванюк В. А., Федорчук В. А. Адаптивний метод ідентифікації моделей нелінійних динамічних систем інтегральними рядами Вольтерри. *Електронне моделювання*, 2019. Т. 41, № 3. С. 33–42.

91. Карташов Э. М., Кудинов В. А., Калашников В. В. Теория тепломассопереноса: решение задач для многослойных конструкций. Москва, 2018. 435 с.

92. Кіселівський Ф.М., Бутаков Г.О., Зеленський К.Х. Control of adaptive welding robotic stations. *V IFAC, Suzdal*, USSR, 1986. pp.373--379.

93. Кіселівський Ф.М., Бутаков Г.О., Зеленський К.Х. Adaptive control of velding robots. *2-d Intern. Conf. Developments in Automated and robotic Welding*, Cambridge: The welding Institute, 1987.-- pp.21-1--21-11.

94. Ким В. С. Теория и практика экструзии полимеров. Москва : Химия, КолосС. 2005. 568 с.

95. Ковалюк Д. О., Москвіна С. М. Моделювання теплотехнологічних об’єктів з розподіленими параметрами: монографія. Вінниця, 2010. 182 с.

96. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Структурная идентификация в частотной области нестационарних объектов с распределенными параметрами *Межв. науч.- техн. сб. ``Адаптивные системы автоматического управления''*. 1976, №4. С.19--29.

97. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация объектов с распределенными параметрами методами теории чувствительности. *Труды всес. шк.- семинара Чувствительность систем управления* Т.2. м. Владивосток, 1976. С. 95--109.

98. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация объектов с распределенными параметрами в режиме нормального функционирования. *Изв. Вузов, Приборостроение*. Т.XX, №10, 1977. С. 35--47.

99. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация нестационарных объектов с распределенными параметрами в частотной области. *Труды IV-го симпозиума ИФАК ``Идентификация и оценка параметров систем''*. Тбилиси, ``Мецниереба'', 1976, Т.3--С. 103--116.

100. Костюк В.И., Краскевич В.Е., Зеленский К.Х. Идентификация нелинейных объектов с распределенными параметрами. *Изв. АН СССР, серия техн. кибернетика*, 1978, №2. С. 27-35.

101. Кострицький В. В., Кириченко Ю. О. Математична модель напружено– деформованого стану рулонованого матеріалу. *Вісник КНУХТ*. 2004. № 3. 123 с.

102. Кретов Д. И. Система оптимального управления процессом охлаждения кабельной изоляции. Авторефер. дисс. ...канд.техн. наук. Самара, 2007. 23 с.

103. Крылов А.Н. Исследование и разработка системы косвенного индукционного нагрева при производстве пенополистирольных плит. Автор.

дис ... канд. техн. наук, Самара, 2005. 18 с.

104. Кудинов И.В. Математическое моделирование процессов теплопроводности и гидродинамики численно-аналитическими методами на основе использования дополнительных граничных условий. Автор...дисс. к.т.н., 2011. 22 с.

105. Куликова Т.Г., Труфанов Н.А Численное решение краевой задачи термомеханики для кристаллизующегося вязкоупругого полимера. *Выч. механика сплошных сред*. 2008. Т.1, №2. С. 38-52.

106. Кушнир М. С., Сивецкий В. И., Коваленко К. Г. и др. Моделирование процесса плавления полимеров в каналах червячных экструдеров *Химическое и нефтегазовое машиностроение*. – 2013. – № 11. – С. 29-33.

107. Ладиков-Роев Ю. П., Черемных О. К. Математические модели сплошных сред. Київ : Наукова думка, 2010. 551 с.

108. Литвинов В. Г. Движение нелинейно-вязкой жидкости. Москва : Наука, 1982. 374 с.

109. Малкин А. Я. Современное состояние реологии полимеров: достижения и проблемы *Высокомол. соединения*, 2009. Т. 51. С.106–136.

110. Малыгин Е. Н., Карпушкин С.В., Карпов С.В. Моделирование и расчет процессов индукционного нагрева прессового оборудования при производстве резинотехнических изделий *Наука и образование. Научное издание МГТУ им. Н. Э. Баумана*, 2013. №.3. С. 85--102.

111. Мартыненко Н. А., Пустыльников Л. М. Конечные интегральные преобразования и их применение к исследованию систем с распределенными параметрами. Москва, 1986. 303 с.

112. Марчук Г. И. Методы вычислительной математики. Учебное пособие. Москва, Главная редакция физ.-мат. литературы, 1989. 608 с.

112. Мелешко В. В., Краснопольская Т. С. Смешивание вязких жидкостей *Нелинейная динамика*. 2005. Т. 1. № 1. С. 69–109.

113. Мікульонок І. О., Сокольський О. Л., Соколенко В. В. Полімерна ізоляція кабельних виробів. Напрями забезпечення якості *Хімічна промисло вість України*. 2015. № 2(127). С. 7–15

114. Митрошин В.Н. Регулирование давления расплава полимера в зоне дозирования одночервячного экструдера при пульсирующем градиенте давления. *Вестник Самар. гос. техн. ун-та. Сер. Технические науки: Научный журнал*. Самара: СамГТУ, 2011, № 1(29). С. 39 – 44

115. Митрошин В.Н., Митрошин Ю.В. Автоматизация процесса наложения изоляции при непрерывном производстве проводных кабелей связи. *Информационные, измерительные и управляющие системы (ИИУС-2010)*. Материалы Международной научно-технической конференции (Самара, 17-21 мая 2010 г.). Самара: Самар. гос. техн. ун-т, 2010. С. 36 – 40.

116. Митрошин В. Н. Структурное моделирование процесса охлаждения изолированной кабельной жилы при ее изготовлении на экструзионной линии. *Вестн. Самар. гос. техн. ун-та. Сер.: «Технические науки»*, 2006, № 40. С. 22 - 33.

117. Митрошин В.Н., Колпащиков С.А. Автоматизация процесса наложения полимерной изоляции при изготовлении проводных кабелей связи. *Вестник Самарского государственного университета. Сер. Технические науки*, 2018. № 3(59). С. 28–40.

118. Митрошин В.Н. Автоматизация технологических процессов производства кабелей связи. М.:Машиностроение-1, 2006. 140 с.

119. Митрошин В.Н. Автоматическое управление объектами с распределенными параметрами в технологических процессах изолирования кабелей связи. М.: Машиностроение, 2007. 184 с.

120. Митрошин В.Н., Кулешова Д.И. Разработка системы управления процессом изолирования кабелей связи, обеспечивающей достижение требуемого эксплуатационного качества продукции. *Вестник Самарского*

государственного университета. Сер. Технические науки, 2015. № 1(45). С. 71–77.

121. Митрошин В. Н. Математичесское моделирование процесса охлаждения изолированной кабельной жилы при ее изготовлении на экструзионной линии как объекта управления с рапределенными параметрами, *Вестник Самарского государственного университета. Сер. Технические науки*, 2005. С. 122--128.

122. Митрошин В. Н., Нечаев А.С. Модель системы распределенного управления температурой расплава полимера сс учетом его реологических особенностей при наложении кабельной изоляции методом экструзии. *Вести Самар. гос. техн. ун-та. Серия физико-мататматические науки*, 2006, № 43. С. 104--110.

123. Митрошин В. Н. Описание одночервячного экструдера как объекта управления с распределенными параметрами. *Известия Вузов. Поволжский регион*, 2007, № 1. С. 162--173.

124. Островерхов М.Я., Сільвестров А.М., Зеленський К.Х. Методи дослідження електротехнічних систем і комплексів. Монографія, Київ, Талком, 2019. 300 с.

125. Панов А. К., Апасов А. Р. Гидродинамика потоков аномально-вязких полимерных систем в формующих каналах. Уфа : Изд-во УГНТУ, 1994. 260 с.

126. Первадчук В. П., Труфанова Н. М., Янков В. И. Математическая модель плавления полимерных материалов в экструдерах. *Химические волокна*. 1984. № 3. С. 51–53.

127. Первадчук В. П., Труфанова П. М., Янков В. И. Математическая модель и численный анализ процессов теплообмена при плавлении полимеров в пластицирующих экструдерах. *Инж.–физ. журнал*. 1985. Т. 48, № 1. С. 75–80.

128. Плешивцева Ю.Э. Последовательная параметризация управляющих воздействий и полубесконечная оптимизация алгоритмов управления

технологическими объектами с распределенными параметрами. Дис. … д-ра техн. наук. СамГТУ. Самара, 2009. 416 с.

129. Полосин А. Н., Чистякова Т. Б. Система моделирования процессов экструзии и формообразования полимерных материалов для управления качеством руканых пленок. *Компьютерные исследования и моделирование*, 2004, т.6. №1. С.117--158.

130. Пристінні ефекти в процесах переробки полімерних матеріалів /В. І. Сівецький, О. С. Сахаров, О. Л. Сокольський, Д. Д. Рябінін. Київ: НТУУ КПІ, 2009. 140 с.

131. Пупков К. А., Капалин В. И., Ющенко А. С. Функциональные ряды в теории нелинейных систем. Москва, 1976. 448 с.

132. Пупков К. А., Шмыкова Н. А. Анализ и расчет нелинейных систем с помощью функциональных степенных рядов. Москва, 1989. 150 с.

133. Рапопорт Э.Я. Оптимизация процессов индукционного нагрева металла. Москва. Металлургия, 1993. 279 с.

134. Рапопорт Э.Я. Структурно-параметрический синтез систем автоматического управления с распределенными параметрами. *Изв. РАН. Теория и системы управления*, 2006. № 4. С. 47-60.

135. Рапопорт Э.Я. Структурное моделирование объектов и систем управления с распределенными параметрами. Москва, Высшая школа, 2003.

136. Рапопорт Э.Я., Плешивцева Ю.Э.. Оптимальное управление температурными режимами индукционного нагрева. Москва, Наука, 2012. 309 с.

137. Рапопорт Э.Я., Митрошин В.Н., Кретов Л.И. Оптимальное управление процессом охлаждения полимерной кабельной изоляции при ее наложении на экструзионной линии. *Вести Самар. гос. техн. ун-та. Серия физико- мататматические науки*, 2006, № 43. С. 146--153.

138. Раувендаль К. Экструзия полимеров / Пер. с англ. под ред. А.Я. Малкина. СПб.: Профессия,2008. 768 с.

139. Самарский А. А., Вабищевич П. Н. Вычислительная теплопередача. Москва, 2003. 784 с.

140. Самойленко А.М., Ронто Н.Н. Численно-аналитические методы исследования решений краевых задач. Киев, Наукова думка, 1986. 222 с.

141. Самойленко А.М., Ткач Б.П. Числленно-аналитические методы в теории периодических решений уравнений с частными производными. Киев, Наукова думка, 1992. 208 с.

142. Сахаров О. С., Баженов В. А., Цыхановский В. К. Моментная схема метода конечных элементов в задачах нелинейной механики сплошной среды *Прикладная механика*, 2002. Т. 38. №6. С. 24–63.

143. Сахаров О. С., Сівецький В. І., Сокольський О. Л. Моделювання процесів плавлення та гомогенізації полімерних композицій в черв'ячному устаткуванні: монографія. Київ : ВП «Едельвейс», 2012. 120 с.

144. Сергиенко И. В., Скопецкий В. В., Стоян В. А., Благовещенская Т. Ю., Богаенко В. А. О программно-аналитическом моделировании задач динамики систем с распределенными параметрами. *Кибернетика и системный анализ*, 2005. 41, № 2. С. 35–55.

145. Скачков В. В., Торнер Р. В., Стунгур Ю. В., Реутов С. В. Моделирование и оптимизация экструзии полимеров. Ленинград : Химия, 1984. 152 с.

146. Скопецький, В. В., Стоян В. А., Зваридчук В. Б. Математичне моделювання динаміки розподілених просторово-часових процесів. Київ, 2008. 316 с

147. Скульский О.И. Численное моделирование одночервячных экструдеров. *Пластические массы*, 1997. №8. С. 39–44.

148. Слесаренко А.П. Аналитические, численные и аналоговые методы в задачах теплопродности. Киев, Наукова думка, 1977. С. 28--38.

149. Сорокин А.Г., Горбачевский Н.И., Мифтабова Л.Х. Методы

моделирования электромагнитных и тепловых полей системы индукционного нагрева длятехнологических комплексовпроизводства пластмассы.

150. Стоян В. А. Математичне моделювання лінійних, квазілінійних і нелінійних динамічних систем. Київ, ВПЦ «Київський університет». 2011, 319 с.

151. Субботин Е. В. Пространственные неизотермические течения в рабочих и охлаждающих каналах пластицирующего экструдера. Автореф. дис.... канд. техн. наук, 2013. 17 с.

152. Субботин Е. В., Щербинин А.Г., Труфанова Н.М. Численное исследование процессов течения полимеров в условиях фазового перехода в винтовых каналах экструдеров при производстве пластмассовой изоляциии *Известия Томского полит. ун-та*, 2002. Т. 320. №4. С.171--177.

153. Таланчук П.М., Зеленський К.Х., Болховітін В.М. Моделювання процесу ізоляційного покриття. *Вісник Університету ``Україна''. серія : інформатика, обчислювальна техніка та кібернетика*, Київ, 2016, №1 (18). С. 5--13.

154. Торнер Р.В. Теоретические основы переработки полимеров. М.: Химия, 1977. 464 с.

155. Федорчук В. А., Іванюк В. А., Бойко Ю. Д. Алгоритм приближения передаточных функций цепными дробями. *Электронное моделирование*, 2007. Т. 29. № 3. С. 93–100.

156. Федоткин И. М., Бурляй И. Ю., Рюмшин Н. А., Бурляй Ю. И. Математическое моделирование технологических процессов: Тепловые процессы, плавление, замораживание, теплопроводность, регулярный режим. Київ, 2004. 388 с.

157. Флетчер К. Вычислительные методы в динамике жидкостей: В 2-х томах: Т. 2: Пер. с англ. Москва : Мир, 1991. 552 с.

158. Хованский А.А. Приложение цепных дробей и их обобщений к вопросам приближенного анализа. Москва, 1956. 206 с.

159. Черняев В.В. Математическое моделирование влияния геометрических параметров шнека на процессы тепломассопереноса. *Пермский исслед. политех. ун-т*. С.85--92.

160. Чостковский Б.К. Методы и системы оптимального управления процессами производства кабелей связи. М.: Машиностроение, 2009. 190 с.

161. Швецов Г.А., Алимова Д.У., Барышникова М.Д. Технология переработки пластических масс. М.: Химия, 1988. 512 с.

162. Шардаков И.Н., Труфанов Н.А. Моделирование термомеханики кристаллизующихся полимеров. *Изв. Тульского ГУ. Естественные науки*, 2008. Вып. 2. С. 117--123.

163. Шен М. Вязкоупругие релаксации в полимерах. Москва : Мир, 1974. 270 с.

164. Щербинин А. Г. Процессы движения и теплообмена нелинейных полимерных сред в условиях фазового перехода в каналах экструзионного оборудования. Авторефер. дис.... докт. техн. наук, 2006. 34 с.

165. Щербинин А.Г., Савченко В.Г. Исследование влияния геометрии шнека на характеристики пластицирующего экструдера. *Интеллектуальные системы в производстве*, 2010. №1(15). С. 198--206.

166. Abeykoon C. Single screw extrusion control: A comprehensive review and directions for improvements. *Control Engineering Practice*, vol. 51, 2016, p. 69–80.

167. Abeykoon C., Li k., Mcafee M. A new model based approach for the prediction, optimisation of thermal homogeneity in single screw extrusion. *Control Engineering Practice*, vol. 19, no 8, 2011, p. 862–874.

168. Abeykoon C., Kelly A., Brown E. The effect of materials, process settings, screw geometry on energy consumption and melt temperature in single screw extrusion. *Applied Energy*, vol. 180, 2016, p. 880–894.

169. Abeykoon C., Martin P., Kelly A. A review, evaluation of melt temperature sensors for polymer extrusion. *Sensors and actuators A: Physical*, vol.

182, 2012, p. 16–27.

170. Adams R. Sobolev Spaces. New York: Academic Press (1975) 268 pp.

171. Agassant J., Avenas P., Carreau, P. Polymer Processing. Principles and Modelling, 2nd ed.; Carl Hanser Verlag: Munich, Germany, 2017

172. Aigner M., Praher B.,Kneidinger C. Verifying the Melting Behavior in Single-Screw Plasticization Units Using a Novel Simulation Model and Experimental Method. *Int. Polym. Proc.* 2014, 29, P. 624–634

173. Alexiades A., Solomon A. Mathematical Modeling of Melting and Freezing Processes, Hemisphere Publ. Co., Washington DC, 1993.

174. Altinkaynak A., Gupta M., Spalding M. Melting in a Single Screw Extruder: Experiments and 3D Finite Element Simulations. *Int. Polym. Proc.* 2011, 26, 182–196.

175. Amano O., Utsugi S. Temperature measurements of polymer melts in the heating barrel during injection molding. Part I. Temperature distribution along the screw axis in the reservoir. *Polymer Engineering & Science*, vol. 28, no 23, 1988, p. 1565–1571, Wiley Online Library.

176. Andreucci D, Fasano A., Primicerio M. Numerical simulation of polymer crystallization. *Mathematical Models and Methods in Applied Sciences*, November 2011. 11 p.

177. ANSYS Inc. ANSYS Polyflow, CDF for Extrusion, Forming and Molding; ANSYS Inc.: Canonsburg, PA, USA, 2019.

178. Bassett D. Principles of Polymer Morphology. Cambridge: Cambridge University Press (1981) 251 pp.

179. Bereaux Y., Moguedet M., Raoul X. Series Solutions for Viscous, Viscoelastic Fluids Flow in the Helical Rectangular Channel of an Extruder Screw. *J. Non- Newtonian Fluid Mech.*, vol. 123, 2004, p. 237.

180. Bird R., Armstrong R, O. Hassager O. Dynamics of polymer Liquids. *Fluid mechanics*, 1987, V. 1. 649 p.

181. Broyer E., Tadmor Z. Solids conveying in screw extruders part I: A modified isothermal model. *Polymer Engineering & Science*, vol. 12, no 1, 1972, p. 12–24, Wiley Online Library.

182. Brown E., Olley P., Coates P. In line melt temperature measurement during real time ultrasound monitoring of single screw extrusion. *Plastics, rubber, composites*, vol. 29, no 1, 2000, p. 3–13, Taylor & Francis.

183. Brown E., Kelly A. , Coates P. Melt temperature field measurement in single screw extrusion using thermocouple meshes. *Review of scientific instruments*, vol. 75, no 11, 2004, p. 4742–4748

184. Bu L., Agressly Y., Reax Y. Thermal homogeneity of plastication processes in single-screw extruders. *AIP Conference Proceedings*, vol. 1960 AIP Publishing, 2018, p. 120 -- 136.

185. Buick J., Cosgrove J. Numerical simulation of the flow field in the mixing section of a screw extruder by the lattice Boltzmann model. *Chemical engineering science*, vol. 61, no 10, 2006, p. 3323–3326.

186. Buick J. Lattice Boltzmann simulation of power-law fluid flow in the mixing section of a singlescrew extruder. *Chemical Engineering Science*, vol. 64, no 1, 2009, p. 52–58.

187. Bukkapatnam S., Clark B. Dynamic modeling, monitoring of contour crafting. An extrusion- based layered manufacturing process. *In Journal of manufacturing Science, Engineering*, 129(1), p. 135-142. ASME 2007

188. Bur A., Roth S., Spalding M. Temperature gradients in the channels of a single-screw extruder. *Polymer Engineering & Science*, vol. 44, no 11, 2004, p. 2148–2157, Wiley Online Library.

189. Burger M. Direct and Inverse Problems in Polymer Crystallization Processes. Linz: PhD-Thesis, University Linz (2000) 157 pp.

190. Burger M. Iterative regularization of an identification problem arising in polymer crystallization. *SIAM J. Num. Anal.* 39 (2001) 1029–1055.

191. Burger M., Capasso V. , Engl H. Inverse problems related to crystallization of polymers. *Inverse Problems* 15 (1999) p. 155–173.

192. Burger M., Capasso V., Salani C. Modelling multi-dimensional crystallization of polymers in interaction with heat transfer. *Nonlin. Anal. Real World Applic*. 3 (2002) p. 139–160.

193. Burger M., Capasso V., Eder G. Modelling crystallization of polymers in temperature fields. *Zeitschrift Angew. Math. Mech*. 82 (2002) p. 51–63.

194. Burger M., Capasso V. Mathematical modelling and simulation of non-isothermal crystallization of polymers. *Math. Models Methods Appl. Sci*. 11 (2001) p. 1029–1053.

195. Burger M. Iterative regularization of an identification problem arising in polymer crystallization. *SIAM J. Num. Anal*. 39 (2001) p. 1029–1055.

196. Campbell G., Spalding M. Analyzing and Troubleshooting Single- Screw Extruders; Carl Hanser Verlag: Munich, Germany, 2013

197. Capasso V., Engl H., Periaux J.(Eds.). Computational Mathematics Driven by Industrial Problems, Springer, Berlin/Heidelberg, 2000; Mathematical models for polymer crystallization processes, pp. 39–67.

198. Capasso V., Escobedo R., Salani C. Moving Bands and Moving Boundaries in an Hybrid Model for the Crystallization of Polymers, Free Boundary Problems *Theory and Applications*, vol. 147, ed. by P. Colli, C. Verdi, A. Visintin, (Birkhauser, 2004), pp. 75–86

199. Capasso V., Micheletti A. Local spherical contact distribution function and local mean densities for inhomogeneous random sets. *Stochastics Reports* 71 (2000) p. 51–67.

200. Capasso V., Salani C. Stochastic birth-and-growth processes modelling crystallization of polymers in a spatially heterogenous temperature field. *Nonlin. Anal. Real World Applic*. 1 (2000) p. 485–498.

201. Chang R., Lin K. The hybrid FEM/FDM computer model for analysis of

themetering section of a singlescrew extruder. *Polymer Engineering & Science*, vol. 35, no 22, 1995, p. 1748–1757, Wiley Online Library.

202. Chung T. J. Computational Fluid Dynamics. Cambridge University Press, Cambridge. 2002. P. 533–580.

203. Chatwin P., Allen C. Mathematical Models of Dispersion in Rivers, Estuaries. *Chemical Engineering Journal*, vol. 17, no 1, 1985, p. 119–149.

204. Chieuwvwlla R., Jaluria Y., Abib A. Numerical simulation of fluid flow, heat transfer in a single-screw extruder with different dies. *Polymer Engineering & Science*, vol. 35, no 3, 1995, p. 261–273, Wiley Online Library.

205. Cheroto S., Mikhailov M., Kakac S. Periodic laminar forced convection: solution via symbolic computation, integral transforms. *International journal of thermal sciences*, vol. 38, no 7, 1999, p. 613- 621.

206. Chung C. Extrusion of Polymers. Theory and Practice, 2nd ed.; Carl Hanser Verlag: Munich, Germany, 2010.

207. Cruz D. O. A., Pinho, F. T. Analysis of isothermal flow of a PhanThien–Tanner fluid in a simplified model of a single-screw extruder. *Journal of Non-Newtonian Fluid Mechanics*. 2012. V. 167-168. P. 95–105.

208. Deen W. Analysis of Transport Phenomena. Oxford University Press, 2013.

209. Diagne M., Shang P., Wang Z. Feedback Stabilization of a Food Extrusion Process Described by 1D PDEs Defined on Coupled Time- Varying Spatial Domains. *In Proceedings of the 13th IFAC Workshop on Time-Delay Systems*, vol. 48, no. 12, 2015, p. 51-56, IFAC, 2015.

210. Diagne M., Shang P., Wang Z. Feedback Stabilization for the Mass Balance Equations of an Extrusion Process. *In IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 760-765, 2016.

211. Drotman D., Diagne M., Krstic M. Control-Oriented Energy-Based Modeling of a Screw Extruder Used for 3D Printing. *Dynamic Systems, Control*

Conference, vol. 1001, p. 481-09. ASME, 2016.

212. Dubay R., Bell A. An experimental comparison of cooling time for cylindrical plastic components using heat conduction models in the nonconservative and conservative forms. *Polym. Eng. Sci.* 38 (7) (1998) 1048–1059.

213. Eder G. Crystallization kinetic equations incorporating surface and bulk nucleation. *Zeitschrift Angew. Math. Mech*. 76 (1996) S4, p. 489–492.

214. Eder G. Fundamentals of structure formation in crystallizing polymers. In: K. Hatada, T. Kitayama and O. Vogl (eds), Macromolecular Design of Polymeric Materials. New York: Marcel Dekker (1997) pp. 761–782.

215. Escobedo R., Fernandez L. Free boundary problems (FBP) and optimal control of axisymmetric polymer crystallization processes. *Computers and Mathematics with applications*. 68 (2014), 27--43.

216. Escobedo R., Fernandez L. Classical One-Phase Stefan Problems for Describing Polymer Crystallization Processes. *In SIAM Journal on Applied Mathematics*, p. 254-280. SIAM, 2013.

217. Escobedo R., Fernandez L. Optimal cooling strategies in polymer crystallization. *J. Math. Chem.* 50 (2012) p. 313–324.

218. Escobedo R., Capasso V. Reduction of a Mathematical Model for Polymer Crystallization. *Progress in Industrial Mathematics at ECMI* 2004, pp. 259–263

219. Escobedo R., Capasso V. Moving bands and moving boundaries with decreasing speed in polymer crystallization. *Math. Models Methods Appl. Sci.* 15 (3) (2005) p. 325–341.

220. Escobedo R., Fernandez L. Optimal control of chemical birth and growth processes in a deterministic model. *J Math Chem* (2010) 48, p. 118–127. DOI 10.1007 /s10910-009-9638-x

221. Esseghir M., Sernas V. On the measurements of the radial temperature distribution in an extruder channel. *Advances in Polymer Technology*: Journal of the Polymer Processing Institute, vol. 13, no 2, 1994, p. 133–140.

222. Fakoor-Pakdaman M., Ishen-Tadbir M. Unsteady laminar forced-convective tube flow under dynamic time-dependent heat flux. *Journal of Heat Transfer*, vol. 136, no 4, 2014, p. 041706, American Society of Mechanical Engineers.

223. Fang S., Chen L., Zhu F. Studies on the theory of single screw plasticating extrusion. Part II: Non-plug flow solid conveying. *Polymer Engineering & Science*, vol. 31, no 15, 1991, p. 1117–22.

224. Fasano A. Mathematical models in polymer processing, *Meccanica* 35 (2000) p. 163–198.

225. Ferras L. L., Nobrega J. M., Pinho F. T. Analytical solutions for Newtonian and inelastic non-Newtonian flows with wall slip // Journal of Non-Newtonian Fluid Mechanics. 2012. № 175. P. 76–88.

226. Fernandes C., Pontes A., Viana J. Modeling of Plasticating Injection Molding Experimental Assessment. *International Polymer Processing*, vol. 29, no 5, 2014, p. 558–569.

227. Fernandes C., Pontes A., Gaspar-Cunha A. Modeling, Optimization of the Injection-Molding Process: A Review. *Advances in Polymer Technology*, vol. 37, no 2, 2018, p. 429–449, Wiley Online Library.

228. Friedman L., Chrzan D. Scaling theory of the Hall-Petch relation for multilayers. *Phys. Rev. Lett*. 81 (1998) p. 2715–2719.

229. Friedman A., Velazquez A. A free boundary problem associated with crystallization of polymers in a temperature field. *Indiana Univ. Math. J.* 50 (2001) p. 1609–1649.

230. Geng F. Z., Cui M. G. Analytical Approximation to Solutions of Singularly Perturbed Boundary Value Problems. *Bulletin of the Malaysian Mathematical Sciences Society*, 2010. Vol. 33. № 2. P. 221–232.

231. Guillet M. S. Quantitative evaluation of extrudate swell from viscoelastic properties of polystyrene. *Rheologica Acta*. 1991. N 30. P. 540–548.

232. Han C. D., Lee K. Y., Wheeler N. C. An experimental study on plasticating single-screw extrusion. *Polymer Engineering & Science*, vol. 30, no 24, 1990, p. 1557–1567, Wiley Online Library.

233. Han C. D., Lee K. Y., Wheeler N. C. Plasticating single–screw extrusion of amorphous polymers: development of a mathematical model and comparison with experiment. *Polymer Engineering & Science*. 1996. Vol. 36. № 10. P. 1360–1376.

234. Huang H. X., Peng Y. C. Theoretical modeling of dispersive melting mechanism of polymers in an extruder *Adv. in polym. technol*, 1993. Vol. 12. №47. P. 343–352.

235. Huminic G., Huminic A. Heat transfer characteristics in double tube helical heat exchangers using nanofluids. *International Journal of Heat, Mass Transfer*, vol. 54, no 19-20, 2011, p. 4280– 4287, Elsevier.

236. Ho Q., Carmeliet J., Datta A.. Multiscale modeling in food engineering, *J. food Eng.*, 114 (2013), P. 279–291.

237. Hopmann C., Michaeli W. Extrusion Dies for Plastics and Rubber, 4th ed.; Hanser Publisher: New York, NY, USA, 2016

238. Hopmann C., Kremer C., Grammel S. Predicting the Melting Behavior Within a Single Screw Extruder Using 3D FVM simulation. *In Proceedings of the Polymer Processing Society 28th Annual Meeting (PPS-28)*, Pattaya, Thailand, 11–15 December 2012.

239. Gao F., Jin Z., Chen X. A visual barrel system for study of reciprocating screw injection molding. *Polymer Engineering & Science*, vol. 40, no 6, 2000, p. 1334–43.

240. Gaspar-Cunha A. Optimization of Single Screw Extrusion. Th`ese de doctorat, PhD. Thesis, University of Minho, Guimaraes, Portugal, 2000.

241. Gill W., Sankarasubramanian R. Exact analysis of unsteady convective diffusion. *Proceedings of the Royal Society of London A: Mathematical, Physical, Engineering Sciences*, vol. 316, no 1526, 1970, p. 341–350, The Royal Society.

242. Ghoreishy M., Rafizadeh M. Numerical simulation of thermoplastic melt flow in single screw extruder using finite element method. *Plastics rubber, composites processing, applications*, vol. 25, no 3, 1996, p. 120–125, Institute of Materials.

243. Gotz T., Pinnau R., Struckmeier G. Optimal control of crystallization processes. *Math. Mod. Meth. Appl. Sci.* (M3AS) 16, p. 2029–2045 (2006)

244. Gupta S. The classical Stefan problem. Basic concepts, Modeling and Analysis. *In Applied Mathematics, Mechanics*. North Holland, 2003

245. Ivaniuk V., Fedorchuk V. Application of correlation method for identification of models of the linear dynamic systems with distributed. *Danish Scientific Journal*, 2019. № 28 (3). P. 45–53.

246. Iwko J., Steller R., Wroblewski R. Experimental verification of computer model for polymer plastication process in injection molding. *Polimery*, vol. 60, 2015.

247. John A., Nidhi M. Modelling, Analysis of an Automotive Bumper Used for a Low Passenger Vehicle. *International Journal Of Engineering Trends, Technology (IJETT)*, vol. 15, no 7.

248. Kacir L., Tadmor Z. Solids conveying in screw extruders part iii: The delay zone. *Polymer Engineering & Science*, vol. 12, no 5, 1972, p. 387–395, Wiley Online Library.

249. Kaka S., Yener Y. Exact solution of the transient forced convection energy equation for timewise variation of inlet temperature. *International Journal of Heat, Mass Transfer*, vol. 16, no 12, 1973, p. 2205 - 2214.

250. Kakac S., LI W., Cotta R. Unsteady laminar forced convection in ducts with periodic variation of inlet temperature. *Journal of Heat Transfer*, vol. 112, no 4, 1990, p. 913–920, American Society of Mechanical Engineers.

251. Karapetsas G., Tsamopoulos J. Transient squeeze flow of viscoelastic materials *Journal of Non-Newtonian Fluid Mechanics*, V. 133. 2006. P. 35–56.

252. Kazmer D., Grosskopf C., Venoor V. Vortical Fountain Flows in

Plasticating Screws. *Polymers*. 2018, 10, P. 823

253. Kelly A., Brown E., Coates P. The effect of screw geometry on melt temperature profile in single screw extrusion. *Polymer Engineering & Science*, vol. 46, no 12, 2006, p. 1706–1714.

254. Kim S., Kwoh T. Development of numerical simulation methods, analysis of extrusion processes of particle-filled plastic materials subject to slip at the wall. *Powder technology*, vol. 85, no 3, 1995, p. 227–239.

255. Kissilevski F, Zelensky K. Adaptive control for welding robot. *Proceedings of intern. conf. Automation and robotisation in welding and allied processes*. Strasbourg, France, 1985. C. 127--135.

256. Klonk S., Bay F. Numerical Analysis of Computational Models for Induction Heat Treatment of Complex Geometrical Parts. *International Journal of Microstructure and materials Properties*. 2016. vol. 11. no. 1-2. pp. 48–70

257. Koga S., Diagne M., Krstic M.. Backstepping control of the one-phase stefan problem. *In 2016 American Control Conference (ACC)*, pages 2548–2553. IEEE, 2016.

258. Koga S., M. Diagne M., Krstic M. Control, State Estimation of the One-Phase Stefan Problem via Backstepping Design. Preprint.

259. Koga S., Vazquez R., Krstic M. Backstepping control of the Stefan problem with flowing liquid. *In 2017 American Control Conference (ACC)*, p. 1151-1156. IEEE 2017.

260. Koga S., Diagne M., Krstic M.. Backstepping control of the one-phase stefan problem. *In 2016 American Control Conference (ACC)*, pages 2548–2553. IEEE, 2016.

261. Koga S., M. Diagne M., Krstic M. Control, State Estimation of the One-Phase Stefan Problem via Backstepping Design. Preprint.

262. Koga S., Vazquez R., Krstic M. Backstepping control of the Stefan problem with flowing liquid. *In 2017 American Control Conference (ACC)*, p. 1151-

1156. IEEE 2017.

263. Kostic M. M., Reifschneider L. G. Design of Extrusion Dies. Encyclopedia of Chemical Processing. 2006. P. 633–649.

264. Kreyszig E. Advanced Engineering Mathematics. 10th edition. John Wiley & Sons, Inc., 2010. 1280 p.

265. Krstic M. Compensating actuator, sensor dynamics governed by diffusion PDEs. *Systems & Control Letters*, vol. 58, pp. 372–377, 2009.

266. Van Krevelen D. Properties of Polymers. Amsterdam: Elsevier (1990) 875 pp.

267. Krstic M., Smyshlyaev A., Boundary Control of PDEs: A Course on Backstepping Designs. Singapore: SIAM, 2008.

268. Kostyuk V., Kraskevitch V., Zelensky K. Frequency domain identification of complex systems. *Systems Sceince*, 1977, V.2, C. 5--12.

269. Kumar S., Dubey N. Investigation, Thermal Analysis of Heat Dissipation Rate of Single Cylinder SI Engine, 2017.

270. Li C. Modelling Extrusion Cooking. *Mathematical, Computer Modelling*, 33, p. 553-563. PERGAMON, 2001

271. Liu Y. On Model Reduction of Distributed Parameter Models. Stockholm, 2002. 148 p.

272. Lamnawar K., Maazouz A., Cabrera G. Interfactial tension properties in biopolymer blends: From deformed drop retraction method (DDRM) to shear, elongation rheology-application to blown film extrusion. *Int. Polymer Process*, 33 (2018), P. 411–424.

273. Launay J., Allanic N., Mousseau P. Intrusive measurement of polymer flow temperature. *Polymer Engineering & Science*, vol. 54, no 12, 2014, p. 2806–2814, Wiley Online Library.

274. Leonov A. I., Prokunin A. N. // Nonlinear Phenomena in Flows of Viscoelastic Polymer Fluids. London : Chapman and Hall, 1994. 475 p.

275. Lewandowski A., Wilczynski K. General Model of Polymer Melting in Extrusion Process. *Polimery*, 2018, 63, P. 444–452

276. Lewandowski A., Wilczynski K. Global Modeling of Single Screw Extrusion with Slip Effects. *Int. Polym. Proc.*, 2019, 34, P. 81–90.

277. Lin P., Jaluria Y. Conjugate Thermal Transport in the Channel of an Extruder for Non-Newtonian Materials. *Int. J. Heat Mass Transfer*, 1998, vol. 41, pp. 3239-3253.

278. Lindt J. T. A dynamic melting model for a single-screw extruder. Polymer Engineering & Science, vol. 16, no 4, 1976, p. 284–291.

279. Lindt J. T. Pressure development in the melting zone of a single-screw extruder. Polymer Engineering & Science, vol. 21, no 17, 1981, p. 1162–1166.

280. Lindt J. T. Mathematical modeling of melting of polymers in a single-screw extruder a critical review. Polymer Engineering & Science, vol. 25, no 10, 1985, p. 585–588.

281. Lindt J., Elbirli B. Effect of the cross-channel flow on the melting performance of a single-screw extruder. Polymer Engineering & Science, vol. 25, no 7, 1985, p. 412–418.

282. Lions J., Magenes E. Non-Homogenous Boundary Value Problems and Applications, Volume II. Berlin, Heidelberg, New York: Springer (1972) 242 pp.

283. Liu T., Wong A. C., Zhu F. Prediction of screw length required for polymer melting and melting characteristics. *Intern. polym. process*. 2001. Vol. 16. №2. P. 113–123.

284. Maddock B. A Visual Analysis of Flow, Mixing in Extruder Screws. *SPE. J.*, vol. 15, no 383, 1959, p. 9.

285. Manuel A. A., Fernando T. P., Paulo J. O. Study of steady pipe and channel flows of a single-mode Phan-Thien–Tanner fluid. *Journal of Non-Newtonian Fluid Mechanics*,,V. 101. 2001. P. 55–76.

286. Matthews M. T., Hill J. M. Newtonian flow with nonlinear Navier

boundary condition. Acta Mechanica. 2007. № 191. P. 195–217.

287. Miaw C., Hasson A., Balch G. Melt temperature measurement in a single screw extruder. *43 rd Annual Technical Conference, Society of Plastics Engineers*, Inc, 1985, p. 76–79.

288. Mironov V., Boland T., Trusk T.. (2003). Organ printing: computer-aided jet-based 3D tissue engineering. *In TRENDS in Biotechnology*, 21(4), p. 157-161.

289. Mohamed O., Masood S., Bhowmik J.(2015). Optimization of fused deposition modeling process parameters: a review of current research, future prospects. *In Advances in Manufacturing*, 3(1), p. 42-53.

290. Mohr W., Clapp J., Starr F. Flow patterns in a non-Newtonian fluid in a single-screw extruder. *Polymer Engineering & Science*, vol. 1, no 3, 1961, p. 113–120, Wiley Online Library.

291. Muller J., Kummer S., Fischer D. New ultrasonic probes for in-line monitoring of polymer melts. *Measurement Science and Technology*, vol. 20, no 9, 2009, p. 097002, IOP Publishing.

292. Mathematical Modelling for Polymer Processing. Polymerization, Crystallization, Manufacturing. / ed. V. Capasso, Heidelberg: 2002, 320 pp.

293. Matthews M., Hill J. Newtonian flow with nonlinear Navier boundary condition. *Acta Mechanica*. 2007. № 191. P. 195–217.

294. Micheletti A., Capasso V. The stochastic geometry of polymer crystallization processes. *Stoch. Anal. Applic.* 15 (1997) p. 355–373.

295. Moeller J. Random Johnson-Mehl tessellations. *Adv. Appl. Prob.* 24 (1992) p. 814–844.

296. Moeller J. Generation of Johnson-Mehl crystals and comparative analysis of models for random nucleation. *Adv. Appl. Prob.* 27 (1995) p. 367–383.

297. Monasse D., Haudin J. Thermal dependence of nucleation and growth rate in polypropylene by nonisothermal calorimetry. *Colloid Polymer Sci.* 264 (1986) p. 117–122.

298. Micheletti A., Capasso V. The stochastic geometry of polymer crystallization processes. *Stoch. Anal. Applic.* 15 (1997) p. 355–373.

299. Nietsch T., Cassagnau P., Michel A. Melt temperatures, residence times in an extruder by infrared spectroscopy. *International Polymer Processing*, vol. 12, no 4, 1997, p. 307–315, Carl Hanser Verlag.

300. Del Pilar Noriega M., Osswald T., Ferrier N. In line measurement of the polymer melting behavior in single screw extruders. *Journal of polymer engineering*, vol. 24, no 6, 2004, p. 557–578, De Gruyter

301. Ostrikov A., Abramov O., Vasilenko V.. Mathematical modeling of anomalously viscous flow in the channels of extruders. Publishing, Printing Center of Voronezh State University, (2010), 240. *Mathematical Biosciences, Engineering*, V. 16, Issue 4, p. 2875–2905.

302. Ostrikov A., Shakhov S., Ospanov A. Mathematical modeling of product melt flow in the molding channel of an extruding machine with meat filling feeding, *J. Food Process Eng.*, 41 (2018).

303. Pinho F. T., Oliveira P. J. Axial annular flow of a nonlinear viscoelastic fluid an analytical solution. *Journal of Non-Newtonian Fluid Mechanics*. V. 93. 2000. P. 325–337.

304. Potente H., Bornemann M., Kurte-Jardin M. Analytical Model for the Throughput and Drive Power Calculation in the Melting Section of Single Screw Plasticizing Units Considering Wall-Slippage. *Int. Polym. Proc.*, 2009, 24, P. 23–30

305. Pavlov V., Nastenko Ie., Nosovetc E., Zelensky K.Optimal complexity models in individual control strategy task for objects that cannot be related. *2019 IEEE 14th International conference on computer Sciences and information technologies*

306. Pavlov V., Nastenko Ie., Nosovetc E., Zelensky K. Solving the Individual Control Strategy Tasks Using the Optimal Complexity Models Built on the Class of Similar Objects. *Advances in Intelligent Systems and Computing IV. CCSIT 2019*.

Advances in Intelligent Systems and Computing, vol 1080. Springer

307. Ping W., Chen G. Environmental dispersion in a tidal wetland with sorption by vegetation. *Communications in Nonlinear Science, Numerical Simulation*, vol. 22, no 1-3, 2015, p. 348–366.

308. Pleshivtseva Yu., Rapoport E., Efimov A. Special Method of Parametric Optimization of Induction Heating Systems. *International Scientific Colloquium Modelling for Electromagnetic Processing*. Hannover, Oct. 27–29 2008. P. 229–234.

309. Pujos C., Reginter N., Defaye G. Determination of the inlet temperature profile of an extrusion die in unsteady flow. *Chemical Engineering, Processing: Process Intensification*, vol. 47, no 3, 2008, p. 456–462.

310. Pujos C., Defaye G. Determination of the inlet temperature profile of an extrusion die in unsteady flow. *Chemical Engineering, Processing: Process Intensification*, vol. 47, no 3, 2008, p. 456–462, Elsevier.

311. Ramos J. Propagation and interaction of moving fronts in polymer crystallization. *Appl. Math. Comput.* 189, p. 780–795 (2007)

312. Rao N. Computer Aided Design of Plasticating Screws. 1986.

313. Ratajski E., Janeschitz-Kriegl H. How to determine high growth speeds in polymer crystallization. *Colloid Polymer Sci.* 274 (1996) p. 938–951.

314. Rauwendaal C. Polymer Extrusion. Munich : Carl Hanser Verlag GmbH, 2014. 950 p.

315. Rauwendaal C. Conveying, Melting in Screw Extruders with Axial Screw Movement. I *nt. Polym. Proc.*, vol. 17, 1992, p. 26.

316. Ruan C. Kinetics and morphology of flow indused polimer cristallization in 3D shear flow investigated by monte Carlo simulation. *Cristals*, 2017, 7,51. 16 p.

317. Ruberg M., Elsass M., Kelsay S. Experimental Study on the Energy Efficiency of Different Screw Designs for Injection Molding. *SPC ANTEC Conference Proceedings*, 2008, p. 474–477.

318. Ruberg M., Waterfield R., Elsass M. Experimental Study on the Energy

Efficiency of Different Screw Designs for Injection Molding. *SPC ANTEC Conference Proceedings*, 2008, p. 474–477.

319. Rudnev V., Loveless D., Cook R. Handbook of Induction Heating: 2nd edition, CRC Press. 2017. 780 p.

320. Sarkar A., Jayaraman G. The effect of wall absorption on dispersion in oscillatory flow in an annulus: application to a catheterized artery. *Acta Mechanica*, vol. 172, no 3, 2004, p. 151–167.

321. Scott G., Kilgour D. The density of random close packing of spheres. Journal of Physics D: *Applied Physics*, vol. 2, no 6, 2014, p. 863–866.

322. Schowalter W. R. The behavior of complex fluids at solid boundaries. *Journal of Non-Newtonian Fluid Mechanics*, 1988. № 29. P. 25–36.

323. Shapiro J., Halmos A., Pearson J. Melting in single screw extruders. *Polymer*, vol. 17, no 10, 1976, p. 905–918.

324. Song F., Ni Y., TAN Z. Optimization design, modeling, dynamic analysis for composite wind turbine blade. *Procedia Engineering*, vol. 16, 2011, p. 369–375.

325. Steller R., Iwko J. Polymer Plastication during Injection Molding. *Int. Polym. Proc.*, vol. 23, 2008, p. 252.

326. Stevens M., Covas J. Extruder principles, operation. Springer Science & Business Media, 2012.

327. Syrja S. Numerical study of fully developed non-Newtonian fluid flow, heat transfer in a rectangular channel with a moving wall. *International communications in heat, mass transfer*, vol. 24, no 1, 1997, p. 11–25.

328. Syrjala S. Numerical simulation of nonisothermal flow of polymer melt in a single-screw extruder: a validation study. *Numerical Heat Transfer*, Part A. 2000. V. 37. P. 897–915.

329. Syrjala S. On the analysis of fluid flow and heat transfer in the melt conveying section of a single-screw extruder. *Numer. Heat Transfer*, Part A, 1999, vol. 35, no. 1, pp. 25-47.

330. Supaphol P., Spruiell J. Thermal properties and isothermal crystallization of syndiodactic polypropylenes: differential scanning calorimetry and overall crystallization kinetics. *J. Appl. Polymer Sci.* 75 (2000) p. 44–59.

331. Seitz H., Rieder W., Irsen S. Three dimensional printing of porous ceramic scaffolds for bone tissue engineering. *In Journal of Biomedical Materials Research Part B: Applied Biomaterials*, 74(2), p. 782-788, 2005

332. Tadmor Z., Gogos G. Principles of polymer processing. John Wiley & Sons, 2006, 964 p.

333. Tadmor Z. Fundamentals of plasticating extrusion. I. A theoretical model for melting. *Polymer Engineering & Science*, vol. 6, no 3, 1966, p. 185–190, Wiley Online Library.

334. Tadmor Z., Klein I. Engineering principles of plasticating extrusion. Van Nostrand Reinhold Co., 1970.

335. Tanner R. I. Phan-Thien N., Huang X. Two and three-dimensional finite volume methods for flows of viscoelastic fluids. Seville, Spain.: Proc. 4th Eur. Cong. Rheology. 1994. P. 362–364.

336. Taylor F. Dispersion of soluble matter in solvent flowing slowly through a tube. Proc. R. Soc. Lond. A, vol. 219, no 1137, 1953, p. 186–203.

337. Trippe J., Schoppner V. Modeling of Solid Conveying Pressure Throughput Behavior of Single Screw Smooth Barrel Extruders under Consideration of Backpressure and High Screw Speeds. *Int. Polym. Process.* 2018, 33, P. 486–496

338. Trofymchuk O., Zelensky K., Nastenko Ie. Modeling of a temperature field for extruder body. *System investigation and information theory*. 2021, N 1.

339. Xu J., Lu T., Hodson H., Fleck N. Analysis of thermal dispersion in an array of parallel plates with fully-developed laminar flow. *Internatinal Journal of Heat, Fluid Flow*, vol. 31, no 1, 2010, p. 57–69.

340. Xue S., Phan-Thien R., Tanner C. Three dimensional numerical simulations of viscoelastic flows through planar contractions. *J. Non-Newtonian*

Fluid Mec. 1998. V.74. P. 129-245.

341. Yang Y., Yang W., Zhong H. Temperature distribution measurement, control of extrusion process by tomography. 2008 IEEE International Workshop on Imaging Systems, Techniques IEEE, 2008, p. 170–174.

342. Yu J. Dispersion in laminar flow through tubes by simultaneous diffusion, convection. *Journal of Applied Mechanics*, vol. 48, no 2, 1981, p. 217–223.

343. Yung K. , Xu Y. Analysis of a Melting Model for an Extruder with Reciprocation. *J. Mater. Process. Technol.*, vol. 117, 2001, p. 21.

344. Yung K., Xu Y., Lau K. Simulation of transient process in melting section of reciprocating extruder. *Polymer*, vol. 43, no 3, 2002, p. 983–988, Elsevier.

345. Yung K. L., Xu Y., Lau K. H. Transient Melting Models for the Three Stages of Reciprocating Extrusion. *J. Mater. Process. Technol.*, vol. 139, 2003, p. 170.

346. Yu J. Dispersion in laminar flow through tubes by simultaneous diffusion, convection. *Journal of Applied Mechanics*, vol. 48, no 2, 1981, p. 217–223,

347. Vlachopoulos J. Polymer Rheology and Extrusion; McMaster University: Hamilton, ON, Canada, 2011.

348. Vera-Sorroche J., Kelly A., Brown E. The effect of melt viscosity on thermal efficiency for single screw extrusion of HDPE. *Chemical Engineering Research, Design*, vol. 92, no 11, 2014, p. 2404 - 2412.

349. Vera-Sorroche J., Kelly A., Brown E. Infrared melt temperature measurement of single screw extrusion. *Polymer Engineering & Science*, vol. 55, no 5, 2015, p. 1059–1066, Wiley Online Library.

350. Vinnikova L. Extrusion processing of products with dietary fiber, *Food Ind.*, 11 (1991), p. 51–55.

351. Wilczynski K., Lewandowski A., Nastaj A. Modeling for Starve Fed/Flood Fed Mixing Single-Screw Extruders. *Int. Polym. Proc.* 2016, 31, P. 82–91.

352. Wilczynski, K., Lewandowski A., Nastaj A. Global Model for Starve-Fed

Nonconventional Single-Screw Extrusion of Thermoplastics. *Adv. Polym. Technol.*, 2017, 36, P. 23–35.

353. Wilczynski K. Single-screw extrusion model for plasticating extruders. *Polymer- Plastics Technology and Engineering,* vol. 38, no 4, 1999, p. 581–608.

354. Zelenskaya N., Zelensky K. Approximation of Bessel functions by rational functions. *Electronics and control systems*. 2015, N 2 (44). p. 123--129.

355. Zhao J., Mascia L., Nassehi V. Simulation of the rheological behavior of polymer blends by finite element analysis // Advances in Polymer Technology. 1997.Vol. 16. № 3. P. 206–226

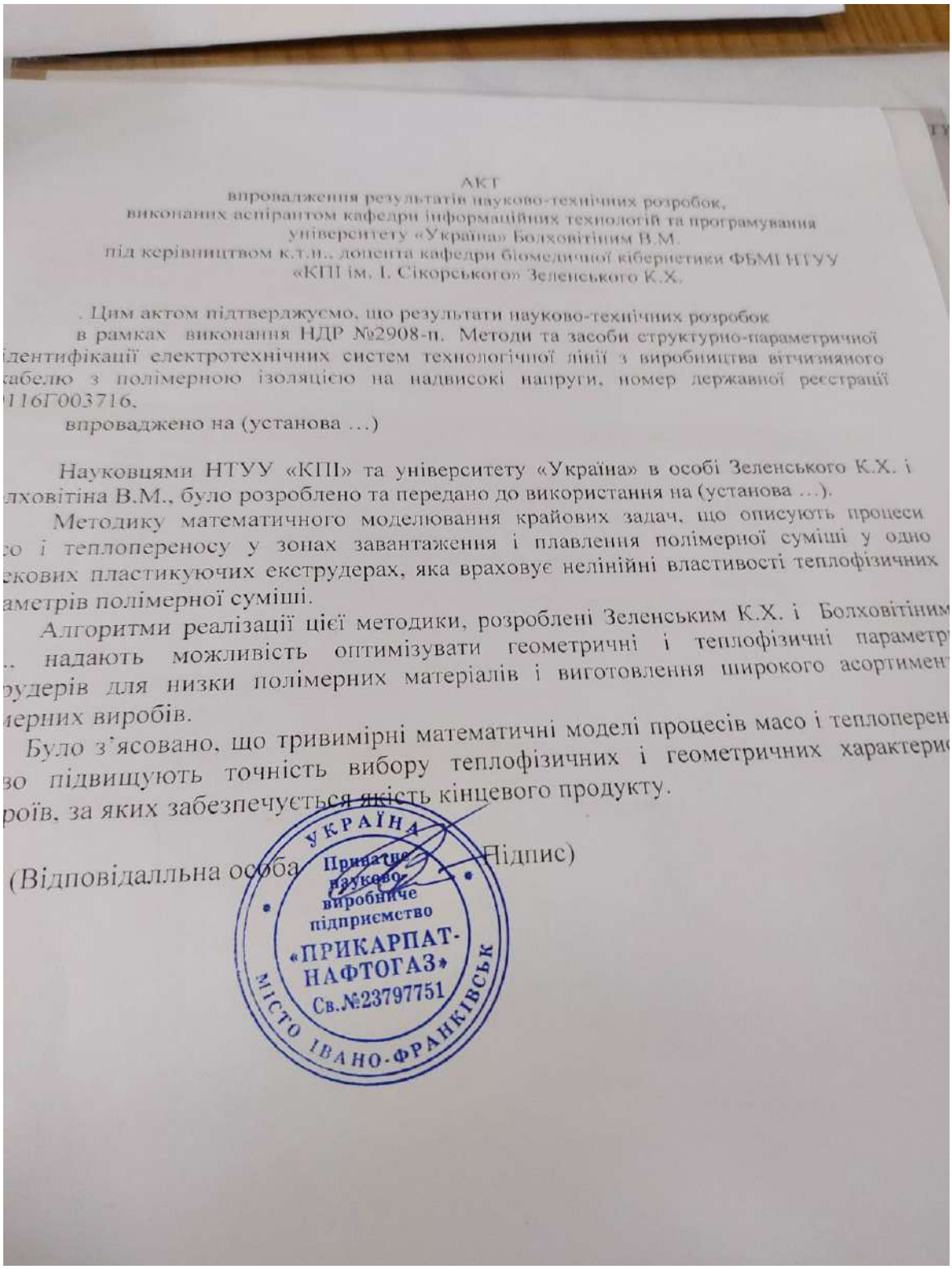
356. Zelensky K., Nastenko E.A. Simulation of vortex flows in the left ventricle of the heart. *Electronics and control systems*. 2017, N 2 (46). p. 73--79.

357. Zenev T., Gogos C. Principles of Polymer Processing, 2nd Edition, John Wiley & Sons, (2013), 624 p.

358. Zhu F., Chen L. Studies on the theory of single screw plasticating extrusion. Part I: A new experimental method for extrusion. *Polymer Engineering & Science*, vol. 31, no 15, 1991, p. 1113–1116.

ДОДАТКИ

Д.1 Акти впровадження.



АКТ
впровадження результатів науково-технічних розробок,
виконаних аспірантом кафедри інформаційних технологій та програмування
університету «Україна» Болховітіним В.М.
під керівництвом к.т.н., доцента кафедри біомедичної кібернетики ФБМІ НТУУ
«КПІ ім. І. Сікорського» Зеленського К.Х.

. Цим актом підтверджуємо, що результати науково-технічних розробок
в рамках виконання НДР №2908-п. Методи та засоби структурно-параметричної
ідентифікації електротехнічних систем технологічної лінії з виробництва вітчизняного
кабелю з полімерною ізоляцією на надвисокі напруги, номер державної реєстрації
0116Г003716,
впроваджено на (установа …)

Науковцями НТУУ «КПІ» та університету «Україна» в особі Зеленського К.Х. і
лховітіна В.М., було розроблено та передано до використання на (установа …).

Методику математичного моделювання крайових задач, що описують процеси
со і теплопереносу у зонах завантаження і плавлення полімерної суміші у одно
кових пластикуючих екструдерах, яка враховує нелінійні властивості теплофізичних
аметрів полімерної суміші.

Алгоритми реалізації цієї методики, розроблені Зеленським К.Х. і Болховітіним
.. надають можливість оптимізувати геометричні і теплофізичні параметр
рудерів для низки полімерних матеріалів і виготовлення широкого асортимен
мерних виробів.

Було з'ясовано, що тривимірні математичні моделі процесів масо і теплоперен
зо підвищують точність вибору теплофізичних і геометричних характери
роїв, за яких забезпечується якість кінцевого продукту.

(Відповідальна особа Підпис)

УКРАЇНА
Приватне науково-виробниче підприємство
«ПРИКАРПАТ-НАФТОГАЗ»
Св.№23797751
МІСТО ІВАНО-ФРАНКІВСЬК

ТОВАРИСТВО
З ОБМЕЖЕНОЮ ВІДПОВІДАЛЬНІСТЮ
УКРЕКОКОНСАЛТ
Природа знає краще!

01050, м. Київ, вул. Ярославів вал, буд. 13/2Б
Код ЄДРПОУ 40905042, ІПН
Р/р UA923510050000026006632593900,
Публічне акціонерне товариство
«УкрСиббанк»,
МФО 351005, тел. (068) 721-23-34

№ 02/02-2021 11.02.2021

АКТ

про реалізацію результатів дисертаційних досліджень,
виконаних старшим викладачем кафедри біомедичної кібернетики
факультету біомедичної інженерії НТУУ «КПІ ім. Ігоря Сікорського»
Бовсуновською Катериною Сергіївною
під керівництвом к.т.н. доцента кафедри біомедичної кібернетики ФБМІ
НТУУ «КПІ ім. І. Сікорського» Зеленського К.Х.

Цим актом підтверджуємо, що результати науково-технічних розробок впроваджено на ТОВ «УКРЕКОКОНСАЛТ».

Науковцями НТУУ «КПІ» в особі Зеленського К.Х. та Бовсуновської К.С було розроблено та передано до використання на ТОВ «УКРЕКОКОНСАЛТ» методику математичного моделювання крайових задач, що описують процеси масо і теплопереносу у пиловловлюючому обладнанні для очистки повітря.

У процесі розробки проектних рішень системи очищення викидів, на основі методики, розробленої науковцями в особі Бовсуновської К.С. і Зеленського К.Х., та відповідних алгоритмів були оптимізовані геометричні і теплофізичні параметри циклонного фільтру.

За допомогою тривимірної математичної моделі процесів масо і теплопереносу було суттєво підвищено точність визначення необхідних теплофізичних і геометричних характеристик, підвищено ступінь очищення повітря від шкідливих домішок, що реалізовано у рекомендаціях до проектної документації на виготовлення пиловловлюючого обладнання.

Директор Д. Гулевець

Громадська організація
« Асоціація фахівців цивільного захисту »
(ГО «АЦЗ»)

Код ЄДРПОУ 41695173
Україна, 03035, м. Київ, вулиця Георгія Кірпи, 2-А, офіс 513
Тел. / факс: +38-044-502-12-29;
e-mail: info@cpma.org.ua

Public Organisation
«Civil Protection (profi) Manager's Association»
(CPpMAs)

Ukraine, 03035, Kyiv, Georgy Kirpa Street, 2-A, office 513
Phone / fax: +38-044-502-12-29;
e-mail: info@cpma.org.ua

14.04.2021 № 11/21

На № _____ від _____

[Адресат]

[Заголовок до тексту]

ЗАТВЕРДЖУЮ
Голова Асоціації фахівців цивільного захисту
д.т.н., с.н.с. С.М.Чумаченко
"___" _____ 2021 р.

АКТ

впровадження результатів науково-технічних [illegible] виконаних к.т.н., доцентом кафедри біомедичної кібернетики ФБМІ НТУУ «КПІ ім. І. Сікорського» Зеленським К.Х.

Цим актом підтверджуємо, що результати досліджень, виконаних Зеленським К.Х. у дисертаційній роботі «Математичне моделювання нелінійних полімерних матеріалів в екструдерах», здійснено впровадження технічних рекомендацій по уточненому конструктивному виконанню розмірів екструдера для ізоляційного полімерного покриття кабельної продукції.

Апробація розробленої методики комп'ютерного моделювання засвідчила, що урахування нелінійних властивостей полімерів під час нагрівання суміші до температури плавлення із подальшою кристалізацією розплаву полімеру підвищує якість ізоляційного покриття і запобігає виникненню дефектів у полімерному покритті. Надані Зеленським К.Х пропозиції щодо визначення оптимальних конструктивних (геометричних) параметрів екструдера дають змогу проектувати екструзійні лінії для виготовлення широкого асортименту полімерних виробів.

Д.т.н., с.н.с. Євген ЯКОВЛЕВ

К.т.н., доцент Сергій ПОНОМАРЕНКО

ДОДАТОК 2.

Д 2.1 Програмна реалізація моделювання масо теплоперенесення в екструдері

```
#include "local.h"

 int korpus (long double *, int, int);

 void slbr2 (long double *, long double *, int);  // 1800
 int bs2int (long double *, long double *, int, int, int);   // 1830
 long double fintgs (long double *, long double *, long double *, long
double *, int, int, int, int); // 1910

 long double fxs (long double *, long double, long double, long double, int,
int);
 long double fxsd(long double, long double, long double, long double);

 int shn10 (long double *, long double *, long double *, int, int);  // 260
 int trns2p (long double *, long double *, long double *, int, int);
 int nlshn (long double *, long double *, long double *, int, int, int);  //
1320
 int trnsf_con (long double *, long double *, long double *, long double *,
long double *, long double *, int, int); // 470
 void orig1 (long double *, long double *, int);
 int yprvn (long double *, long double *, long double *, int, int, int);
 int pslf (long double *, long double *, int, int);
 int pslfn2 (long double *, long double *, int);
 int pslf1 (long double *, long double *, int, int);
 int yprv2 (long double *, long double *, long double *, int);
 int atbt2 (long double *, long double *, long double *, int, int);
 int atbt2p (long double *, long double *, long double *, int, int);
 long double intab (long double *, long double *, long double, long double,
int);
 long double intt2 (long double *, long double, long double, int);
 int pslfn (long double *, long double *, int, int);
 void read_intg (long double *, long double *, long double *, long double *,
int, int, int); //560
 int plav_pol (long double *, long double *, long double *, int, int); //
610
 int nlpol ( long double *, long double *, long double *, int, int); // 1010
 int cristal (long double *, long double *, int, int);
int ftgtp (long double *, long double *, long double *, int);

 main ()
  {
    FILE *fcl;
    int m, n, k, nk, m2;
    long double *a, *b, *e, *c, *f, *d, *h, *g, *sb, *zs, *ctn, *bt, *tn,
*bpt, tm, *vr1, *vz1, *tp1, *vr2, *vz2, *tp2;
    static long double g1[]={0,1,0.3,2.5,0,0,0}, h1[]={0.8,1,0.5,3,0,0,0},
                ctnl[300], bt1[50], bpt1[300], vr11[300], vz11[300],
ttp1[300];
```

```
    static long double vr21[300], vz21[300], ttp2[300], a1[20], c1[40],
e1[10], d1[10];
    static long double  zz[]={2.9381,6.1863,9.3607,12.519,15.67,18.818,
0.4733,0.49348,0.49712,0.49838,0.49897,0.49928};
      // 2,-4,0,5,2,1}
    static long double sbr[]={4.964, 9.253, 13.66, 18.1, 22.56, 27.03},
      b1[]={0.165,0.32,2.,0.58,0.4,1, 6.2,4.7,1.,.45,0.6,1,
            12.6,9.5,4.,1.,8,1, -2.7,1,-4,.48,0.26,1};
  /* b=b1; a=a1; e=e1; d=d1;
   pslfn(a,b,4,1);
   for (n=0;n<4; n++)
     { nk=6*n;
       for (k=0;k<6;k++) *(d+k)=*(b+nk+k);
       printf("\n simpl. sum[1-4]V2_%u(p)-> V2(p) -> G(t)\n",n);
       orig1 (e,d,1);

       for (k=0;k<6;k++) *(c+k+nk)=*(e+k);
     }
   printf("\n simpl. V2(p)-> G(t)\n");
   orig1 (e,a,1);
   for (k=0;k<6;k++) *(c+k+24)=*(e+k);
   fcl = fopen("uprvt4-1.dat","w");
   for (k=0; k<30; k++) fprintf(fcl,"%g,",*(c+k));
   fclose(fcl);
  */
    vr1=vr11; vz1=vz11;
    tp1=ttp1;
    vr2=vr21; vz2=vz21; tp2=ttp2;
     /*
         a=a1; c=c1;
         d=d1; f=f1;
         intt2 (d,0,2,1);
         for (k=0;k<3;k++) *(d+k)*=0.151*0.1631;
         atbt2p (c,d,f,0,1);
         orig1 (a,f,1);
         orig1 (a,d,1);

         tranpol (vr2, vz2, tp2, 6,6);
    */

    sb=sbr;
    bpt=bpt1;
    zs=zz;
    ctn=ctnl;
    bt=bt1;
    m=6;
    n=6;
    cristal (sb, zs, m, n);
    /*
    h=h1;
    bs2int (sb, h, 0, m, 0);

    h=g1;
    bs2int (zs, h, 1, n, 0);

    h=h1;
```

```
    *h=0.6;
    *(h+1)=0.8;
    *(h+2)=0.5;
    *(h+3)=3.5;
    *(h+4)=0;
    printf("\n h=");
    for (k=0;k<5;k++) printf("%8.2f",*(h+k));
    bs2int (sb, h, 0, m, 1);

    korpus (ctn, m, n);

    shn10 (bpt, bt, ctn, m, n);

    plav_pol (vr1, vz1, tp1, m, n);

    nlpol (vr1, vz1, tp1, m, n);
    */
  }


 int korpus (long double *g, int mm, int nn)
  {
    FILE *fcl;
    int i, j, n, m, k, nb, mb, nm, m6, n2, m2, n3, k2;
     long double *b, *c, *bt, *z, *bnt, *brz, *bsi, *d, r1, r2, s1, s2, d1,
b0, b1, br, bz;
    long double a, bl, cv0, al, gr, gr0, ro, ra, rb, zl, z0, grb, gra, z1,
t0k, tLk, h;
    brz=(long double *)calloc(100,sizeof(long double));
    bnt=(long double *)calloc(100,sizeof(long double));
    b=(long double *)calloc(10,sizeof(long double));
    c=(long double *)calloc(10,sizeof(long double));
    bt=(long double *)calloc(100,sizeof(long double));
    z=(long double *)calloc(150,sizeof(long double));
    bsi=(long double *)calloc(30,sizeof(long double));
    d=(long double *)calloc(50,sizeof(long double));
    if ( b==NULL || c==NULL || bt==NULL || z==NULL || bnt==NULL || bsi==NULL
|| brz==NULL || d==NULL )
     { printf("\n\t arrays in korpus0=0\n");
       return(-1);
     }
    else
     {

    // Intg. transforms for bessel2.
       m2=3*mm;
       fcl = fopen("bsik1.dat","r");     // bsk1.dat
      for (i=0; i<m2; i++) fscanf(fcl,"%le,",(bsi+i));
       fclose(fcl);
       printf("\n\t R_n(r): sb(%u) and int R_n(r)\n",mm);
       for (k=0;k<mm;k++) printf("%11.4g",*(bsi+k)); PRN;
       for (k=0;k<mm;k++) printf("%11.4g",*(bsi+k+mm)); PRN;

    // Intg. transforms for z.
       n2=3*nn;
       fcl = fopen("zint2.dat","r");
```

```
        for (i=0; i<n2; i++) fscanf(fcl,"%le,",(z+i));
        fclose(fcl);

        printf("\n\t  self-values for Z_k(%u)\n",nn);
        for (k=0;k<nn;k++) printf("%11.5g",*(z+k)); PRN;
        printf("\n\t  int z(%u)=zi(nn)\n",nn);
        for (k=0;k<nn;k++) printf("%10.5g",*(z+k+nn)); PRN;
        printf("\n\t  \| z(%u)\|=z2(nn)\n",nn);
        for (k=0;k<nn;k++) printf("%11.5g",*(z+k+nn+nn)); PRN;
        n2=nn*nn;
        n3=3*nn;

        ra=0.8;
        rb=1.;
        h=0.2;
        bl=76;
        cv0=0.457;
        ro=7800;
        al=bl/(cv0*ro);
        z0=1;
        zl=1.9;
        t0k=100;   // 45
        tLk=120;
        grb=1200; // 1200
        gra=100;
        // a=0.8;
        printf("\n korpus: z0=%6.2f, zl=%6.2f; gra=%8.2g, grb=%10.4g,
tLk=%8.2g, lbk=%8.2g, alk=%8.2g\n",z0,zl,gra,grb,tLk,bl,al);
        for (n=0;n<mm;n++)
          {
            s1=*(bsi+n);

            *(b+n)=fxs (&s1,0.3,h,ra,0,1);
            b1=fxs (&s1,0.3,h,rb,0,1);
            br=grb*b1 -gra*(*(b+n)); // Bound. cond on R/R0
        //    printf("\n\t grb*b1=%10.4g*%10.4g; gra%10.4g*b(%u)%10.4g=
br(%u)=%10.4g\n",grb,b1,gra,n,*(b+n),n,br);
            nb=n*mm;
            for (k=0;k<nn;k++)
              { z1=*(z+k);
                *(bt+nb+k)=r2=(s1*s1+z1*z1)*al;
                bz=(h*sin(z1*zl)-z1*cos(z1*zl));
                *(brz+nb+k)=*(z+k+nn)*br+(tLk*bz+(1-
h)*h/z1*t0k)*(*(bsi+n+mm));

             //   printf("\n\t brz(%u)=%10.4g\n",nb+k,*(brz+nb+k));
              }
          }
        printf("\n\t korpus: Rn(%5.2f)\n",ra);
        for (k=0;k<nn;k++) printf("%10.3g",*(b+k)); PRN;
        printf("\n\t korpus: brz(n=1,%u,k=1,%u)=\n",mm,nn);
        nm=nn*mm;
        for (i=0;i<nm;i++)
            printf("%10.3g%c",*(brz+i), (i%(mm)==mm-1) ? '\n' : ' ');

        printf("\n\t korpus: bt(n=1,%u,k=1,%u)=\n",mm,nn);
```

```
        for (i=0;i<nm;i++)
            printf("%10.4g%c",*(bt+i), (i%(mm)==mm-1) ? '\n' : ' ');   PRN;

        nm=nn*mm;
        for (n=0;n<nn;n++)
          { n2=n*nn;
            for (m=0;m<mm;m++)
              { k2=2*m;
                m2=m*nn;
                *(d+k2)=*(brz+m2+n)*(*(b+m));
                *(d+k2+1)=*(bt+m2+n);
              }
            pslf1(c,d,mm,0);
            *(g+n2+2)=*c/(*(c+3));
            *(g+n2)=*(c+1)-(*(g+n2+2))*(*(c+4));
            *(g+n2+1)=-(*(g+n2+2))*(*(c+5));
            for (k=3;k<6;k++) *(g+n2+k)=*(c+k);
          }
        printf("\n\t korpus: bound. values br(%u,6;p)-krp22.dat: \n",nn);
        n2=6*nn;
        for (k=0;k<n2;k++) printf("%11.4g%c",*(g+k), (k%6==5) ? '\n' : ' ');
        for (n=0;n<nn;n++)
          { n2=n*nn;
            for (k=0;k<6;k++) *(c+k)=*(g+n2+k);
            orig1 (b,c,0);
            for (k=0;k<6;k++) *(bnt+n2+k)=*(b+k);
          }
        printf("\n\t korpus: z[%3.1f,%3.1f]; bound. values bnt(%u,6;t)-
krt22.dat: \n",z0,zl,nn);
        printf(" gra=%8.4g, grb=%8.4g,  tLk=%8.4g, lbk=%8.4g,
alk=%8.4g,\n",gra,grb,tLk,bl,al);
        n2=6*nn;
        for (k=0;k<n2;k++) printf("%11.4g%c",*(bnt+k), (k%6==5) ? '\n' : '
');

        fcl = fopen("c:/dis_figs/krt22.dat","w");
        for (k=0; k<nm; k++) fprintf(fcl,"%g,",*(bnt+k));
        fclose(fcl);

        free(d);
        free(bsi); free(z);
        free(bt); free(c);
        free(b); free(bnt);
        free(brz);

        return(nm);
      }
  }

 long double fxs (long double *sl, long double a, long double h, long double
x, int ps, int m)
  { int k;
    long double fx, x2, r1, r2;
   //  printf("\n\tsl=%8.2f,a=%8.2f,x=%8.2f\n",*sl,a,x);
    fx=1;
    for (k=0;k<m;k++)
```

```
      { x2=*(sl+k);
        if (ps==0 )
         { r1=yn(1,x2*a)/jn(1,x2*a);
           r2=yn(0,x2*x)-r1*jn(0,x2*x); }
        if (ps==1 ) r2=cos(x2*x)+h/x2*sin(x2*x);
        fx*=r2;
      }
    if ( ps==0 ) fx*=x;
   // printf("\n\tfxs: fx(%u:%10.4g,%10.4g)=%10.4g\n",m,x2,x,fx);
    return(fx);
  }

 long double fxsp (long double *sl, long double a, long double h, long
double x, int ps, int m)
  {
    // ps=0:  fx=Rn(s*r)*dR_j(s1*r)/dr; ps=1:  fx=Zn(s*z)*dZ_j(s1*z)/dz;

    int k;
    long double fx, x1, x2, r1, r2;
    x1=*sl;
    x2=*(sl+1);
    if ( ps==0 )
     { r1=(x1*yn(1,x1*a)+h*yn(0,x1*a))/(x1*jn(1,x1*a)+h*jn(0,x1*a));
       r2=yn(0,x1*x)-r1*jn(0,x1*x);
       r1=(x2*yn(1,x2*a)+h*yn(0,x2*a))/(x2*jn(1,x2*a)+h*jn(0,x2*a));
       fx=-(yn(1,x2*x)-r1*jn(1,x2*x))*r2;
     }
    if ( ps==1 )
     { r2=h*cos(x2*x)-x2*sin(x2*x);
       fx=(cos(x1*x)+h/x1*sin(x1*x))*r2;
     }
    if ( ps==0 ) fx*=x;
    // printf("\n\tfxs: fx(%10.4g,%10.4g)=%10.4g\n",x2,r1,fx);
    return(fx);
  }

 long double fxsd(long double sl, long double a, long double h, long double
x)
  {
    long double fx, r1;
    r1=(sl*yn(1,sl*a)+h*yn(0,sl*a))/(sl*jn(1,sl*a)+h*jn(0,sl*a));
    fx=-(yn(1,sl*x)-r1*jn(1,sl*x)); //*sl;
  //  printf("\n\tfxsd: fx(%6.3f,%10.4g,%10.4g)=%10.4g\n",a,sl,r1,fx);
    return(fx);
  }

 long double accur (long double *, int, int, int);  // 1212

 int shn10 (long double *tpt, long double *bt, long double *br, int mm, int
nn)
  {
       FILE *fcl;
    // br(nn,6) -- Tk(r_3;nn,t) from korpus; bt(mm,nn,t) -- output;
    // bt - self-values; bpt -- T_p(mm,nn,t).

    int i, j, n, m, k, k2, nb, n1, nm, m2, n2, m6, n6, mb, lp, m1, it;
```

```
    long double *tpp0, *tpp, *z, *z2, *bsi, *sb, *b, *c, *d, *e;
    long double r1, r2, s1, s2, d1, b0, b1, bz, bt1;
    long double blk, blp, bl, cv0, al, gr, gr0, ro, ra, rb, zl, z0, z1, t0,
h;
    long double dz, dt, dr, rr, zz, tt, tt0, tl, rs, rc, f1, eps=0.03, dlt;
    tpp0=(long double *)calloc(400,sizeof(long double));
   // tpt1=(long double *)calloc(400,sizeof(long double));
    tpp=(long double *)calloc(400,sizeof(long double));
    b=(long double *)calloc(30,sizeof(long double));
    c=(long double *)calloc(60,sizeof(long double));
    z=(long double *)calloc(130,sizeof(long double));
    bsi=(long double *)calloc(130,sizeof(long double));
    d=(long double *)calloc(20,sizeof(long double));
    e=(long double *)calloc(10,sizeof(long double));
    if (tpp0==NULL || tpp==NULL || b==NULL || c==NULL || z==NULL ||
bsi==NULL || d==NULL || e==NULL )
     { printf("\n\t arrays in shn10=0\n");
       return(-1); }
    else
     {
        /* Input: values on bound from kropus: br=V^{m,n}_6(p).
           Output: Tp(m,n,t) -- array bt(m,n,6;t); */

          // Intg. transforms for bessel2.
       fcl = fopen("bsp1.dat","r");
       for (i=0; i<mm+mm; i++) fscanf(fcl,"%le,",(bsi+i));
       fclose(fcl);
       printf("\n \t self-values and int(B_m(r))=bsi(%u)\n",mm);
       for (k=0;k<mm;k++) printf("%10.4g",*(bsi+k)); PRN;
       for (k=0;k<mm;k++) printf("%10.4g",*(bsi+k+mm)); PRN;

    // Intg. transforms for z.
       n1=3*nn;
       n2=n1+2*nn*nn;
        fcl = fopen("zint1.dat","r");
        for (i=0; i<n2; i++) fscanf(fcl,"%le,",(z+i));
        fclose(fcl);
       printf("\n \t  z(%u)=zl(nn),zi(nn)\n",nn);
       for (k=0;k<nn;k++) printf("%11.5g",*(z+k)); PRN;
       for (k=0;k<nn;k++) printf("%11.5g",*(z+k+nn)); PRN;
       n2=nn*nn;
       n1=6*nn;
       printf("\n\t boundary korpus--shnek T_n(p)\n");
       for (i=0;i<n1;i++)
           printf("%10.4g%c",*(br+i), (i%6==5) ? '\n' : ' ');

           // parameters for polimer
       blk=57;
       blp=1.82;
       cv0=0.25;  // 2.5
       ro=779;
       al=blp/(cv0*ro);

       zl=1;
       ra=0.3;
       rb=0.8;
```

```
        t0=120;
        h=0.3;
        printf("\n\t shnek zl=%5.2f: lb=%9.3g; cv0=%9.3g; rho=%9.3g;
al=%10.4g\n",zl,blp,cv0,ro,al);
        bl=blk/blp;
       // n1=3*nn;
        for (m=0;m<mm;m++)
          { m2=m*mm;
            mb=6*m*mm;
            s1=*(bsi+m);
            *(d+m)=fxs ((bsi+m),ra,h,rb,0,1);
            for (n=0;n<nn;n++)
              { n2=n*nn;
                nb=6*n;
                z1=*(z+n);
                bt1=*(bt+m2+n)=(s1*s1+z1*z1)*al;
                r2=t0*(*(z+n+nn))*(*(bsi+m+mm));
                for (k=0;k<nn;k++)
                  { k2=6*k;
                    r1=*(z+n2+n1+k); // *(*(d+m));
                    for (j=0;j<6;j++) *(c+j+k2)= (j<3) ? *(br+k2+j)*r1 :
*(br+k2+j);
                  }

                pslfn(e,c,nn,0);
                *(e+2)+=r2;
                for (k=0;k<3;k++)  *(e+k)*=*(d+m);
                *c=1.;
                *(c+1)=bt1;
                yprv2 (b,c,e,0);
                for (k=0;k<6;k++) *(tpp0+k+nb+mb)=*(b+k);  // Tp0(mm,nn,p)

                orig1 (c,b,0);
                for (k=0;k<6;k++) *(tpt+k+nb+mb)=*(c+k);  // Tp0(mm,nn,t)
              }
          }
        printf("\n\t R_m(%6.2f)=d=",ra);
        for (j=0;j<mm;j++) printf("%10.4g",*(d+j)); PRN;

        printf("\n\t sigma = bt(%u,%u)\n",mm,nn);
        for (m=0;m<mm;m++)
          { mb=6*m;
            for (j=0;j<nn;j++) printf("%10.4g",*(bt+mb+j)); PRN; }
        printf("\n\t shn10 -- Linear part:
cv0=%7.3f;tpp0(mm,nn;p)\n",cv0,mm,nn);
        for (m=0;m<mm;m++)
          { mb=6*m*mm;
            for (k=0;k<nn;k++)
              { n2=6*k;
                for (j=0;j<6;j++) printf("%10.4g",*(tpp0+mb+n2+j)); PRN; }
            PRN; }
        printf("\n\t Linear part: cv0=%7.3f; tpt(mm,nn;6) --
Tp(%u,%u,t)\n",cv0,mm,nn);
        for (m=0;m<mm;m++)
          { mb=6*m*mm;
            for (k=0;k<nn;k++)
```

```
            { n2=6*k;
              for (j=0;j<6;j++) printf("%10.4g",*(tpt+mb+n2+j)); PRN; }
          PRN; }

       n2=6*mm*nn;
       fcl = fopen("shn0.dat","w");
       for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tpt+k));
       fclose(fcl);
      /*
       for (k=0;k<mm;k++)
         { *(c+k)=*(bsi+k);
           *(c+mm+k)=*(z+k); }
       */
       dlt=1.;
    //   it=1;
    //   while (dlt > eps )
       for (it=1;it<3;it++ )
         { printf ("\n\t shn10: it=%u; dlt=%10.4g, eps=%8.3f\n",it,dlt,eps);

           nlshn (tpp, tpp0, bt, mm, nn, it);

          for (m=0;m<mm;m++)
            { mb=6*m*mm;
              m2=m*mm;
              for (k=0;k<nn;k++)
                { n2=6*k;
                  for (j=0;j<6;j++)
                    { *(d+j+6)=*(tpp0+mb+n2+j);  //tp0^(it-1)
                      *(d+j)=*(tpp+mb+n2+j); }   // +N_1
                  printf("\n\t accur: m=%u, %u\n",m,k);
                  pslfn2 (c,d,1);
                  for (j=0;j<6;j++) *(tpp0+mb+n2+j)=*(c+j); // Tpp0^{it}(p)

                  orig1 (d,c,1);
                  for (j=0;j<6;j++) *(tpt+mb+n2+j)=*(d+j);  // tp1^(it)(t)
                }
            }
          printf("\n\tshnek0:  NonLinear part: tpp0(mm,nn;6) --
Tp(it=%u;%u,%u,p)+N_1(p)\n",it,mm,nn);
          for (m=0;m<mm;m++)
            { mb=6*m*mm;
              for (k=0;k<nn;k++)
                { n2=6*k;
                  for (j=0;j<6;j++) printf("%10.4g",*(tpp0+mb+n2+j)); PRN; }
              PRN; }
          printf("\n\tshnek0:  NonLinear part: tpt(mm,nn;6) --
Tp(it=%u;%u,%u,t)\n",it,mm,nn);
          for (m=0;m<mm;m++)
            { mb=6*m*mm;
              for (k=0;k<nn;k++)
                { n2=6*k;
                  for (j=0;j<6;j++) printf("%10.4g",*(tpt+mb+n2+j)); PRN; }
              PRN; }

          n2=6*mm*nn;
          if ( it==1 )
```

```
           { fcl = fopen("shn1.dat","w");
             for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tpt+k));
             fclose(fcl); }
           if ( it==2 )
            { fcl = fopen("shn2.dat","w");
             for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tpt+k));
             fclose(fcl); }
           if ( it==3 )
            { fcl = fopen("shn3.dat","w");
             for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tpt+k));
             fclose(fcl); }

           dlt= fabs(accur (tpt, mm, nn, it));
           // it+=1;

        }

       free(e);
       free(d); free(bsi);
       free(z); free(c);
       free(b);
       free(tpp); free(tpp0);
       return(nm);
     }
  }

 int trnsf_con (long double *vrz, long double *vr, long double *vz, long
double *vt, long double *bs, long double *bz, int mm, int nn)
  {
    /* conv_member: L{v_rdF(t)/dr+v_zdF(t)/dz}=vrz(mm,nn;p).
       bs -- intbs3(mm,mm); bz -- intz3(nn,nn);
       vt -- F(t).
    */
    int i, j, m, m1, mb, mb1, m2, m3, n3, k, k2, n, n1, n2, nb;
    long double b0, *e, *b, *c, *d, *f, *a;
    a=(long double *)calloc(10,sizeof(long double));
    b=(long double *)calloc(100,sizeof(long double));
    c=(long double *)calloc(30,sizeof(long double));
    d=(long double *)calloc(20,sizeof(long double));
    e=(long double *)calloc(70,sizeof(long double));
    f=(long double *)calloc(20,sizeof(long double));
    if ( a==NULL || f==NULL || c==NULL || b==NULL || d==NULL || e==NULL )
     { printf("\n\t arrays in trnsf_con = 0\n");
       return(-1); }
    else
     {
       m3=mm*mm;
       n3=nn*nn;
       n2=6*mm*nn;

       printf("\n\t vr(%u,%u)\n",mm,nn);
       for (i=0;i<n2;i++)
           printf("%10.4g%c",*(vr+i), (i%6==5) ? '\n' : ' ');
       printf("\n\t vz(%u,%u)\n",mm,nn);
       for (i=0;i<n2;i++)
           printf("%10.4g%c",*(vz+i), (i%6==5) ? '\n' : ' ');
```

```
        printf("\n\t vt(%u,%u)\n",mm,nn);
        for (i=0;i<n2;i++)
            printf("%10.4g%c",*(vt+i), (i%6==5) ? '\n' : ' ');

        for (m=0;m<mm;m++)
          { m2=m*mm;
            mb=6*m*mm;
            for (n=0;n<nn;n++)
              { n2=n*nn;
                nb=6*n;

                for (m1=0;m1<mm;m1++)
                  { n1=m1*m;
                    mb1=6*m1*mm;
                    for (k=0;k<nn;k++)
                      { k2=6*k;
                        b0=*(bs+m3+m2+m1)*(*(bz+n2+k));
                 //   printf("\n\t trans_com: m=%u, n=%u; m3+m2=%u n2=%u:
b0=%11.5g\n",m,n,m3+m2+m1,n2+k,b0);
                        for (i=0;i<6;i++) *(a+i)= (i<3 ) ? *(vr+mb1+k2+i)*b0
: *(vr+mb1+k2+i);
                        for (i=0;i<6;i++) *(c+i)=*(vt+mb1+k2+i);

                        atbt2p (d,a,c,0,0);
                        for (i=0;i<6;i++) *(e+k2+i)=*(d+i);
                      }
                   // printf("\n\t trans_com: m=%u, n=%u; mb1+ k2=%u
\n",m,n,mb1+k2);
                    pslfn (f,e,nn,0);
                    for (k=0;k<nn;k++)
                      { k2=6*k;
                        b0=*(bs+m2+m1)*(*(bz+n3+n2+k));
                        for (i=0;i<6;i++) *(a+i)= (i<3 ) ? *(vz+mb1+k2+i)*b0
: *(vz+mb1+k2+i);
                        for (i=0;i<6;i++) *(c+i)=*(vt+mb1+k2+i);
                        atbt2p (d,a,c,0,0);
                        for (i=0;i<6;i++) *(e+k2+i)=*(d+i);
                      }
                    pslfn (d,e,nn,0);
                    for (i=0;i<6;i++) *(f+i+6)=*(d+i);
                    pslfn2 (d,f,0);
                    k2=6*m1;
                    for (i=0;i<6;i++) *(b+k2+i)=*(d+i);
                  }
              //  printf("\n\t trans_com: m=%u, n=%u \n",m,n);
                pslfn (d,b,mm,0);
                for (i=0;i<6;i++) *(vrz+mb+nb+i)=*(d+i);
              }
          }
        printf("\n\t  vrz(%u,%u;p)=vr*dF(t)/dr+vz*dF(t)/dz\n",mm,nn);
        n2=6*mm*nn;
        for (i=0;i<n2;i++)
            printf("%10.4g%c",*(vrz+i), (i%6==5) ? '\n' : ' ');
        free(f); free(e);
        free(d); free(c);
        free(b); free(a);
```

```
       return(1);
     }
  }

 void read_intg (long double *bsi, long double *z2, long double *bs, long
double *z, int mm, int nn, int lo)
  {
    FILE *fcl;
    int i, m1, m2, m3, n2, n1;
        /* Input: values on bound with kropus: br=V^{m,n}_6(p).
           Output: Tp(m,n,t) -- array bt(m,n,6;t); lo=1 -- print */

         // Intg. transforms for bessel2.
    m2=mm*mm;
    n2=nn*nn;
    fcl = fopen("bsnp3.dat","r");
    for (i=0; i<2*m2; i++) fscanf(fcl,"%le,",(bsi+i));
    fclose(fcl);

    if ( lo==1 )
     { printf("\n\tread_intg:
bsnp3(%u)=int[R_m*R_l*R_i]rdr+int[R_m*R_l*dR_i/dr]rdr\n",2*m2);
       for (i=0;i<2*m2;i++)
       printf("%11.4g%c",*(bsi+i), (i%mm==mm-1) ? '\n' : ' ');
     }
    m3=3*mm;
    m1=m3+2*m2;
    fcl = fopen("bsp1.dat","r");
    for (i=0; i<m1; i++) fscanf(fcl,"%le,",(bs+i));
    fclose(fcl);
    if ( lo==1 )
     { printf("\n\tread_intg: bs(%u)=sl+int[R_m]\n",m2);
       for (i=0;i<m1;i++) printf("%11.4g%c",*(bs+i), (i%mm==mm-1) ? '\n' : '
');
     }
    fcl = fopen("zint3.dat","r");
    for (i=0; i<2*n2; i++) fscanf(fcl,"%le,",(z2+i));
    fclose(fcl);
    if ( lo==1 )
     { printf("\n\tread_intg: z3(%u)=int[Z_n*Z_k*Z_j]\n",n2);
       for (i=0;i<2*n2;i++) printf("%11.4g%c",*(z2+i), (i%nn==nn-1) ? '\n' :
' ');
     }
    n1=3*nn+2*n2;
    fcl = fopen("zint1.dat","r");
    for (i=0; i<n1; i++) fscanf(fcl,"%le,",(z+i));
    fclose(fcl);
    if ( lo==1 )
     { printf("\n\tread_intg: zint1: z(%u)=int[Z_n]+int[Z_n*Z_k]\n",n1);
       for (i=0;i<n1;i++)  printf("%11.4g%c",*(z+i), (i%nn==nn-1) ? '\n' : '
');
     }
  }

int yprv3 (long double *, long double *, long double *, int);
```

```
 int plav_pol (long double *vr1, long double *vz1, long double *tp1, int mm,
int nn)
  {
       FILE *fcl;
    /* bt - self-values; bpt -- T_p(mm,nn,t).
       vr^1(m,n,t);  vz^1(m,n,t);     */

    int i, j, n, m, k,l, k2, nb, n1, nm, m2, n2, m6, n6, mb, mb1, lp, m1,
it;
    long double *vr2, *vz2, *tp2, *vr0, *vz0, *tp00, *tp01, *z, *z3, *bs3,
*bs, *b, *c, *d, *e, *f, *bt, *bsd, *zd;
    long double v0, vp0, fi, r1, r2, s1, s2, d1, bz, bv1, bt1, rop, ros, tc,
tp, trg, trz, rz0, c0;
    long double blk, blp, bls, bl, cv0, alv, alp, alq, gr, gr0, ro, ra, rb,
zl, z0, z1, t0, h, mu0, rr;
    long double dz, dt, dr, zz, tt, tt0, tl, rs, rc, f1, eps=0.05, dlt, b0,
b1, b2, b3;

    tp2=(long double *)calloc(300,sizeof(long double));
    tp00=(long double *)calloc(70,sizeof(long double));
    tp01=(long double *)calloc(70,sizeof(long double));
    bt=(long double *)calloc(70,sizeof(long double));
    vz2=(long double *)calloc(300,sizeof(long double));
    vr2=(long double *)calloc(300,sizeof(long double));
    vr0=(long double *)calloc(70,sizeof(long double));
    vz0=(long double *)calloc(70,sizeof(long double));
    z3=(long double *)calloc(100,sizeof(long double));
    b=(long double *)calloc(100,sizeof(long double));
    c=(long double *)calloc(60,sizeof(long double));
    z=(long double *)calloc(100,sizeof(long double));
    bs3=(long double *)calloc(100,sizeof(long double));
    bs=(long double *)calloc(100,sizeof(long double));
    d=(long double *)calloc(100,sizeof(long double));
    e=(long double *)calloc(70,sizeof(long double));
    f=(long double *)calloc(70,sizeof(long double));
    if ( tp2==NULL || tp00==NULL || tp01==NULL || z3==NULL || vr0==NULL ||
vz0==NULL || bs3==NULL || bs==NULL || c==NULL || b==NULL || z==NULL ||
vr2==NULL || vz2==NULL )
     { printf("\n\t arrays in shn10=0\n");
       return(-1); }
    else
     {
       read_intg (bs3, z3, bs, z, mm, nn, 1);

           // parameters for polimer
       mu0=10.5;
       ro=420;
       alv=mu0/ro;

       zl=2;
       ra=0.3;
       rb=0.8;
       t0=120;
       h=0.3;
       v0=10;
       fi=0.18;
```

```
      // printf("\n\t shnek zl=%5.2f: lb=%9.3g; cv0=%9.3g; rho=%9.3g;
al=%10.4g\n",zl,blp,cv0,ro,alv);
       for (m=0;m<mm;m++)
         { m2=m*mm;
           mb=6*m*mm;
           s1=*(bs+m);
           for (n=0;n<nn;n++)
             { n2=n*nn;
               z1=*(z+n);
               *(bt+m2+n)=(s1*s1+z1*z1);
               r2=*(z+n+nn)*(*(bs+m+mm));
             //  printf("\n\t r2(%u,%u)=%11.4g\n",m,n,r2);
               *(vz0+m2+n)=v0*cos(fi)*r2;
               *(vr0+m2+n)=v0*sin(fi)*r2;
             }
         }
       printf("\n\t sigma_v = bt(%u,%u)\n",mm,nn);
       for (m=0;m<mm;m++)
         { mb=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(bt+mb+j)*alv); PRN; }
       printf("\n\t Linear part for v: v0=%8.3g\n",v0);
       printf("\n\t vr(%u,%u)\n",mm,nn);
       n2=mm*nn;
       for (i=0;i<n2;i++)
           printf("%10.4g%c",*(vr0+i), (i%6==5) ? '\n' : ' ');
       printf("\n\t vz(%u,%u)\n",mm,nn);
       for (i=0;i<n2;i++)
           printf("%10.4g%c",*(vz0+i), (i%6==5) ? '\n' : ' ');
       rr=1;
       blp=0.185;
       bls=0.345;
       cv0=0.45;  // KPa c
       rop=150;
       ros=300;
       alp=bls/(cv0*ros);
       tc=180;
       tp=130;
       zl=2;
       z0=1.;
       ra=0.3;
       rb=0.8;
       t0=120;
       h=0.3;
       printf("\n polT1: z0=%5.2f;zl=%5.2f: blp=%9.3g; cv0=%9.3g; rho=%9.3g;
alv=%10.4g; alp=%10.4g\n",z0,zl,blp,cv0,ros,alv,alp);
       for (m=0;m<mm;m++)
         { m2=m*mm;
           s1=*(bs+m);
           *(d+m)=fxs ((bs+m),ra,h,ra,0,1);
           *(c+m)=fxs ((bs+m),ra,h,rb,0,1);
         //  printf("\n\t d(%u,s1=%10.4g)=%11.5g;
c(%u)=%11.5g\n",m,s1,*(d+m),m,*(c+m));

           trg=*(c+m)*tc-(*(d+m))*tp;

           for (n=0;n<nn;n++)
```

```
            { n2=2*n;
              z1=*(z+n);
              bt1=*(bt+m2+n)*alp;
              rz0=t0*(*(bs+m+mm))*(*(z+n+nn));
              trz=*(z+n+nn)*trg;
        //  printf("\n\t trg(%u)=%11.5g; rz0(%u,%u)=%11.5g;
trz/bt=%11.5g\n",m,trg,m,n,rz0,trz/bt1);
              *(tp00+m2+n)=trz/bt1;
             // printf("\n\t rz0(%u,%u)=%10.4g;
tp00(%u,%u)=%10.4g\n",m,n,rz0,m,n,trz/bt1);

              *(tp01+m2+n)=(rz0-trz/bt1); // tp0=t0/p+t1/(p+alp).
            }
        }
       printf("\n\t R_m(%6.2f)\n",ra);
       for (j=0;j<mm;j++) printf("%10.4g",*(d+j)); PRN;
       printf("\n\t R_m(%6.2f)\n",rb);
       for (j=0;j<mm;j++) printf("%10.4g",*(c+j)); PRN;
       printf("\n\t sigma_v = bt_v(%u,%u)*alv=%10.4g; \n",mm,nn,alv);
       for (m=0;m<mm;m++)
         { m2=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(bt+m2+j)*alv); PRN; }
       printf("\n\t sigma_t = bt_t(%u,%u)*alp=%10.4g; \n",mm,nn,alp);
       for (m=0;m<mm;m++)
         { m2=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(bt+m2+j)*alp); PRN; }

       printf("\n\t alp=%10.4g;
tp0(%u,%u)=tp01/p+tp02/(p+alp)\n",alp,mm,nn);
       m2=mm*nn;
       printf("\n\t  tp00(%u,%u)=tp00/p+tp01/(p+alp)\n",mm,nn);
       for (m=0;m<mm;m++)
         { m2=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(tp00+m2+j)); PRN; }
       PRN;
       nm=mm*mm;
       for (m=0;m<mm;m++)
         { m2=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(tp01+m2+j)); PRN; }
      /*
       fcl = fopen("tp00.dat","w");
       for (k=0; k<nm; k++) fprintf(fcl,"%g,",*(tp00+k));
       fclose(fcl);
       fcl = fopen("tp01.dat","w");
       for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tp01+k));
       fclose(fcl);
      */
       n6=nn*nn;
       for (m=0; m<mm; m++)
         { m2=m*mm+3*mm;
           mb1=6*m*mm;
           for (n=0;n<mm;n++)
             {
               n2=n*nn+3*nn;
               for (m1=0;m1<mm;m1++)
                 { n1=m1*mm;
```

```
            for (k=0;k<nn;k++)
              { k2=6*k;
                bt1=*(bt+n1+k)*alp;
                bv1=*(bt+n1+k)*alv;
                b1=bv1*bt1;
                b0=*(bs+nm+m2+m1)*(*(z+n2+k))/2.;
                r1=b0*rr*(*(tp00+n1+k))/bv1;
                *(d+k2+2)=-r1;
                b2=(*(vr0+n1+k)+r1);
                *(d+k2)=b2/bv1-b0*rr*(*(tp01+n1+k))/b1;
                *(d+k2+1)=b2;
                *(d+k2+5)=1;
                *(d+k2+4)=1/bv1+1/bt1;
                *(d+k2+3)=1/b1;
              }
         //   printf("\n\tm=%u; n=%u; m1=%u\n",m,n,m1);
            pslfn (c,d,nn,0);
            k2=6*m1;
            for (i=0;i<6;i++) *(b+i+k2)=*(c+i);
          }    // -dp/dz =-rr*tp0
        pslfn (c,b,mm,0);
        k2=mb1+6*n;
   //   printf("\n\tm=%u; n=%u vr1(%u,t)\n",m,n,k2);
        for (i=0;i<6;i++) *(vr1+k2+i)=*(c+i);
        orig1 (e,c,0);
        for (i=0;i<6;i++) *(vr2+k2+i)=*(e+i);
      }
  }
 printf("\n\t plav_pol: v0=%8.3g; vr1(%u,%u;p)\n",v0,mm,nn);
 n2=6*mm*nn;
 for (i=0;i<n2;i++)
    printf("%10.4g%c",*(vr1+i), (i%6==5) ? '\n' : ' ');
/*
 fcl = fopen("vrp1.dat","w");
 for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vr1+k));
 fclose(fcl);
*/
 printf("\n\t  plav_pol: vrt1(%u,%u;t)\n",mm,nn);
 for (i=0;i<n2;i++)
    printf("%10.4g%c",*(vr2+i), (i%6==5) ? '\n' : ' ');
 /*
 fcl = fopen("vrt1.dat","w");
 for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vr2+k));
 fclose(fcl);
 */
 for (i=0;i<n2;i++) *(vr2+i)=0.;
 for (m=0; m<mm; m++)
   { m2=m*mm+3*mm;
     mb1=6*m*mm;
     for (n=0;n<nn;n++)
       {
         n2=n*nn+3*nn;
         for (m1=0;m1<mm;m1++)
           { n1=m1*mm;

             for (k=0;k<nn;k++)
```

```
                { k2=6*k;
                  bt1=*(bt+n1+k)*alp;
                  bv1=*(bt+n1+k)*alv;
                  b1=bv1*bt1;
                  b0=*(bs+m2+m1)*(*(z+n6+n2+k))/2.;
                  r1=b0*rr*(*(tp00+n1+k))/bv1;
                  *(d+k2+2)=-r1;
                  b2=(*(vz0+n1+k)+r1);

                  *(d+k2)=(b2/bv1-b0*rr*(*(tp01+n1+k)))/b1;
                  *(d+k2+1)=b2;
                  *(d+k2+3)=1;
                  *(d+k2+4)=1/bv1+1/bt1;
                  *(d+k2+5)=1/b1;
                }
              // printf("\n\tm=%u; n=%u; m1=%u\n",m,n,m1);
               pslfn (c,d,nn,0);
               k2=6*m1;
               for (i=0;i<6;i++) *(b+i+k2)=*(c+i);
            }    // -dp/dz =-rr*tp0
          pslfn (c,b,mm,0);
          k2=mb1+6*n;
      // printf("\n\tm=%u; n=%u vz1(%u)\n",m,n,k2);
          for (i=0;i<6;i++) *(vz1+k2+i)=*(c+i);
          orig1 (e,c,0);
          for (i=0;i<6;i++) *(vz2+k2+i)=*(e+i);
        }
    }

printf("\n\t  plav_pol: v0=%8.3g; vz1(%u,%u;p)\n",v0,mm,nn);
n2=6*mm*nn;
for (i=0;i<n2;i++)
    printf("%10.4g%c",*(vz1+i), (i%6==5) ? '\n' : ' ');
printf("\n\t file: vzt1(%u,%u;t); alv=%10.4g;\n",mm,nn,alv);
for (i=0;i<n2;i++)
    printf("%10.4g%c",*(vz2+i), (i%6==5) ? '\n' : ' ');

fcl = fopen("vzp1.dat","w");
for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vz1+k));
fclose(fcl);
fcl = fopen("vzt1.dat","w");
for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vz2+k));
fclose(fcl);

it=1;

trnsf_con (vr2,vr1,vz1,vr1,bs3,z3,mm,nn);
trnsf_con (vz2,vr1,vz1,vz1,bs3,z3,mm,nn);

alq=5*mu0/(ros*cv0);

for (m=0;m<mm;m++)
  { m2=m*mm;
    mb1=6*m*mm;
    for (n=0;n<nn;n++)
      {
```

```
                n2=6*n;
                for (k=0;k<6;k++) *(b+k)=*(vr2+mb1+n2+k);
                *d=-1;
                *(d+1)=*(bt+m2+n)*alv;
                yprv2 (c,d,b,0);
                for (k=0;k<6;k++) *(c+k+6)=*(vr1+mb1+n2+k);

                pslfn2 (d,c,0);
                for (k=0;k<6;k++) *(vr2+mb1+n2+k)=*(d+k);

                *d=-1;
                *(d+1)=*(bt+m2+n)*alv;
                for (k=0;k<6;k++) *(b+k)=*(vz2+mb1+n2+k);
                yprv2 (c,d,b,0);
                for (k=0;k<6;k++) *(c+k+6)=*(vz1+mb1+n2+k);
                pslfn2 (d,c,0);
                for (k=0;k<6;k++) *(vz2+mb1+n2+k)=*(d+k);
              }
          }
        n2=6*mm*nn;
        printf("\n\t  plav_pol: vr2(%u,%u;p); alv=%10.4g\n",mm,nn,alv);
        for (i=0;i<n2;i++) printf("%10.4g%c",*(vr2+i), (i%6==5) ? '\n' : '
');
        printf("\n\t  vz2(%u,%u;p)\n",mm,nn);
        for (i=0;i<n2;i++) printf("%10.4g%c",*(vz2+i), (i%6==5) ? '\n' : '
');

                printf("\n\t  obtain Q_T1(%u,%u); alq=%10.4g\n",m,n,alq);

        for (m=0;m<mm;m++)
          { m2=m*mm;
            mb=6*m*mm;
            for (n=0;n<nn;n++)
              {
                n2=6*n;
                for (m1=0;m1<mm;m1++)
                  { n1=m1*mm;
                    mb1=6*m1*mm;
                    for (l=0;l<nn;l++)
                      { k2=6*l;

                        b0=*(bs3+nm+m2+m1)*(*(z3+n2+l))*alq;

                        for (k=0;k<6;k++) *(b+k) = *(vr2+mb1+k2+k);
                        atbt2p (c,b,b,0,0);
                        for (k=0;k<3;k++) *(c+k)*=b0;
                       //  for (k=0;k<6;k++) printf("%11.4g",*(c+k)); PRN;
                        b0=*(bs3+m2+m1)*(*(z3+n6+n2+l))*alq;
                        for (k=0;k<6;k++) *(b+k)=*(vz2+mb1+k2+k);
                        atbt2p (e,b,b,0,0);
                        for (k=0;k<3;k++) *(e+k)*=b0;
                       // for (k=0;k<6;k++) printf("%11.4g",*(e+k)); PRN;
                        for (k=0;k<6;k++) *(c+k+6) = *(e+k);
                 // printf("\n\tm=%u, n=%u; m1=%u, l=%u\n",m,n,m1,l);
                        pslfn2 (z,c,0); //  z_[m_1,l] =
(dv_r/dr)^2+(dv_z/dz)^2.
```

```
                    for (k=0;k<6;k++) *(c+k+k2) = *(z+k);
                  }
             // printf("\n\tm=%u, n=%u; m1=%u\n",m,n,m1);
                  pslfn (d,c,nn,0);
                  for (k=0;k<6;k++) *(f+k+n1) = *(d+k);
                }
           //   printf("\n\tm=%u, n=%u\n",m,n);
              pslfn (z,f,mm,0);
              *(d+1)=alq;
               *d=*(tp01+m2+n);
              *(d+2)=*(bt+m2+n)*alp;
              yprv3 (b,d,z,0);
              for (k=0;k<6;k++) *(tp2+mb+n2+k)=*(b+k);

            }
        }
      n2=6*mm*nn;
      printf("\n\t  plav_pol: tp2(p)=tp0(p)+Q_T(p); alq=%10.4g\n",alq);
      for (k=0;k<n2;k++) printf("%11.4g%c",*(tp2+k), (k%6==5) ? '\n' : '
');
      fcl = fopen("tpp1.dat","w");
      for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tp2+k));
      fclose(fcl);
    /*  fcl = fopen("tpt11.dat","w");
      for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tp2+k));
      fclose(fcl);
     */
      printf("\n\t vr2(p) -> vr2t(t);  vz2(p) -> vz2t(t)  tp2(p) ->
tp2t(t)\n");

      for (m=0;m<mm;m++)
        { m2=m*mm;
          mb1=6*m*mm;
          for (n=0;n<nn;n++)
            {
              n2=6*n;
              for (k=0;k<6;k++) *(b+k)=*(vr2+mb1+n2+k);
              orig1 (c,b,0);
              for (k=0;k<6;k++) *(vr2+mb1+n2+k)=*(c+k);
              for (k=0;k<6;k++) *(b+k)=*(vz2+mb1+n2+k);
              orig1 (c,b,0);
              for (k=0;k<6;k++) *(vz2+mb1+n2+k)=*(c+k);
              for (k=0;k<6;k++) *(b+k)=*(tp2+mb1+n2+k);
              orig1 (c,b,0);
              for (k=0;k<6;k++) *(tp2+mb1+n2+k)=*(c+k);
            }
        }
      n2=6*mm*nn;
      printf("\n\t  plav_pol:  vr12(%u,%u;t)\n",mm,nn);
      for (i=0;i<n2;i++)
          printf("%10.4g%c",*(vr2+i), (i%6==5) ? '\n' : ' ');
      printf("\n\t  plav_pol:file: vz12(%u,%u;t)\n",mm,nn);
      for (i=0;i<n2;i++)
          printf("%10.4g%c",*(vz2+i), (i%6==5) ? '\n' : ' ');
      printf("\n\t  plav_pol: file: tp12(%u,%u;t)\n",mm,nn);
      for (i=0;i<n2;i++)
```

```
            printf("%10.4g%c",*(tp2+i), (i%6==5) ? '\n' : ' ');
        /*
         fcl = fopen("vrt12.dat","w");
         for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vr2+k));
         fclose(fcl);
         fcl = fopen("vzt12.dat","w");
         for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vz2+k));
         fclose(fcl);
         fcl = fopen("tpt12.dat","w");
         for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tp2+k));
         fclose(fcl);
          */
         free(f);
         free(e); free(d);
         free(bs); free(bs3);
         free(z); free(c);
         free(b); free(z3);
         free(vr0); free(vz0);
         free(vr2); free(vz2);
         free(bt); free(tp01);
         free(tp00); free(tp2);
         return(it);
       }
  }


 int nlpol ( long double *vr2, long double *vz2, long double *tp2, int mm,
int nn)
  {
    FILE *fcl;
   /* obtain nonlinear part of melting
      vr1(p),vz1(p),tp1(p) --  velosity and temperature on it-iteration
      vr2(p),vz2(p),tp2(p) --  velosity and temperature on (it+1)-iteration
   */
    int i, k, l, m, n, m1, mb, mb1, m2, m3, n3, n1, n2, k2, it;

    long double *z, *z3, *bs3, *bs, *b, *c, *d, *e, *f, *bt, b0, s1, z1;
    long double *vr1, *vz1, *tp1, cv0, ro, mu0, rop, ros, alp, alq, alv,
bls;

    vr1=(long double *)calloc(300,sizeof(long double));
    vz1=(long double *)calloc(300,sizeof(long double));
    tp1=(long double *)calloc(300,sizeof(long double));

    bt=(long double *)calloc(40,sizeof(long double));
    z3=(long double *)calloc(140,sizeof(long double));
    b=(long double *)calloc(200,sizeof(long double));
    c=(long double *)calloc(60,sizeof(long double));
    z=(long double *)calloc(100,sizeof(long double));
    bs3=(long double *)calloc(140,sizeof(long double));
    bs=(long double *)calloc(130,sizeof(long double));
    d=(long double *)calloc(10,sizeof(long double));
    e=(long double *)calloc(70,sizeof(long double));
    f=(long double *)calloc(70,sizeof(long double));
```

```
    if ( tp1==NULL || z3==NULL || vr1==NULL || vz1==NULL || bs3==NULL ||
bs==NULL || c==NULL || bt==NULL || b==NULL || z==NULL  || d==NULL || e==NULL
|| f==NULL )
     { printf("\n\t arrays in nlpol=0\n");
       return(-1); }
    else
     { printf("\n\t nlpol\n");
       read_intg (bs3, z3, bs, z, mm, nn, 1);
       m2=6*mm*nn;
       fcl = fopen("vrp1.dat","r");
       for (i=0; i<m2; i++) fscanf(fcl,"%le,",(vr1+i));
       fclose(fcl);
       fcl = fopen("vzp1.dat","r");
       for (i=0; i<m2; i++) fscanf(fcl,"%le,",(vz1+i));
       fclose(fcl);
       fcl = fopen("tpp1.dat","r");
       for (i=0; i<m2; i++) fscanf(fcl,"%le,",(tp1+i));
       fclose(fcl);
       printf("\n\tnlpol:  vr1(%u,%u;p)\n",mm,nn);
       for (i=0;i<m2;i++) printf("%10.4g%c",*(vr1+i), (i%6==5) ? '\n' : '
');
       printf("\n\t nlpol:  vz1(%u,%u;p)\n",mm,nn);
       for (i=0;i<m2;i++) printf("%10.4g%c",*(vz1+i), (i%6==5) ? '\n' : '
');
       printf("\n\t nlpol: tp1(%u,%u;p)\n",mm,nn);
       for (i=0;i<m2;i++) printf("%10.4g%c",*(tp1+i), (i%6==5) ? '\n' : '
');
       bls=0.345;
       cv0=0.45;  // KPa c
       rop=3*77.9;
       ros=91.9;
       alp=bls/(cv0*ros);

       mu0=10.5;
       ro=920;
       alq=5*mu0/(ros*cv0);    // 5*
       alv=mu0/rop;
       it=1;

       trnsf_con (vr2,vr1,vz1,vr1,bs3,z3,mm,nn);
       trnsf_con (vz2,vr1,vz1,vz1,bs3,z3,mm,nn);
       trnsf_con (tp2,vr1,vz1,tp1,bs3,z3,mm,nn);

       for (m=0;m<mm;m++)
         { m2=m*mm;
           mb=6*m*mm;
           s1=*(bs+m);
           for (n=0;n<nn;n++)
             { n2=n*nn;
               z1=*(z+n);
               *(bt+m2+n)=(s1*s1+z1*z1);
               for (k=0;k<6;k++) *(b+k)=*(vr2+mb+n2+k);
               *d=-1;
               *(d+1)=*(bt+m2+n)*alv;
               yprv2 (c,d,b,0);
               for (k=0;k<6;k++) *(c+k+6)=*(vr1+mb+n2+k);
```

```
                pslfn2 (b,c,0);
                for (k=0;k<6;k++) *(vr2+mb+n2+k)=*(b+k);
                *d=-1;
                *(d+1)=*(bt+m2+n)*alv;
                for (k=0;k<6;k++) *(b+k)=*(vz2+mb+n2+k);
                yprv2 (c,d,b,0);
                for (k=0;k<6;k++) *(c+k+6)=*(vz1+mb+n2+k);
                pslfn2 (b,c,0);
                for (k=0;k<6;k++) *(vz2+mb+n2+k)=*(b+k);
              }
          }
        n2=6*mm*nn;
        printf("\n\t  vr2(%u,%u;p)=vr1(p)-Con vr1(p)\n",mm,nn);
        for (i=0;i<n2;i++) printf("%10.4g%c",*(vr2+i), (i%6==5) ? '\n' : '
');
        printf("\n\t  vz2(%u,%u;p)=vz1(p)-Con vz1(p)\n",mm,nn);
        for (i=0;i<n2;i++) printf("%10.4g%c",*(vz2+i), (i%6==5) ? '\n' : '
');

                printf("\n\t  obtain Q_T(%u,%u)\n",m,n);
        m3=mm*mm;
        n3=nn*nn;
        for (m=0;m<mm;m++)
          { m2=m*mm;
            mb=6*m*mm;
            for (n=0;n<nn;n++)
              {
                n2=6*n;
                for (m1=0;m1<mm;m1++)
                  { n1=m1*mm;
                    mb1=6*m1*mm;
                    for (l=0;l<nn;l++)
                      { k2=6*l;

                        b0=*(bs3+m3+m2+m1)*(*(z3+n2+l));
                        for (k=0;k<6;k++) *(b+k) = *(vr2+mb1+k2+k);
                        atbt2p (c,b,b,0,0);
                        for (k=0;k<3;k++) *(c+k) =*(c+k)*b0;
                       //  printf("\n vr2^2(%u,%u)=",m1,l);
                        // for (k=0;k<6;k++) printf("%11.4g",*(c+k)); PRN;

                        b0=*(bs3+m2+m1)*(*(z3+n3+n2+l));

                        for (k=0;k<6;k++) *(b+k)=*(vz2+mb1+k2+k);
                        atbt2p (e,b,b,0,0);
                        for (k=0;k<3;k++) *(e+k)*=b0;
                       // printf("\n vz2^2(%u,%u)=",m1,l);
                       // for (k=0;k<6;k++) printf("%11.4g",*(e+k)); PRN;

                        for (k=0;k<6;k++) *(c+k+6) = *(e+k);
                        pslfn2 (z,c,0); //  z_[m_1,l] =
(dv_r/dr)^2+(dv_z/dz)^2.
                        for (k=0;k<6;k++) *(c+k+k2) = *(z+k);
                      }
                  // printf("\n\t vr2^2+vz2^2(%u,%u;%u)\n",m,n,m1);
                    pslfn (d,c,nn,0);
```

```
                for (k=0;k<6;k++) *(f+k+n1) = *(d+k);
              }
             //     printf("\n\t   sum_{m_1,l} Q_T(%u,%u)",m,n);
             pslfn (e,f,mm,0);
             *d=alq;
             *(d+1)=*(bt+m2+n)*alp;
             yprv2 (c,d,e,0);  // alq/(p+alp)*Q_T(p)
             for (k=0;k<6;k++) *(c+k+6)=*(tp1+mb+n2+k);
             *d=-1;
             *(d+1)=*(bt+m2+n)*alp;
             for (k=0;k<6;k++) *(b+k)=*(tp2+mb+n2+k);
             yprv2 (e,d,b,0);  // -1/(p+alp)*Cp2(p)
             for (k=0;k<6;k++) *(c+k+12)=*(z+k);
             pslfn (b,c,3,0);
             for (k=0;k<6;k++) *(tp2+mb+n2+k)=*(b+k);
            // printf("\n tp2(%u,%u;p)=tp1(p)+Con tp1(p)",mb,n2);
            // for (k=0;k<6;k++) printf("%11.4g",*(b+k));
           }
       }
     n2=6*mm*nn;
     printf("\n\t  tp2(%u,%u;p)=tp1(p)+Q_T(p)-Con tp1(p);
alq=%10.4g\n",mm,nn,alq);
     for (i=0;i<n2;i++) printf("%10.4g%c",*(tp2+i), (i%6==5) ? '\n' : '
');

     printf("\n\t vr2(p) -> vr2t(t);  vz2(p) -> vz2t(t)  tp2(p) ->
tp2t(t)\n");

     for (m=0;m<mm;m++)
       { m2=m*mm;
         mb=6*m*mm;
         for (n=0;n<nn;n++)
           {
             n2=6*n;
             for (k=0;k<6;k++) *(b+k)=*(vr2+mb+n2+k);
             orig1 (c,b,0);
             for (k=0;k<6;k++) *(vr2+mb+n2+k)=*(c+k);
             for (k=0;k<6;k++) *(b+k)=*(vz2+mb+n2+k);
             orig1 (c,b,0);
             for (k=0;k<6;k++) *(vz2+mb+n2+k)=*(c+k);
             for (k=0;k<6;k++) *(b+k)=*(tp2+mb+n2+k);
             orig1 (c,b,0);
             for (k=0;k<6;k++) *(tp2+mb+n2+k)=*(c+k);
           }
       }

     n2=6*mm*nn;
     printf("\n\t nlpol:  vr2t.dat:  vrt2(%u,%u;t)\n",mm,nn);
     for (i=0;i<n2;i++)
         printf("%10.4g%c",*(vr2+i), (i%6==5) ? '\n' : ' ');
     printf("\n\t nlpol:  vz2t.dat: vzt2(%u,%u;t)\n",mm,nn);
     for (i=0;i<n2;i++)
         printf("%10.4g%c",*(vz2+i), (i%6==5) ? '\n' : ' ');
     printf("\n\t nlpol:  tp2t.dat: alq=%11.4g;
tpt2(%u,%u;t)\n",alq,mm,nn);
     for (i=0;i<n2;i++)
```

```
            printf("%10.4g%c",*(tp2+i), (i%6==5) ? '\n' : ' ');
          /*
          fcl = fopen("vr2t.dat","w");
          for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vr2+k));
          fclose(fcl);
          fcl = fopen("vz2t.dat","w");
          for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(vz2+k));
          fclose(fcl);
          fcl = fopen("tp2t.dat","w");
          for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tp2+k));
          fclose(fcl);
          */
          free(f);
          free(e); free(d);
          free(bs); free(bs3);
          free(z); free(c);
          free(b); free(z3);
          free(bt); free(tp1);
          free(vz1); free(vr1);
          return(it);
        }
    }

 int cristal (long double *sb, long double *zs, int mm, int nn)
    {
      /* gomogeniz. and crist.
         tpt -- temperature of melting polimer in point r=r_2
      */

       FILE *fcl;
       static long double  a1[10], d1[10], c1[30], e1[10],
        b1[]={0.165,0.32,2.,4.58,4.2,1, 6.2,14.7,1.,3.45,3.6,1,
              12.6,9.5,4.,3.1,2,1, -2.7,1,-4,5.28,1.26,1};
      long double *bu, *au, *cu, *du, *eu;
      long double *ta, *tt, *z, *bs, *b, *c, *d, *e, *f, rc, c0, r1, r0, c2,
r2, ra, rb, x2;
      int m1, i, k, l, m, n, k1, n1, nk;
      au=a1; bu=b1; cu=c1; du=d1; eu=e1;
      ta=(long double *)calloc(300,sizeof(long double));
      tt=(long double *)calloc(300,sizeof(long double));
      b=(long double *)calloc(50,sizeof(long double));
      c=(long double *)calloc(60,sizeof(long double));
      z=(long double *)calloc(50,sizeof(long double));
      bs=(long double *)calloc(30,sizeof(long double));
      d=(long double *)calloc(10,sizeof(long double));
      e=(long double *)calloc(20,sizeof(long double));
      f=(long double *)calloc(20,sizeof(long double));
      if ( bs==NULL || c==NULL || tt==NULL || b==NULL || z==NULL  || d==NULL
|| e==NULL || f==NULL )
       { printf("\n\t arrays in nlpol=0\n");
         return(-1);
       }
      else
       {
```

```
m1=6*mm*nn;
fcl = fopen("tpt2.dat","r");
for (i=0; i<m1; i++) fscanf(fcl,"%le,",(tt+i));
fclose(fcl);

printf("\n Temperature of melt-- array tt\n");
m1=6*mm*nn;
for (i=0;i<m1;i++) printf("%10.4g%c",*(tt+i), (i%6==5) ? '\n' : ' ');

r0=0.06;
c0=.51;

rc=r0/c0;
c2=rc*rc;
printf("\n Gomogenization -- c0=%8.3f\n",c0);
r1=c2*(c0*rc-1);
*f=c2*(2-3*rc*c0);
*(f+1)=c2*(3*rc*c0-1);
*(f+2)=c2*c0*rc;
*d=rc*(6*(*f)+3*(*(f+1))+2*(*(f+2)));
*(d+1)=-c0*(5*(*f)+4*(*(f+1))+3*(*(f+2)));
*(d+2)=*f+(*(f+1))+(*(f+2));
*(d+3)=0;
*c=-6*c0*c0*c0;
*(c+1)=11*c0*c0;
*(c+2)=-6*c0;
*(c+3)=1;
printf("\n coef. c: 1-st iter.");
for (k=0;k<4;k++) printf("%10.4g",*(c+k));
for (k=0;k<3;k++) *(e+k)=*(c+k+1);
yprvn (c,d,b,4,2,1);
*(b+2)=r1;
ra=0.3;
rb=0.4;
r1=-(*(e+1)*(*(e+4))+(*(e+2))*(*(e+5)));
r2=*(e+1)*(*(e+5))-(*(e+2))*(*(e+4));
*(b)=*b-r1*(*(e+5))+r2*(*(e+4));
*(b+1)=*(b+1)-r2;
orig1 (e,b,1);
for (l=0;l<mm;l++)
  { x2=*(sb+l);
    r1=yn(1,x2*ra)/jn(1,x2*ra);
   *(bs+l)=yn(0,x2*rb)-r1*jn(0,x2*rb); }

printf("\n B(sb*rb)=");
for (k=0;k<mm;k++) printf("%10.4g",*(bs+k)); PRN;

for (m=0;m<mm;m++)
  { m1=6*m*nn;
    for (n=0;n<nn;n++)
      {
        n1=m1+6*n;
        for (k=0;k<6;k++) *(c+k)=*(tt+n1+k);
       // trns2p (d,c,e,0,1);
        ftgtp (d,c,e,0,1);
        orig1 (f,d,1);
```

```
              for (k=0;k<6;k++) *(ta+n1+k)=*(f+k);
            }

          }
        printf("\n Gomogenization -- array ta; c0=%8.3f\n",c0);
        m1=6*mm*nn;
        for (i=0;i<m1;i++) printf("%10.4g%c",*(ta+i), (i%6==5) ? '\n' : ' ');

        fcl = fopen("c:/dis_figs/crist1.dat","w");
        for (k=0; k<m1; k++) fprintf(fcl,"%g,",*(ta+k));
        fclose(fcl);

   // printf("\n Cristallization\n -- c0=%8.3f",c0);

        free(f); free(e);
        free(d); free(bs);
        free(z); free(c);
        free(b); free(tt);
        free(ta);
        return(1);
      }
  }

 long double accur (long double *tpt, int mm, int nn, int it)
  {
    FILE *fcl;
   /* Obtaining of accuracy of two iterations
      acc=int[0,T][T(m)(r,z,t) -T(m-1)(r,z,t)]^2dt. */

    int i, j, k, k2, l, l2, m, n, mb, nb, m2, m3, n2, n3;
    long double r1, r2, *a, *b, *c, *d, *z, *z2, *bs2, *zi, *bs, *bsi, dlt,
ttm, tm, tm1, tm2, tm3, tt, dlta;
    a=(long double *)calloc(10,sizeof(long double));
    b=(long double *)calloc(10,sizeof(long double));
    c=(long double *)calloc(20,sizeof(long double));
    d=(long double *)calloc(60,sizeof(long double));
    bs2=(long double *)calloc(100,sizeof(long double));
    z2=(long double *)calloc(100,sizeof(long double));
    bs=(long double *)calloc(10,sizeof(long double));
    bsi=(long double *)calloc(100,sizeof(long double));
    zi=(long double *)calloc(100,sizeof(long double));
    z=(long double *)calloc(10,sizeof(long double));
    if ( zi==NULL ||  z2==NULL || bs2==NULL ||  bs==NULL || bsi==NULL ||
d==NULL || c==NULL || b==NULL || a==NULL )
     { printf("\n\t arrays z2, bs, d, c, b, a in accur =0\n");
       return(-1); }
    else
     { n2=nn*nn;
       n3=3*nn;
       m3=3*mm;
       m2=mm*mm;
       fcl = fopen("bsp1.dat","r");
       for (i=0; i<m3+2*m2; i++) fscanf(fcl,"%le,",(bs2+i));
       fclose(fcl);
       for (j=0;j<mm;j++) *(bs+j)=*(bs2+mm+j);  // int[bs]
       for (j=0;j<m2;j++) *(bsi+j)=*(bs2+m3+j); // int [bs_m*bs_k]
```

```
        printf("\n accur: bsi(%u)=",mm);
        for (j=0;j<mm;j++) printf("%11.5g",*(bs+j)); PRN;
        fcl = fopen("zint2.dat","r");
        for (i=0; i<n2+n3; i++) fscanf(fcl,"%le,",(z2+i));
        fclose(fcl);
        printf("\n\t  accur: int z_k(%u,z)*z_k1(z)\n",n2);

        for (i=0;i<nn;i++) *(z+i)=*(z2+nn+i);
        printf("\n accur:  zi(z)=");
        for (i=0;i<nn;i++) printf("%10.4g",*(z+i));
         PRN;

        printf("\n\t accur: tpt(mm,nn;6) -- Tp(%u:%u,%u,t)\n",it,mm,nn);
        for (m=0;m<mm;m++)
          { mb=6*m*mm;
            for (k=0;k<nn;k++)
              { n2=6*k;
                for (j=0;j<6;j++) printf("%10.4g",*(tpt+mb+n2+j)); PRN; }
            PRN; }

        dlt=0.;
        ttm=0.;

        tt=1.;
        tm=0;
            tm1=0.;
            tm2=0.;
            tm3=0.;
        for (m=0;m<mm;m++)
          { mb=6*m*mm;
            m2=m*mm;

            for (n=0;n<nn;n++)
              {
                n2=6*n;
                for (j=0;j<6;j++) *(a+j)=*(tpt+mb+n2+j);
                r1=*(bs+m)*(*(z+n));
                tm+=r2=intt2 (a,0,tt,1)*r1;
            printf("\n\t accur: r1(%u,%u)=%10.4g;
r2(%u)=%10.4g\n",m,n,r1,it,r2);
              /*
                for (l=0;l<nn;l++)
                  { l2=6*l*mm;
                    for (k=0;k<nn;k++)
                      { k2=6*k;
                        r1=*(bsi+m2+l)*(*(zi+n2+k));

                        for (j=0;j<6;j++) *(b+j)=*(tpt+l2+k2+j);

                        tm+=r2=intab (a,b, 0, tt, 1)*r1;  //
T^(it)(n)*T^(it)(k)
                        printf("\n\taccur r1(%u,%u)=%12.4g --
r2=%10.4g\n",m2+l,n2+k,r1,r2);

                           for (j=0;j<6;j++) *(b+j)=*(tpt1+mb+k2+j);
```

```
                        tm1+=intab (a,b, 0, tt, 0); // *(*(zi+n2+k)); //
T^(it)(n)*T^(it-1)(k)
                    }
                }
               for (j=0;j<6;j++) *(a+j)=*(tpt1+mb+n2+j);
               for (k=0;k<nn;k++)
                 { k2=6*k;
                   for (j=0;j<6;j++) *(b+j)=*(tpt1+mb+k2+j);
                   tm2+=intab (a,b, 0, tt, 0); // *(*(zi+n2+k)); // T^(it-
1)(n)*T^(it-1)(k)
                 }

               printf("\n\t tm(%u,%u)=%10.4g\n",m,n,tm);
             */
             }
         }

       printf("\n\taccur: tm(%u)=%10.4g \n",it,fabs(tm));

       free(z); free(zi);
       free(bsi);
       free(bs); free(z2);
       free(d); free(c);
       free(b); free(a);
       return (tm);
     }
  }

 int trns2p (long double *, long double *, long double *, int, int);

 int nlshn (long double *cnp, long double *bn, long double *bt, int mm, int
nn, int it)
  {
   //  obtain L{(Cv1+Cv2*T)*T*dT/dt}=sum_{m=1,mm;n=1,nn}
Vn_{m,n}=(b_0+b_1p)/(b_3+b_4p+b_5p^2);
   //  input: bt(nn*nn) -- selfvalues; brz(mm*mm) -- integrals
Zn_1Zn_2Zn_3(r).
   // mm, nn -- size on  r and z

    FILE *fcl;

    int i, j, k, k6, l, lk, l6, m, n, n1, n2, m2, m6, n6;
    long double r1, cv0, cv1, cv2, *b, *c, *d, *e, *bsi, *bs, *zi, *zi3,
*cnt, *cs;
    b=(long double *)calloc(10,sizeof(long double));
    c=(long double *)calloc(20,sizeof(long double));
    d=(long double *)calloc(60,sizeof(long double));
    e=(long double *)calloc(60,sizeof(long double));
    bsi=(long double *)calloc(100,sizeof(long double));
    bs=(long double *)calloc(130,sizeof(long double));
    cnt=(long double *)calloc(400,sizeof(long double));
    zi=(long double *)calloc(130,sizeof(long double));
    zi3=(long double *)calloc(100,sizeof(long double));
    cs=(long double *)calloc(20,sizeof(long double));
    if ( zi==NULL || bsi==NULL || e==NULL || d==NULL || c==NULL || b==NULL
|| bs==NULL || cnt==NULL || zi3==NULL || cs==NULL)
```

```
     { printf("\n\t arrays in nlshn = 0\n");
       return(-1); }
     else
      {
       read_intg (bsi, zi3, bs, zi, mm, nn,0);

             printf("\n\t nlshn: self-values for r,z\n");

       for (j=0;j<mm;j++) *(cs+j)=*(bs+j);
       for (j=0;j<nn;j++) *(cs+j+mm)=*(zi+j);

      /*
       printf("\n\t nlshn: sigma = bt(%u,%u)\n",mm,nn);
       for (m=0;m<mm;m++)
         { m2=6*m;
           for (j=0;j<nn;j++) printf("%10.4g",*(bt+m2+j)); PRN; }
       printf("\n\t nlshn: sigma = bt(%u,%u)\n",mm,nn);
       for (m=0;m<mm;m++)
         { m2=6*m*mm;
           for (k=0;k<nn;k++)
             { n2=6*k;
               for (j=0;j<6;j++) printf("%10.4g",*(bn+m2+n2+j)); PRN; }
           PRN; }
       */

       cv0=2.5;
       cv1=-0.0024/cv0;
       cv2=0.00057/cv0;;
       printf("\n\tnlshn: cv1=%12.4g, cv2=%12.4g\n",cv1,cv2);
       for (m=0;m<mm;m++)
         { m2=m*mm;
           m6=6*mm*m;
           for (n=0;n<nn;n++)
             { n2=n*nn;
               n6=6*n;
               // printf("\n\t Sum(m1,n1)\n"):
               for (l=0; l<mm; l++)
                 { l6=6*l; //*mm;
                   lk=6*l*mm;
                   for (j=0;j<6;j++) *(c+j)=*(bn+lk+j);
                   for (k=0; k<nn; k++)
                     { k6=6*k;
                       for (j=0;j<6;j++) *(e+j)=*(bn+k6+j);
                       r1=*(bsi+m2+l)*(*(zi3+n2+k));
      //  printf("\n\tnlshn: bsi(%u,%u)=%10.4g; zi(%u,%u)=%10.4g;
r1(%u,%u)=%12.4g\n",m,l,*(bsi+m2+l),n,k,*(zi3+n2+k),m2+l,n2+k,r1);
                       for (j=0;j<6;j++) *(e+j)= (j<3 ) ? *(e+j)*r1 :
*(e+j);
                       trns2p (b,c,e,1,0);        //
b(p)=L_{t}[T(l,j)*dT(l,j)/dt];
                       for (j=0;j<6;j++) *(d+k6+j)=*(b+j);
                      //  for (j=0;j<6;j++) printf("%11.4g",*(d+j+k6));
                      }
                  //   printf("\n\t d(%u,%u,%u)=\n",m,n,l6);
                   pslfn(b,d,nn,0);
                   for (k=0;k<6;k++) *(e+k+l6)=*(b+k);
```

```
                  // for (k=0;k<6;k++) printf("%11.4g",*(b+k));
                 }
               // printf("\n\t nlshn: m=%u, n=%u\n",m,n);
               pslfn(b,e,mm,0);
               for (k=0;k<3;k++) *(b+k)*=cv1;

               *c=1;
               *(c+1)=*(bt+m2+n);
               yprv2 (d,c,b,0);
               for (k=0;k<6;k++) *(cnp+k+m6+n6)=*(d+k);  //  N_1(m,n;p)
               orig1 (c,d,0);
               for (k=0;k<6;k++) *(cnt+k+m6+n6)=*(c+k);  //  N_1(m,n;t)
             }
         }

       printf("\n\t nlshn:  N_1(m,n;p)=cnp(%u;%u,p)\n",mm,nn);
       for (m=0;m<mm;m++)
         { m6=6*m*mm;
           for (n=0;n<nn;n++)
             { n6=6*n;
               for (k=0;k<6;k++) printf("%11.4g",*(cnp+k+n6+m6)); PRN;
             }
           PRN;
         }

       printf("\n\t nlshn:  cnt(%u;%u,t)=N_1(T,dT/dt)=\n",mm,nn);
       for (m=0;m<mm;m++)
         { m6=6*m*mm;
           for (n=0;n<nn;n++)
             { n6=6*n;
               for (k=0;k<6;k++) printf("%11.4g",*(cnt+k+n6+m6)); PRN;
             }
           PRN;
         }

       free(cs); free(zi3);
       free(zi); free(cnt);
       free(bs); free(bsi);
       free(e); free(d);
       free(c); free(b);
       return (1);
     }
  }

int trns2p (long double *p, long double *a, long double *b, int lp, int lo)
 {
   /* f_1(t)*f_2(t)=> Y(p)=a_2/p+(a_0+a_1*p)/(a_3+a_4*p+a_5*p^2) .
      f_k(t)=b_(6k)+e^(-
b_{6k+4)*t)*[b_{6k+1}sin(b_{6k+5}*t)+b_{6k+2}cos(b_{6k+5}*t)]: b_{6k+3}=-1;
                or e^(-b_{6k+4)*t)*[b_{6k+1}sh(b_{6k+5}*t)+b_{6k+2}
ch(b_{6k+5}*t)] : b_{6k+3}=1.
      lp=1 -- TdT/dt; lo=1 -- print
    */
  int i, k, k2, k6, ki, nk, np;
  long double *c, *d, *f, *e, r0, r1, e1, e2, b1, b2, d0, d1, d2,
eps=0.00001;
```

```
  d=(long double *)calloc(10,sizeof(long double));
  c=(long double *)calloc(30,sizeof(long double));
  e=(long double *)calloc(10,sizeof(long double));
  f=(long double *)calloc(40,sizeof(long double));
  if ( f!=NULL && b!=NULL && c!=NULL && d!=NULL )
   {
     if ( lo==1 )
      { printf("\n trns2p: b(p)=\n");
        for (k=0;k<6;k++) printf("%10.4g",*(a+k));  PRN;
        for (k=0;k<6;k++) printf("%10.4g",*(b+k)); PRN; }
     for (k=0;k<30;k++) *(c+k)=0.;
     r0=*a*(*b);

     if ( fabs(*(a+5)-(*(b+5)))>eps   && fabs(*(a+4)-(*(b+4)))>eps ) //
f1(a)\ne f2(b)
      {
        if ( *(a+3)==*(b+3) )
         { ki=1;
           nk=4;
           for (k=0;k<2;k++)
             { np=0;
               k2=6*k;
               *(c+k+1)=*b*(*(a+k+1));
               *(c+k+4)=*(a+4+k);
               *(c+k2+3)=*(c+k2+15)=*(a+3);
               *(c+k+7)=*a*(*(b+1+k));
               *(c+k+10)=*(b+4+k);

               *(c+k2+13)=(*(a+1)*(*(b+2))+ki*(*(a+2))*(*(b+1)))/2.;

*(c+k2+14)=(*(a+2)*(*(b+2))+ki*(*(a+3))*(*(a+1))*(*(b+1)))/2.;
               *(c+k2+16)=*(a+4)+(*(b+4));
               *(c+k2+17)=*(a+5)+ki*(*(b+5));
               ki=-1;
             }
           if ( lo==1 )
            { k6=6*nk;
              printf("\n\t trns2p: nk=%u array c(%u)\n",nk,k6);
              for (i=0;i<k6;i++) printf("%10.4g%c",*(c+i), (i%6==5) ? '\n' :
' ');
            }
         }
        else
         { np=1;
           if (*(a+3) == -1 )
            { for (k=0;k<6;k++) *(d+k)=*(b+k);
              for (k=0;k<6;k++) *(a+k)=*(d+k);
              for (k=0;k<6;k++) *(b+k)=*(a+k); }
           e1=*b*(*(a+2)+(*(a+1)))/2.;
           e2=*b*(*(a+2)-(*(a+1)))/2.;
           b1=*(a+4)+(*(a+5));
           b2=*(a+4)-(*(a+5));
           *(f+12)=(e1*b2+e2*b1);
           *(f+13)=(e1+e2);
           *(f+15)=b1*b2;
           *(f+16)=b1+b2;
```

```
            *(f+17)=1.;
            nk=2;
            ki=1;
            for (k=0;k<2;k++)
              { k2=6*k;
                e1=(*(a+2)+ki*(*(a+1)))/2.;
                *(c+k2)=e1*(*(b+k));
                *(c+k2+1)=e1*(*(b+k+1));
                *(c+k2+2)=*(b+3);
                *(c+k2+3)=*(a+4)+(*(b+4))+ki*(*(a+5));
                *(c+k2+4)=*(b+5);
              }
            if ( lo==1 )
              { k6=6*nk;
                for (i=0;i<k6;i++) printf("%10.4g%c",*(c+i), (i%6==5) ? '\n'
: ' ');   }
           }
       }
      else
       { np=1;
         nk=2;
         *(c+3)=*(c+9)=*(a+3);
         for (k=0;k<2;k++)
           { k2=6*k;
             *(c+k2+4)=(k+1)*(*(a+4));
             *(c+k2+5)=(k+1)*(*(a+5));
             if (lp==1 ) *(c+k+1)=2*(*a)*(*(a+k+1)); }

         if ( lp==0 )
          { r0=*a*(*a);
            *(c+6)=(*(a+2)*(*(a+2))+(*(a+1))*(*(a+1)))/2.;
            *(c+7)=*(a+1)*(*(a+2));
            *(c+8)=(*(a+2)*(*(a+2))-(*(a+1))*(*(a+1)))/2.;
          }
         if ( lp==1 )
          { r0=0;
            b1=-(*(a+4)*(*(a+1))+(*(a+5))*(*(a+2)))/2.;
            b2=(*(a+5)*(*(a+1))-(*(a+4))*(*(a+2)))/2.;
           // printf("\n\t b1=%11.4g, b2=%11.4g\n",b1,b2);
            *(c)=*a*b1;
            *(c+1)=*a*b2;
            *(c+2)=0.;
            *(c+8)=*(a+2)*b2+(*(a+1))*b1;
            *(c+7)=*(a+2)*b2-(*(a+1))*b1;
            *(c+6)=*(a+1)*b2+(*(a+2))*b1;
          /*  k6=6*nk;
            printf("\n\t trns2p: T*dT/dt\n");
            for (i=0;i<k6;i++) printf("%10.4g%c",*(c+i), (i%6==5) ? '\n' : '
');
          */
          }
       }
             // Transorm to p
      for (k=0;k<nk;k++)
        {
          k6=6*k;
```

```
         *(f+k6)=*(c+k6)*(*(c+k6+5))+(*(c+k6+1))*(*(c+k6+4));
         *(f+k6+1)=*(c+k6+1);
         *(f+k6+2)=0;
         *(f+k6+3)=(*(c+k6+4))*(*(c+k6+4))-
(*(c+k6+3))*(*(c+k6+5))*(*(c+k6+5));
         *(f+k6+4)=2*(*(c+k6+4));
         *(f+k6+5)=1.;
        /* if ( fabs(*(c+k6+2)) >0.01 )
          { *d=*(c+k6+2);
            *(d+1)=*(c+k6+4);
            for (i=0;i<6;i++) *(e+i)=*(f+k6+i);
            yprv2 (c,d,e,2);
            for (i=0;i<6;i++) *(f+k6+i)=*(c+i);
          }
        */
       }
     *(f+2)=r0;
   /*  k6=6*nk;
     printf("\n\t lp=%u; r0=%12.5g; f(%u)\n",lp,r0,k6);
     for (i=0;i<k6;i++) printf("%9.4g%c",*(f+i), (i%6==5) ? '\n' : ' ');
    */
     pslfn (p,f,nk,lo);
     *(p+2)=r0;
     if ( lo==1 )
      { printf("\n output trns2p p=");
        for (i=0;i<6;i++) printf("%9.3g",*(p+i)); }
     free(f); free(e);
     free(c); free(d);
     return(1);
   }
  else
   { printf("\n no memory for c, d, e or f in  trns2p:\n");
     return(-1); }
 }

 long double intt2 (long double *c, long double t0, long double tt, int nk)
  {
    int j, k, k2, k6, ns, nc, ne;
    long double fint, et, r0, r1, r2, e0, om;
    fint=0.;
    for (k=0;k<nk;k++)
      { k6=6*k;
       // printf("\n\t intt2: c");
        //for (j=0;j<6;j++) printf("%12.5g",*(c+k6+j));
        fint+=*(c+k6)*(tt-t0);
        e0=-(*(c+k6+4));
        om=*(c+k6+5);
        et=(exp(e0*tt)-exp(e0*t0))/(e0*e0+om*om);
        r1=om*(*(c+k6+2))+e0*(*(c+k6+1));
        r2=e0*(*(c+k6+2))+(*(c+k6+3))*om*(*(c+k6+1));
        r0= ( om==0 ) ? 1 : cos(om*tt)-cos(om*t0);
        fint+=et*(r1*(sin(om*tt)-sin(om*t0))+r2*r0);
       //  printf("\n intt2:tt=%6.2f;  fint=%10.4g\n",tt,fint);
      }
    return(fint);
  }
```

```
 int atbt2 (long double *, long double *, long double *, int, int);

 long double intab (long double *a, long double *b, long double t0, long
double tt, int lo)
  {
   int i, j, k, m, nk;
   long double *c, tn;
   c=(long double *)calloc(30,sizeof(long double));
   if ( c!=NULL )
    {
      nk=atbt2 (c,a,b,0,lo);
      tn=intt2 (c,t0,tt,nk);
      if (lo==1 )  printf("\n\tintab: t0=%7.3f, tt=%7.3f,
tn=%10.4g\n",t0,tt,tn);
      free(c);
      return(tn);
    }
   else
    { printf("\n no memory for e in  intab\n");
      return(-1); }
 }


int atbt2p (long double *e, long double *a, long double *b, int lp, int lo)
 {
   // input A(p), B(p) -> a(t)*b(t); output V2(p).
   // lp=1 - dT/dt; lo=1 -- printf

   int k, j, k2, nk;
   long double *c, *d, *f, *g;
   d=(long double *)calloc(30,sizeof(long double));
   c=(long double *)calloc(30,sizeof(long double));
   f=(long double *)calloc(30,sizeof(long double));
   g=(long double *)calloc(30,sizeof(long double));
   if ( c!=NULL || d!=NULL || f!=NULL || g!=NULL )
    {
      if ( lo==1 )
       { printf("\n\t  atbt2p: a(p),b(p)\n");
         for (j=0;j<6;j++) printf("%11.4g",*(a+j)); PRN;
         for (j=0;j<6;j++) printf("%11.4g",*(b+j)); PRN; }
      orig1 (f,a,0);
      orig1 (g,b,0);
      nk=atbt2(c,f,g,lp,0);
      *(d+2)=*c;
      for (k=0;k<nk;k++)
        {
          k2=6*k;
          *(d+k2) =*(c+k2+1)*(*(c+k2+5))+(*(c+k2+2))*(*(c+k2+4));
          *(d+k2+1)=*(c+k2+2);
          *(d+k2+3)=*(c+k2+4)*(*(c+k2+4))-
(*(c+k2+3))*(*(c+k2+5))*(*(c+k2+5));
          *(d+k2+4)=*(c+k2+4)*2;
          *(d+k2+5)=1;
        }
     // printf("\n\t atbt2p: V(p)\n");
```

```
      pslfn (e,d,nk,1);
      if ( lo==1 )
       { printf("\n\t  atbt2p: e=");
         for (j=0;j<6;j++) printf("%11.4g",*(e+j)); PRN; }

      free(g); free(f);
      free(c); free(d);
      return(nk);
    }
   else
    { printf("\n no memory for e in  atbt2p\n");
      return(-1); }
 }

int atbt2 (long double *c, long double *a, long double *b, int lp, int lo)
 {
  /* f_1(t)*f_2(t)= r0+ sum(k=1,nk)[e^{-
al_k*t)(c_1f_1(om_k*t)+c_2f_2(om_k*t)].
     lp=1: T(t)*dT/dt */
   int i, k, k2, k6, ki, kz, nk;
   long double *d, *e, r0, e1, e2, r1, zi, eps=0.0001;
   d=(long double *)calloc(10,sizeof(long double));
   e=(long double *)calloc(10,sizeof(long double));
   if ( d!=NULL && e!=NULL )
    {
      if ( lo==1 )
       { printf("\n\t atbt2:  f(t)*g(t): a, b=\n");
         for (k=0;k<6;k++) printf("%10.4g",*(a+k));  PRN;
         for (k=0;k<6;k++) printf("%10.4g",*(b+k)); PRN; }
      for (k=0;k<30;k++) *(c+k)=0.;
      *c=*a*(*b);
      nk=4;
      for (k=0;k<6;k++) *(d+k)=*(a+k);
      for (k=0;k<6;k++) *(e+k)=*(b+k);

      if ( *(a+3)==*(b+3) )
       {
         if (lp==1 )  // T(t)*dT/dt
          { *b=0.;
            r0=-(*(b+4)*(*(b+1))+(*(b+5))*(*(b+2)));
            r1=*(b+5)*(*(b+1))-(*(b+4))*(*(b+2));
            *(b+1)=r0;
            *(b+2)=r1;
            nk=3;
          }
         if (*(b+5) > (*(a+5)) )
          {
            for (k=0;k<6;k++) *(d+k)=*(b+k);
            for (k=0;k<6;k++) *(e+k)=*(a+k); }
         ki=1;
         for (k=0;k<2;k++)
           {
             k2=6*k;
             *(c+k+19)=*e*(*(d+k+1));
             *(c+k+22)=*(d+4+k);
             *(c+k2+3)=*(c+k2+15)=*(d+3);
```

```
            *(c+k+13)=*d*(*(e+1+k));
            *(c+k+16)=*(e+4+k);
            *(c+k2+1)=(*(d+1)*(*(e+2))+ki*(*(d+2))*(*(e+1)))/2.;
            *(c+k2+2)=(*(d+2)*(*(e+2))+ki*(*(d+3))*(*(d+1))*(*(e+1)))/2.;
            *(c+k2+4)=*(d+4)+(*(e+4));
            *(c+k2+5)=*(d+5)+ki*(*(e+5));
            ki=-1;
          }
        if ( fabs(*d) < eps ) nk-=1;
        if ( fabs(*e) < eps ) nk-=1;
       }
      else
       {
        if (*(a+3) == -1 )
         { for (k=0;k<6;k++) *(d+k)=*(b+k);
           for (k=0;k<6;k++) *(e+k)=*(a+k); }
        if ( fabs(*e) > eps )
         { *(c+19)=*e*(*(d+1));
           *(c+20)=*e*(*(d+2));
           for (k=0;k<3;k++) *(c+k+21)=*(d+3+k);
           nk=4;
         }
        else nk=3;

        ki=1;
        for (k=0;k<3;k++)
          { k2=6*k;
            e1= (k==2) ? *d : (*(d+2)+ki*(*(d+1)))/2.;
            *(c+k2+1)=e1*(*(e+1));
            *(c+k2+2)=e1*(*(e+2));
            *(c+k2+3)=*(e+3);
            *(c+k2+4)= (k==2) ? *(e+4) : *(d+4)+(*(e+4))-ki*(*(d+5));
            *(c+k2+5)=*(e+5);
            ki=-1;
          }
       }
     /* if ( lo==1 )
       { k6=6*nk;
         printf("\n\t atbt2(%u): c(%u)\n",nk,k6);
         for (i=0;i<k6;i++) printf("%12.4g%c",*(c+i), (i%6==5) ? '\n' : '
');  PRN; }
     */
      free(e); free(d);
      return(nk);
    }
   else
    { printf("\n no memory for  d or e in atbt2s\n");
      return(-1); }

 }

void slfz (long double *z, long double *h, int n)
 {
   /* h=[h, L0, Lz, x0]; n - number of self-values.
      z[n] -- array of self-values; Lz=u_1; x0=h_2;
      [dZ/dz-h*Z]_(z=0)=0; [dZ/dz+h*Z]_(z=Lz)=0;
```

```
      f(x)=2*h*x*cos(x*Lz)+(h*h-x*x)*sin(x*Lz)=0.
      g(x)=f'(x)=(2*h+h^2-x^2)*cos(x*L)-2*x*(h*L+1)*sin(x*L). */
     /* output - slfz.dat: alz_n; int Z_n(z)dz; ||Z_n(h)||^2.    */

   int i, j, k, n1;
   long double h1, x, dx, x1, Lz, bc, bs, d1, d2;
   FILE *fcl;

   n1=3*n;
   x1=*(h+3);
   Lz=*(h+1)-(*(h));
   h1=*(h+2);
   printf("\n slfz: h="); for (i=0;i<4;i++) printf("%7.2f",*(h+i)); PRN;
   for (i=0; i<n; i++)
     { x=x1;
       dx=1.0;
       while ( fabs(dx) > .0001 )
        { bc=cos(x*Lz);
          bs=sin(x*Lz);
          d1=(x*x-h1*h1)*bs+2*h1*x*bc;
          d2=((x*x-h1*h1)*Lz+2*h1)*bc+2*x*(1-Lz*h1)*bs;
          dx=d1/d2;
          x-=dx;
        }
       *(z+i)=x;
       x1=pi/Lz+x;
     }
    for (k=0;k<n;k++) printf("%11.5g",*(z+k)); PRN;
 }

 void slbr2 (long double *sr, long double *g, int mm)
  {
    // find self-values for Zn(sl*r)=Y0(sl*r)*dj(j,sl*a)-dY(sl*a)*J0(sl*r).

    int k, n, m;
    long double a, b, h, f1, f2, gyb, gjb, gya, gja, hyb, hjb, hya, hja, x,
dx, eps=0.0005;
    a=*g;
    b=*(g+1);
    h=*(g+2);
    x=*(g+3);
    for (m=0;m<mm;m++)
      {
        dx=1.;
        while (dx > eps )
         {
           gyb=h*yn(0,b*x)-x*yn(1,b*x);
           gjb=h*jn(0,b*x)-x*jn(1,b*x);
           hyb=h*yn(1,b*x)-x*yn(0,b*x);
           hjb=h*jn(1,b*x)-x*jn(0,b*x);
           gya=h*yn(0,a*x)+x*yn(1,a*x);
           gja=h*jn(0,a*x)+x*jn(1,a*x);
           hya=x*yn(0,a*x)-h*yn(1,a*x);
           hja=x*jn(0,a*x)-h*jn(1,a*x);
           f1=gyb*gja-gjb*gya;
           f2=b*hyb*gja-a*gyb*hja-(b*hjb*gya-a*gjb*hya);
```

```
           printf("\n\t slbr2: dx(%u)=%10.3g; f1=%10.3g; f2=%10.3g:
x=%10.4g\n",m,dx,f1,f2,x);
           dx=f1/f2;
           x-=dx;
         }
        *(sr+m)=x;
        x+=pi; // /(b-a);
      }
    printf("\n\t p/p slbr2 - self_values for Zn=yn(0,x)-gn*jn(0,x) -
sl(%u)\n",mm);
    for (k=0;k<mm;k++) printf("%11.5g",*(sr+k)); PRN;
  }

 void slbr2 (long double *, long double *, int); // 1960
 void slfz (long double *, long double *, int);

 int bs2int (long double *sl, long double *g, int ps, int mm, int lp)
  {
    FILE *fcl;
    int i, k, n, m, m2, m3, k1, k2, mg;
    long double a, b, h, *c, *x, *sli, *bs1, *bs3, r1, r2;
    c=(long double *)calloc(20,sizeof(long double));
    x=(long double *)calloc(20,sizeof(long double));
    sli=(long double *)calloc(20,sizeof(long double));
    bs1=(long double *)calloc(200,sizeof(long double));
    bs3=(long double *)calloc(200,sizeof(long double));
    if ( c!=NULL && x!=NULL &&  sli!=NULL && bs1!=NULL && bs3!=NULL )
     {
       a=*g;
       b=*(g+1);
       h=*(g+2);
       printf("\n input sl-s: a=%7.2f; b=%7.2f",a,b);
       for (i=0;i<5;i++) printf("%7.3f",*(g+i)); PRN;

       if ( ps==0 )
        { *sl=4.96351;
          *(sl+1)=9.25262;
          *(sl+2)=13.6558;
          *(sl+3)=18.098;
          *(sl+4)=22.5577;
          *(sl+5)=27.0265;
          // slbr2 (sl, g, mm+1);
          printf("\n\t self_values for Rn=yn(0,x)-gn*jn(0,x) - a=%5.2f;
b=%5.2f; h=%5.2f; sl(%u)\n",a,b,h,mm);
        }
       if ( ps==1 )
        { slfz  (sl, g, mm+1);
          printf("\n\t self_values for Zk=cos(sl*z)-h/slsin(sl*z) - a=%5.2f;
b=%5.2f; h=%5.2f; sl(%u)\n",a,b,h,mm);
        }


       // k1= ( ps==0 ) ? 1 : 0;
       for (i=0;i<mm;i++) *(bs1+i)=*(sl+i);
       for (i=0;i<mm;i++) printf("%11.5g",*(bs1+i)); PRN;
       m2=2*mm-1;
```

```
       mg=plgv (c, x, mm+3, 0);
       //printf("\n weights - Leg. c\n");
       //for (i=0;i<mg;i++) printf("%9.3g",*(c+i)); PRN;
       //printf("\n knots - Leg. x\n");
       //for (i=0;i<mg;i++) printf("%10.4g",*(x+i)); PRN;
       m3=3*mm;
       m2=mm*mm;
       for (n=0;n<mm;n++)
         {
           *sli=*(bs1+n);
           *(sli+1)=*(bs1+n);
           *(bs1+n+mm+mm)=r1=fintgs(c,x,g,sli,ps,mg,2,0);
           *(bs1+n+mm)=fintgs(c,x,g,sli,ps, mg, 1, 0)/r1;
           for (m=0;m<mm;m++)
             { k1=mm*n+m;
               *(sli+1)=*(bs1+m);
               *(bs1+m3+k1)=fintgs(c,x,g,sli,ps, mg,2, 0)/r1;
               *(bs1+m2+m3+k1)=fintgs(c,x,g,sli,ps, mg,2, 1)/r1;
               *(sli+2)=*(bs1+m);
               *(bs3+k1)=fintgs(c,x,g,sli,ps, mg,3, 0)/r1;
               *(bs3+k1+m2)=fintgs(c,x,g,sli,ps, mg,3, 1)/r1;
             }
         }

       if ( ps==0 ) printf("\n\t int. trans for Rn(n=%ur; ra_p=%5.2f,
rb_p=%5.2f; lp=%u\n",mm,a,b,lp);
       if ( ps==1 ) printf("\n\t integr. transforms for Zk(n=%ur; h=%5.2f
rb_p=%5.2f\n",mm,h,b);
       if ( ps==0 )printf("\n int BesR(s_1)=");
       if ( ps==1 )printf("\n int Zk=");
       for (i=0;i<mm;i++) printf("%10.4g",*(bs1+i+mm)); PRN;
       if ( ps==0 )printf("\n Norm^2 Bes2=");
       if ( ps==1 )printf("\n Norm^2 Zk(z)=");
       for (i=0;i<mm;i++) printf("%10.4g",*(bs1+i+mm+mm)); PRN;
       if ( ps==0 )
          printf("\n\t int R_n*R_n1(r); lp=%u\n",lp);
       if ( ps==1 )
          printf("\n\t int Z_k*Z_k1(z)\n");
       for (i=0;i<m2;i++) printf("%10.4g%c",*(bs1+m3+i), (i%mm==mm-1) ? '\n'
: ' ');
       if ( ps==0 )
          printf("\n\t int R_n*dR_n1(r)/dr; lp=%u\n",lp);
       if ( ps==1 )
          printf("\n\t int Z_k*dZ_k1(z)/dz\n");
       for (i=0;i<m2;i++) printf("%10.4g%c",*(bs1+m2+m3+i), (i%mm==mm-1) ?
'\n' : ' ');
       if ( ps==0 )printf("\n\t int Rn(r)*R_n1(r)*R_n2(r); lp=%u\n",lp);
       if ( ps==1 )printf("\n\t int Zk(z)*Z_k1(z)*Z_k2(z)\n");
       for (i=0;i<m2;i++) printf("%10.4g%c",*(bs3+i), (i%mm==mm-1) ? '\n' :
' ');
       if ( ps==0 )printf("\n\t int Rn(r)*R_n1(r)*dR_n2(r)/dr; lp=%u\n",lp);
       if ( ps==1 )printf("\n\t int Zk(z)*Z_k1(z)*dZ_k2(z)/dz\n");
       for (i=m2;i<2*m2;i++) printf("%10.4g%c",*(bs3+i), (i%mm==mm-1) ? '\n'
: ' ');
      /*
       if ( ps==0 )
```

```
        { if ( lp==0 ) fcl = fopen("bsk1.dat","w");
          if ( lp==1 ) fcl = fopen("bsp1.dat","w");
        }
       else fcl = fopen("zint1.dat","w");
       k2=m3+2*mm*mm;
       for (i=0; i<k2; i++) fprintf(fcl,"%g,",*(bs1+i));
       fclose(fcl);

       if ( ps==0 )
        { if ( lp==1 ) fcl = fopen("bsnp3.dat","w");
          if ( lp==0 ) fcl = fopen("bsnk3.dat","w");
             for (i=0; i<2*m2; i++) fprintf(fcl,"%g,",*(bs3+i));
          fclose(fcl);
        }
       else
        { fcl = fopen("zint3.dat","w");
          for (i=0; i<2*m2; i++) fprintf(fcl,"%g,",*(bs3+i));
          fclose(fcl); }
      */
       free(bs3);
       free(bs1); free(sli);
       free(x); free(c);
       return(1);
     }
    else
     { printf("\n no memory for  f, c or e in fintgs\n");
       return(-1); }
  }

 long double fxsp (long double *, long double, long double, long double,
int, int);

 long double fintgs (long double *c, long double *x, long double *g, long
double *s, int ps, int m, int n, int lp)
  {
   /* program of obtaining fint = integral f(x) on x in [a,b];
      GAUSS; I=sum[k=0,m] a(k)*f(c(k));
      Legandre, xl[m] -- weight values; m-- order of apr. formula;
      f(x) -- function fxs(x);                                          */

    int i, k, m2;
    long double a, b, h, *f, r1, r2;
    a=*g;
    b=*(g+1);
    h=*(g+2);
    m2=2*m;
    f=(long double *)calloc(m2+2,sizeof(long double));
    if ( f!=NULL )
     {
       for (k=0;k<m;k++)
       { r1=(b-a)/2.*(*(x+k))+(b+a)/2.;
         if (lp==0 )
          { *(f+k)=fxs(s,a,h,r1,ps,n);
         //  printf("\n\t fintgs: f1(%u,%u:
%8.2f;%6.3f)=%12.4g\n",k,n,r1,*s,*(f+k));
             r1=-(b-a)/2.*(*(x+k))+(b+a)/2.;
```

```
           *(f+k)+=fxs(s,a,h,r1,ps,n);
         //  printf("\n\t fintg1: f1(%u,%u:
%8.2f;%6.3f)=%12.4g\n",k,n,r1,*s,*(f+k));
          }
         if (lp==1 )
          { *(f+k)=fxsp(s,a,h,r1,ps,n);
              r1=-(b-a)/2.*(*(x+k))+(b+a)/2.;
           *(f+k)+=fxsp(s,a,h,r1,ps,n);
          }
           //   printf("\n\t fintgs:c(%u)=%10.4g;
f(%u,%u)=%10.4g\n",k,*(c+k),m,k,*(f+k));
       }
    // for (i=0;i<m;i++) printf("%10.4g",*(c+i)); PRN;
    // for (i=0;i<m;i++) printf("%10.4g",*(f+i)); PRN;
       r1=0.;
       for (i=0;i<m;i++) r1+=*(c+i)*(*(f+i));
       r1*=(b-a)/2.;
      // printf("\n\t fintgs: f(a=%7.2f,b=%7.2f)=%10.4g\n",a,b,r1);
       free(f);
       return(r1);
     }
    else
     { printf("\n no memory for  f, c or e in fintgs\n");
       return(-1); }
  }


int plg (long double *c, int mm, int lp);
int gorn (long double *d, long double *b, long double *t, int n);

int plgv (long double *a, long double *xl, int mm, int lp)
 {
/*+-----------------------------------------------------------------+
  |    □®«¨-®¬ ‹¥¦ -¤à  -  lp=0;                                    |
  |    □®«¨-®¬ —¥¡ëè¥¢  -  lp=1;                                    |
  |    □®«¨-®¬ □à¬¨â    -  lp=2;                                    |
  |     Tm(x)=sum[i=0,[mm+1]/2:[c[i]*x^(2i+ip)];                    |
  |    xl[i] - ç¨á«  Šà¨áâ®ä¥«ï - ¤«ï ¨-â¥£à¨à®¢ -¨ï.               |
  |    a[i] - ã§«ë  ¯¯à®ªá¨¬¬àãîé¥£® ¯®«¨-®¬                        |
  +-----------------------------------------------------------------+ */
   int i, i1, k, m, k2, kk, ip;
   long double *d, *e, *c, *f, r1, r2;
   m=(mm+1)/2;
   d=(long double *)calloc(20,sizeof(long double));
   e=(long double *)calloc(20,sizeof(long double));
   c=(long double *)calloc(20,sizeof(long double));
   f=(long double *)calloc(20,sizeof(long double));
   if ( d!=NULL && e!=NULL && c!=NULL && f!=NULL )
    {
    /*  plg(c,mm,lp);
      for (i=0;i<=m;i++) *(d+i)=(*(c+m-i));
      gorn(c,e,f,m);
      ip=0;
      for (i=0;i<m;i++)
       { k2=2*i+ip;
         i1=i+ip;
```

```
         *(a+i1)=sqrt(*(e+k2)); }
       printf("\n\t ã§«ë - ¬ áá¨¢  [%u]\n",m);
       for (i=0;i<m;i++) printf("%9.4g",*(a+i)); PRN;
   */
       if (m==3)
        { *(xl)=0.4679;
          *(xl+1)=0.3608;
          *(xl+2)=0.1713;
          *a=0.2386;
          *(a+1)=0.6612;
          *(a+2)=0.9325;
        }
       if ( m==4 )
        {
          *xl=0.3627;
          *(xl+1)=0.3137;
          *(xl+2)=0.2224;
          *(xl+3)=0.1012;
          *a=0.1834;
          *(a+1)=0.5255;
          *(a+2)=0.7967;
          *(a+3)=0.9603;
        }
       if ( m==5 )
        {
          *xl=0.148874;
          *(xl+1)=0.433395;
          *(xl+2)=0.67941;
          *(xl+3)=0.86506;
          *(xl+4)=0.97391;
          *a=0.29552;
          *(a+1)=0.26927;
          *(a+2)=0.21909;
          *(a+3)=0.14945;
          *(a+4)=0.06667;
        }
     /*
       for (k=0; k<m; k++)
          { k2=k+k;
          r2=*(a+k)*(*(a+k));
            for (i=m,r1=0.;i>0;i--) r1=r2*r1+i*(*(c+i));
          *(xl+k)=.5/((1-r2)*r2*r1*r1);
          }

       printf("\n\t plgv: ¢¥á®¢ë¥ ª®íä--âë: ¬ áá¨¢ xl[%u]\n",m);
       for (i=0;i<m;i++) printf("%9.4g",*(xl+i)); PRN;
      */
       free(f); free(c);
       free(e); free(d);
       return(m);
     }
    else
     { printf("\n no memory for c, d, e or f in plgv\n"); return(-1); }
  }

int plg (long double *c, int mm, int lp)
```

```
 {
/*+-----------------------------------------------------------------+
  |    □®«¨-®¬ ‹¥¦ -¤à  -  lp=0;                                    |
  |    □®«¨-®¬ —¥¡ëè¥¢  -  lp=1;                                    |
  |    □®«¨-®¬ □à¬¨â    -  lp=2;                                    |
  |     Tm(x)=sum[i=0,[mm+1]/2:[c[i]*x^(2i+ip)];                    |
  +-----------------------------------------------------------------+ */
  int i, k, m, n, ip;
  long double *b, r1, r2;
  printf("\n\tmm=%u\n",mm);
  m=(mm+1)/2;
  b=(long double *)calloc(40,sizeof(long double));
  if ( b!=NULL )
   {
     for (i=0;i<30;i++) *(b+i)=0.;
     *b=*(b+10)=1.;
     if ( lp==2 ) *(b+10)=2.;
     for (k=1; k<=mm; k++)
       { r1=(lp==0) ? (2.*k+1)/(k+1.) : 2.;
          if ( lp==0 ) r2=k/(k+1.);
          if ( lp==1 ) r2=1.;
          if ( lp==2 ) r2=2*k;
          n=(k+1)/2;
          ip= ( (k+1)%2 == 0 ) ? 0 : 1;
          if ( (k+1)%2 == 0 ) *(b+20)=-r2*(*b);
          for (i=0;i<=n;i++)
               *(b+i+20+1-ip)=r1*(*(b+i+10))-r2*(*(b+i+1-ip));
          for (i=0;i<=n;i++)
             { *(b+i)=*(b+i+10);
               *(b+i+10)=*(b+i+20); }
       }
     for (i=0;i<=m;i++) *(c+i)=*(b+10+i);
     for (i=0;i<=m;i++) *(c+i+m+1)=*(b+i);
   /*  if ( lp==0 ) printf("\n\t □®«¨-®¬ ‹¥¦ -¤à  -¬ áá¨¢ b[%u]\n",m+1);
     if ( lp==1 ) printf("\n\t □®«¨-®¬ —¥¡ëè¥¢  -¬ áá¨¢ b[%u]\n",m+1);
     if ( lp==2 ) printf("\n\t □®«¨-®¬ □à¬¨â  -¬ áá¨¢ b[%u]\n",m+1);
     for (i=0;i<=m;i++) printf("%9.4g",*(c+i)); PRN;     */
     if ( mm==6 )
      { *(c)=231;
        *(c+1)=-315;
        *(c+2)=105;
        *(c+3)=-5; }
     if ( mm==8 )
      { *(c)=6436;
        *(c+1)=-12012;
        *(c+2)=5930;
        *(c+3)=-1260;
        *(c+4)=35; }
     printf("\n\t coef. of pol-Legandre(%u)\n",mm);
     for (k=0;k<=m;k++) printf("%11.5g",*(c+k));
     free(b);
     return(n);
   }
  else
   { printf("\n no memory for  b in plg\n");
     return(-1); }
```

```
 }


int svpi(long double *a,long double *b, int n, int m);

long double nutp_rf (long double *a, long double eps, int n);

int gorn (long double *d, long double *b, long double *t, int n)
 {
/*+-----------------------------------------------------------------+
  |    ,ëç¨á«¥-¨¥ ª®à-¥©  «£. ãà ¢-¥-¨ï á ¤¥©áâ¢. ª®íä.-¬¨        |
  |  P(x)=Sum[i=0,n:a[i]*x^i]=sum[i=0,[(n+1)/2]:b[2i]+j*b[2i+1]]; |
  |'à¥¡ã¥¬ëé äã-ªæ¨¨: fntp(a,x,n); dfntp(a,x,n); nutp_rf(a,eps,n).|
  |   P(x)=sum{i=0,[(n+1)/2]:t[3i+2]*x^2+t[3i+1]*x+[3i]}.         |
  +-----------------------------------------------------------------+ */
  register i, k;
  int nk, k2, n2, ip, kk, nt;
  long double p, dp, q, dq, r1, r2, *c, *a, *e, e0=.1E-4;
  e=(long double *)calloc(n+1,sizeof(long double));
  c=(long double *)calloc(n+1,sizeof(long double));
  a=(long double *)calloc(n+1,sizeof(long double));
  if ( c!=NULL && a!=NULL && e!=NULL)
   {
    for (i=0;i<=n;i++) *(a+i)=*(d+i)/(*d);
    printf("\n\tŠ®à-¨ ¯®«¨-®¬  áâ¥¯¥-¨ %u\n",n);
    for (i=0;i<=n;i++) printf("%8.3g",*(a+i)); PRN;
    n2=n/2;
    nt=n;
    if ( n%2!=0 )
     { r2=nutp_rf(a,.0001,n);
       *b=*a;
       for (i=1;i<n;i++) *(b+i)=*(a+i)+r2*(*(b+i-1));
       for (i=0;i<n;i++) *(a+i)=*(b+i);
       nt-=1;
       k2=3*n2;
       *(t+k2)=-r2;
       *(b+n+n-2)=r2;
       *(t+k2+1)=1.;
       *(t+k2+2)=*(b+n+n-1)=0.;
     }
    if ( nt > 2 )
     { ip=1;
       *b=*c=1.;
       for (k=0; k<n2-1; k++)
       { nk=nt-2*k;
         if ( fabs(*(a+nk-2)) > e0 )
          { p=dp=*(a+nk-1)/(*(a+nk-2));
            q=dq=*(a+nk)/(*(a+nk-2)); }
          else
         { p=dp=*(a+1)/(*a);
           q=dq=*(a+2)/(*a); }
          kk=0;
          for (i=0;i<=nk;i++) *(e+i)=*(a+i);
          while ( (fabs(dp) > e0) || (fabs(dq) > e0) )
         {
           if ( kk < 50 )
```

```
           { *(b+1)=*(a+1)-p;
           *(c+1)=*(b+1)-p;
           for (i=2;i<=nk;i++)
             { *(b+i)=*(a+i)-p*(*(b+i-1))-q*(*(b+i-2));
               r1= (i<nk-1) ? *(b+i) : 0.;
           *(c+i)=r1-p*(*(c+i-1))-q*(*(c+i-2)); }
           r1=*(c+nk-2)*(*(c+nk-2))-(*(c+nk-1))*(*(c+nk-3));
           dp=(*(b+nk-1)*(*(c+nk-2))-(*(b+nk))*(*(c+nk-3)))/r1;
           dq=(*(b+nk)*(*(c+nk-2))-(*(b+nk-1))*(*(c+nk-1)))/r1;
           p+=dp;
           q+=dq;
           kk+=1;
           }
          else
           { for (i=0;i<=nk;i++) *(a+i)=*(e+nk-i)/(*(e+nk));
           kk=0;
           ip*=-1; }
        }
       printf("\n\t p(%u)=%10.4g, q=%10.4g\n",k,p,q);
         k2=3*k;
         *(t+k2)=(ip==1) ? q : 1./q;
         *(t+k2+1)=(ip==1) ? p : p/q;
         *(t+k2+2)=1.;
         for (i=0;i<nk;i++) *(a+i)=*(b+i);
         for (i=1;i<=nt;i++) *(b+i)=0.;
       }
   }
  k2=3*n2-1;
  for (i=0;i<3;i++) *(t+k2-i)=*(a+i);
  printf("\n\tŠ®íää¨æ¨¥-âë à §«®¦¥-¨ï ¯®«¨-®¬ -¬ áá¨¢ T\n");
  k2=3*((n+1)/2);
  for (i=0;i<k2;i++)
    printf("%9.5g%c",*(t+i), (i%6==5 || i==k2-1) ? '\n' : ' ');
  for (k=0; k<n2; k++)
    { nk=3*k;
    k2=4*k;
    p=-(*(t+nk+1))/2.;
    q=p*p-(*(t+nk));
    r1=( q>=0 ) ? 0. : 1.;
    dp=sqrt(fabs(q));
    *(b+k2)=p-dp*(1-r1);
    *(b+k2+1)=dp*r1;
    *(b+k2+2)=p+dp*(1-r1);
    *(b+k2+3)=-dp*r1;
    printf("\n\tb[%u]=%9.4g+j*(%9.4g)",k2,*(b+k2),*(b+k2+1));
    printf("\n\tb[%u]=%9.4g-j*(%9.4g)",k2+2,*(b+k2+2),*(b+k2+3));
    }
  if ( n%2!=0 ) printf("\n\tb[%u]=%9.4g",n+n-1,r2);
/*  printf("\n\tChecking of decision of equation\n");
  svpi(a, t, 3, n2+1);
  for (i=0;i<=n;i++) printf("%8.3g",*(a+i));  */
  free(a); free(c);
  free(e);
  return(ip);
 }
else
```

```
   { printf("\n no memory for  e or f in gorn\n");
     return(-1); }
 }

/*--------------------------- gorn.c ---------------------------*/

long double detm (long double *, int, int, int);

long double xals (long double *x, long double *a, int nn, int mr)
 {
   /* mr=0 or mr=2 - system of linear equation; mr=1 or mr=3 - Inverse
matrix;
      mr==2 or mr==3 Printing of results                          */
   register int m, l, k, i;
   int nn1, ii, i1, i2, mn, ln, nr;
   long double *b, *c, r1, r2, det, e0=1E-6;
   nn1=nn*(nn+1);
   if ( mr==2 )
    { printf("\n\tŒ âà¨æ  A[%u;%u] \n",nn,nn+1);
      for (i=0;i<nn1;i++)
        printf("%9.3g%c",*(a+i), (i%(nn+1)==nn) ? '\n' : ' ');
    }
   b=(long double *)(calloc)(nn1,sizeof(long double));
   if ( b!=NULL )
    { det=detm(a,nn,1,0);
      if ( fabs(det) < e0 )
       { printf("\n\t¨áâ¥¬  «¨-¥©-ëå ãà ¢-¥-¨© - ®á®¡ ï\n");
       free(b);
       return(det); }
      else
       {
       for (i=0;i<nn1;i++) *(b+i)=*(a+i);
       for (m=1; m<=nn; m++) *(b+m)=*(a+m)/(*a);
       for (m=1; m<nn; m++)
         { mn=m*(nn+1);
           for (l=m; l<nn; l++)
             { r2=r1=0.;
             for (k=0; k<m; k++)
               { i1=l*(nn+1)+k;
                 i2=k*(nn+1)+m;
                 r1+=*(b+i1)*(*(b+i2));
                 i1=mn+k;
                 i2=k*(nn+1)+l+1;
                 r2+=*(b+i1)*(*(b+i2));
               }
             i1=l*(nn+1)+m;
             *(b+i1)=*(a+i1)-r1;
             i1=mn+l+1;
             i2=mn+m;
             *(b+i1)=(*(a+i1)-r2)/(*(b+i2));
              }
         }
       if ( mr==2 )
        { printf("\n\t Œ âà¨æ  C[%u,%u]\n",nn,nn) ;
            for (k=0;k<nn;k++)
            { i1=k*(nn+1);
```

```
        for (i=0;i<=k;i++) printf("%8.3g",*(b+i+i1)); PRN; }
    printf("\n\t Œ âà¨æ  B[%u,%u]\n",nn,nn);
    for (k=0;k<nn;k++)
      { i1=k*(nn+1);
      for (i=0;i<=nn;i++)
          { if ( i>=k ) r1=(i==k) ? 1 : *(b+i+i1);
          else r1=0.;
          printf("%8.3g",r1); }
        PRN;
      }
  }
 if ( mr==0 || mr==2 )
  { *(x+nn-1)=*(b+nn1-1);
    for (m=1; m<nn; m++)
      { r1=0.;
      mn=(m+1)*(nn+1);
      ii=nn-m;
      for (l=ii; l<nn; l++)
        { i2=nn1-mn+l;
          r1+=*(b+i2)*(*(x+l)); }
      i1=nn1-mn+nn;
      *(x+nn-m-1)=*(b+i1)-r1;
      }
    printf("\n\t¥è¥-¨¥ á¨áâ¥¬ë %u «¨-¥©-ëå ãà ¢-¥-¨©\n",nn);
    printf("\t ¬¥â®¤®¬ ƒ ãáá  ¯® áå¥¬¥ • «¥æª®£®\n");
    for (i=0;i<nn;i++) printf("%8.3g",*(x+i)); PRN;
  }
 if ( mr==1 || mr==3 )
  { i1=(nn-1)*(nn+1);
    i2=(nn-1)*(nn+2);
    *(x+i1)=1/(*(b+i2));
    for (m=1; m<nn; m++)
      { mn=(nn-m)*nn;
      for (l=m+1; l<=nn; l++)
        { ii=nn-l+1;
          ln=(nn-l)*nn;
          r1=r2=0.;
          for (k=ii; k<nn; k++)
            { i1=mn+k;
            i2=(nn+1)*k+nn-l;
            r1+=*(x+i1)*(*(b+i2));
            i1=ln+nn-l+k;
            i2=nn*k+nn-m;
            r2+=*(b+i1)*(*(x+i2)); }
          i1=mn+nn-l;
          i2=ln+2*(nn-l);
          *(x+i1)=-r1/(*(b+i2));
          i1=ln+nn-m;
          *(x+i1)=-r2;
        }
      r1=0.;
      for (l=nn-m;l<nn;l++)
        { i1=mn-nn+l;
          i2=(nn+1)*l+nn-m-1;
          r1+=*(x+i1)*(*(b+i2)); }
      i1=mn-m-1;
```

```
            i2=i1+nn-m-1;
            *(x+i1)=(1-r1)/(*(b+i2));
            }
          ii=0;
          for (m=0; m<nn; m++)
            { for (l=0; l<nn; l++)
              { r1=0.;
                for (k=0;k<nn;k++)
                  { i1=m*nn+k;
                   i2=k*(nn+1)+l;
                   r1+=*(x+i1)*(*(a+i2)); }
                *(b+ii)=r1;
                ii+=1;
              }
            }
          if ( mr==3 )
           { printf("\n\t A[%u,%u]*AX[%u,%u]\n",nn,nn,nn,nn);
             for (i=0;i<nn*nn;i++)
               printf("%8.3f%c",*(b+i), (i%nn==nn-1) ? '\n' : ' ');
             printf("\n\tŽ¡à â- ï ¬ âà¨æ  AX[%u,%u]\n",nn,nn);
             for (i=0;i<nn*nn;i++)
               printf("%8.3g%c",*(x+i), (i%nn==nn-1) ? '\n' : ' ');
           }
         }
        free(b);
        return(det);
        }
     }
    else
     { printf("\n\t no memory for b in xals\n");
       return(0.); }
 }
/*ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ xals.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ*/

long double sortm (long double *a, long double *b, int n, int ln);

long double detm (long double *a, int n, int ln, int mr)
 {
/*ÉÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ»
  |  ,ëç¨á«¥-¨¥ ®¯à¥¤¥«¨â¥«ï ¬ âà¨æë € à §¬¥à-®áâ¨ n*n;       |
  |   ln=0 - ¬ âà¨æ ; ln=1 - á¨áâ¥¬  «¨-¥©-ëå ãà ¢-¥-¨©       |
  ÈÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¼ */

   register int m, k, i, nn1, nn, k1, k2, n1, nm;
   long double *b, *c, r, rd, r1, r2;
   nn1=n*(n+ln);
   nn=n*n;
   b=(long double *)(calloc)(nn1,sizeof(long double));
   c=(long double *)(calloc)(nn1,sizeof(long double));
   if ( b!=NULL && c!=NULL )
    {
      if ( mr==2 )
       { printf("\n\tŒ âà¨æ  A[%u;%u] \n",n,n);
       for (k=0;k<n;k++)
         { k1=k*(n+ln);
           for (i=0;i<n;i++) printf("%7.3g",*(a+i+k1));
```

```
           PRN; }
        }
       for (i=0; i<nn1; i++) *(b+i)=0;
     //  rd=sortm (a,c,n,ln);
       r=1.;
       for ( m=1; m<n; m++)
       { n1=n-m+ln;
         if ( m < n-1 ) for ( k=0; k<n-m-1; k++) r*=*c;
         for ( k=0; k<nn1; k++) *(b+k)=0.;
         for ( k=0; k<n-m; k++)
           { k2=k*n1;
             k1=k*(n1+1)+n1+1;
             for (i=0;i<n-m;i++)
               *(b+k2+i)=(*c*(*(c+k1+i+1))-(*(c+k1))*(*(c+i+1)));
           }
         nm=(n-m)*n1;
         for ( k=0; k<nm; k++)  *(c+k)=*(b+k);
       }
      r1=rd*(*c)/r;
      printf("\n\t |A[%u;%u]|=%7.3g\n",n,n,r1);
      free(c); free(b);
      return(r1);
    }
  else
    { printf("\n no memory for b or c in detm\n");
      return(0); }
 }

/*ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ detm.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ*/

long double sortm (long double *a, long double *b, int n, int ln)
 {
   register int m, k, i;
   int nn1, k1, k2, n1, lp;
   long double r, rd, r1, r2, e0=.0001;
   nn1=n*(n+ln);
   for (i=0; i<nn1; i++) *(b+i)=0;
   rd=fabs(*a);
   lp=0;
   n1=n+ln;
   if ( rd < e0 )
    { for (k=1;k<n;k++)
      { k1=k*n1;
        if (fabs(*(a+k1)) > rd )
         { rd=fabs(*(a+k1));
           lp=k; }
      }
      for (k=0; k<n; k++)
      { if ( k < lp ) k2=n1*(k+1);
        if ( k == lp ) k2=0;
        if ( k > lp ) k2=n1*k;
        k1= ( k==lp ) ? lp*n1 : k*n1;
        for (i=0;i<n1;i++) *(b+k2+i)=*(a+k1+i);
      }
      return( ( 2*(lp/2)==lp ) ? 1. : -1. );
    /*  printf("\n\t p/p SORTM - Œ âà¨æ  B[%u;%u] \n",n,n+ln);
```

```
      for (i=0;i<nn1;i++)
        printf("%7.3g%c",*(b+i), (i%(n+1)==n) ? '\n' : ' ');  */
     }
    else
     { for (i=0;i<nn1;i++) *(b+i)=*(a+i);
       return(1); }
 }

/*ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ detm.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ*/

long double fntp (long double *c, long double *x, int n)
 {
   register int i;
   long double r1;
   r1=*c;
   for (i=0;i<n;i++) r1=(*x)*r1+(*(c+i+1));
   return(r1);
 }

long double dfntp (long double *c, long double *x, int n)
 {
   register int i;
   long double r1;
   r1=*c*n;
   for (i=0;i<n-1;i++)
       r1=(*x)*r1+(*(c+i+1))*(n-i-1);
   return(r1);
 }

long double fntp (long double *c, long double *x, int n);
long double dfntp (long double *c, long double *x, int n);

long double nutp_rf (long double *a, long double eps, int n)
 {
/*+-----------------------------------------------------------------+
  |  Œ¥â®¤ □ìîâ®- -□ äá®-  à¥è¥-¨ï  «£¥¡à ¨ç¥á ª®£® ãà ¢-¥-¨ï         |
  |           x[k+1]=x[k]+al*p[k];  m - à Š¬¥à-®áâì;                 |
  |             p[k]=-{f'(x[k])}[-1]*f(x[k]).                        |
  |       'à¥¡ã¥¬ë äã-ªæ¨¨: fntp(a,x,n); dfntp(a,x,n).              |
  +-----------------------------------------------------------------+ */
  long double c, e, f, al, x;
  /* printf("\n\tŒ¥â®¤ □ìîâ®- -□ äá®-  à¥è¥-¨ï  «£¥¡à ¨ç¥cª®£®
ãà ¢-¥-¨ï\n"); */
  c=al=1.;
  x=.5;
  e=fntp(a,&x,n);
  while ( fabs(al*c) > eps )
   { c=e/dfntp(a,&x,n);
     x=x-al*c;
     f=fntp(a,&x,n);
     if ( (f-e) > eps*al*c ) al/=2.;
     e=f;
   }
  printf("\n\tx=%8.3g\n",x);
  return(x);
 }
```

```
int sppi (long double *,long double *,long double *, int, int);

int svpi (long double *a,long double *b, int n, int m)
  {
    register int i, k, k1, mj;
    long double *c, *d;
    d=(long double *)calloc(20,sizeof(long double));
    c=(long double *)calloc(20,sizeof(long double));
    if ( c!=NULL && d!=NULL )
     {
       for (i=0;i<n;i++) *(c+i)=*(a+i)=*(b+i);
       mj=n;
       if ( m > 1 )
        {
          for (k=1;k<m;k++)
             { k1=k*n;
               for (i=0;i<n;i++) *(d+i)=*(b+i+k1);
               mj=sppi(a,c,d,mj,n);
               for (i=0;i<mj;i++) *(c+i)=*(a+i);
             }
       }
     free(c); free(d);
     return(mj);
   }
 else
   { printf("\n\tno memory for c, d in svpi\n");
     return(-1); }
  }

int sppi (long double *a, long double *b, long double *c, int m, int l)
 {
   register int i, k, n1, ml;
   ml=m+l-1;
   if ( l > m )
    { n1=l;
      m=l;
      l=n1;
      for (i=0;i<l;i++) *(a+i)=*(b+i);
      for (i=0;i<m;i++) *(b+i)=*(c+i);
      for (i=0;i<l;i++) *(c+i)=*(a+i);
    }
   for (i=0;i<ml;i++) *(a+i)=0;
   for (k=0; k<l; k++)
     { for (i=0;i<=k;i++)
         (*(a+k))+=*(b+i)*(*(c+k-i)); }
   n1=m-l;
   if ( n1 > 0 )
    { for (k=0; k<n1; k++)
      { for (i=0;i<l;i++)
           (*(a+k+l))+=*(b+i+k+1)*(*(c+l-i-1)); }
    }
   for (k=1; k<l; k++)
     { for (i=0;i<k;i++)
         (*(a+ml-k))+=*(b+m-1-i)*(*(c+l-k+i)); }
   return(ml);
```

```
 }


int yprv2 (long double *e, long double *cv, long double *a, int lo)
 {
   register int i, k;
   long double bt, c0, r0, r1, r2, *d, *c, *f;
   c=(long double *)calloc(10,sizeof(long double));
   d=(long double *)calloc(10,sizeof(long double));
   f=(long double *)calloc(10,sizeof(long double));
   if ( d!=NULL && c!=NULL && f!=NULL )
    { /* cv=[cv_0=A; cv_1=al];
         cv_0/(p+cv_1)*(a_2/p+(a_0+a_1*p)/(a_3+a_4*p+a_5*p^2); lo=0; lo=1 -
print
         cv_0/(p+cv_1)+(a_2/p+(a_0+a_1*p)/(a_3+a_4*p+a_5*p^2); lo=2; lo=3 -
print

        Exit: V_2(p)=(e_0+e_1p)/(e_3+e_4*p+e_5*p^2)+e_2/p.
      */
     if ( lo==1 || lo==3 )
      { if ( lo==1 )
printf("\n\typrv2:%8.3g/(p+%8.3g)*V2(p)=\n",*(cv),*(cv+1));
        if ( lo==3 )
printf("\n\typrv2:%8.3g/(p+%8.3g)+V2(p)=\n",*(cv),*(cv+1));
        for (i=0;i<6;i++) printf("%9.3g",*(a+i));
      }
     for (i=0;i<6;i++) *(f+i)=*(a+i);
     bt=1/(*(cv+1));
     r0=*(cv)*bt;
     c0= ( lo<2 ) ? r0*(*(f+2)) : *(f+2);
     r1=r0*(*(f+2))*bt;
     r2= (lo > 1 ) ? *(f+2) : r1;
     *(f+2)=0.;
     *c=*(f+3);
     *(a+6)=0.;
     *(d+3)=0.;
   //  printf("\n\t yprv2: bt=%11.4g, r0=%11.4g\n",bt,r0);
     for (k=0;k<3;k++)
       {
         *(c+k+1)=*(f+k+4)+bt*(*(f+k+3));
         if ( lo > 1 )   // V_1+V_2
            *(d+k)=( k==0 ) ? *(f+k)+r0*(*(f+k+3)) :
*(f+k)+r0*(*(f+k+3))+(*(f+k-1))*bt;
         else
            *(d+k)=(*(f+k)*bt-r1*(*(f+k+3)));   // V_1*V_2
       }

     k=yprvn (c,d,e,4,2,0); // 0
     *(e+2)=c0;
     if ( lo==1 || lo==3 )
      { printf("\n\typrv2:V2(p)=");
        for (i=0;i<6;i++) printf("%12.3g",*(e+i)); PRN; }
     free(f);
     free(d); free(c);
     return(k);
   }
```

```
  else
   { printf("\n\tno memory for t or c in yprv2\n");
     return(-1); }
 }

/*ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ yprv2.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ*/


int yprv3 (long double *e, long double *cv, long double *a, int lo)
 {
   register int i, k;
   long double bt, c0, r0, r1, r2, *d, *c, *f;
   c=(long double *)calloc(10,sizeof(long double));
   d=(long double *)calloc(10,sizeof(long double));
   f=(long double *)calloc(10,sizeof(long double));
   if ( d!=NULL && c!=NULL && f!=NULL )
    { /* cv=[cv_0=A; cv_1=al];

[cv_0+cv_1][V_2(p)=(a_2/p+(a_0+a_1*p)/(a_3+a_4*p+a_5*p^2)]/(p+cv_2);
        Exit: V_2(p)=(e_0+e_1p)/(e_3+e_4*p+e_5*p^2)+e_2/p.
      */
     if ( lo==1 )
      {
printf("\n\typrv2:[%10.4g+%10.4g*V2(p]/(p+%10.4g)=\n",*(cv),*(cv+1),*(cv+2))
;
        for (i=0;i<6;i++) printf("%9.3g",*(a+i));
      }
     for (i=0;i<6;i++) *(f+i)=*(a+i);
     *(f+6)=0.;
     bt=1/(*(cv+2));

     r0=*(cv+1)*bt*(*(f+2));
     c0=*cv-r0;
     *c=1;
     *(f+2)=0.;
     for (k=0;k<3;k++)
       { *(c+k+1)=*(f+k+4)+bt*(*(f+k+3));
         *(d+k)=(c0*(*(f+3))+(*(cv+1))*(*(f+k)))*bt;
       }
     k=yprvn (c,d,e,4,2,lo); // 0
     *(e+2)=r0;
     if ( lo==1 )
      { printf("\n\typrv3:V2(p)=");
        for (i=0;i<6;i++) printf("%12.3g",*(e+i)); PRN; }
     free(f);
     free(d); free(c);
     return(k);
   }
  else
   { printf("\n\tno memory for t or c in yprv3\n");
     return(-1); }
 }

/*ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ yprv2.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ*/
```

```
int yprvn (long double *a, long double *b, long double *y, int nh, int jq,
int lo)
 {
  register int i, k;
  int kt, lt, lk, k1, k2, kn, kl;
  long double *t1, *t2, *al, *ty, *c, *d, *e, r0, r1, e0=.001;
  if (lo==1 )
   { printf("\n\typrv: nh=%u: b & a\n",nh);
     for (i=0;i<nh;i++)
       printf("%11.4g%c",*(b+i), (i%8==7 || i==nh-1) ? '\n' : ' ');
     for (i=0;i<nh;i++)
       printf("%11.4g%c",*(a+i), (i%8==7 || i==nh-1) ? '\n' : ' ');
   }
  kn=2*jq;
  e=(long double *)calloc(30,sizeof(long double));
  d=(long double *)calloc(30,sizeof(long double));
  c=(long double *)calloc(30,sizeof(long double));
  t1=(long double *)calloc(30,sizeof(long double));
  t2=(long double *)calloc(30,sizeof(long double));
  al=(long double *)calloc(30,sizeof(long double));
  ty=(long double *)calloc(30,sizeof(long double));
  if ( ty!=NULL && al!=NULL && t1!=NULL && t2!=NULL && c!=NULL && d!=NULL &&
e!=NULL )
   {
     for (i=0;i<nh;i++)
       { *(t2+i)=*(b+i)/(*a*(*b));
         *(t1+i)=*(a+i)/(*a); }
     *(ty)=*t2/(*t1);
     for ( i=0; i<nh; i++) *(e+i)=*(t1+i+1)-1/(*t2)*(*(t2+i+1));
     *(ty+1)=*e;
     for ( k=3; k<=kn; k++)
       {
         for ( i=0; i<nh-k/2; i++)
           { *(t1+i)=*(t2+i);
             *(t2+i)=*(e+i); }
       for ( i=0; i<nh-k/2; i++) *(e+i)=(*(t1+i+1))/(*(ty+k-3))-
(*(t2+i+1))/(*(ty+k-2));
       *(ty+k-1)=*e;
       }
     for (i=0;i<kn;i++) *(t1+i)=*(t2+i)=*(e+i)=0.;
     *t1=*t2=*e=1.;
     for (k=1; k<kn; k++)
       {
         for (i=1;i<=k;i++) *(e+i)=*(ty+k)*(*(t1+i-1))+(*(t2+i));
       for (i=0;i<kn;i++)
           { *(t1+i)=*(t2+i);
             *(t2+i)=*(e+i); }
       }
     for (i=0;i<=jq;i++) *(y+i+jq+1)=*(e+i);

     for (i=0;i<nh;i++) *(t1+i)=*(t2+i)=*(e+i)=0.;
     *t2=1.;
     *e=*ty;
     for (k=1; k<kn; k++)
       { for (i=1;i<=k;i++) *(e+i)=*(ty+k)*(*(t1+i-1))+(*(t2+i));
       for (i=0;i<kn;i++)
```

```
           { *(t1+i)=*(t2+i);
             *(t2+i)=*(e+i); }
       }
     for (i=0;i<=jq;i++) *(y+i)=*(e+i)*(*ty)*(*b);
     if (lo==1 )
      { printf("\n\typrv: y[%u]:\n",jq);
        for (i=0;i<=(kn+1);i++)
      printf("%12.5g%c",*(y+i), (i%(jq+1)==jq) ? '\n' : ' '); }
     free(ty); free(al);
     free(t2); free(t1);
     free(c); free(d);
     free(e);
     return(1);
   }
  else
   { printf("\n no memory for ty, tt, t2 or t1 in yprv: nh=%u\n",nh);
     return(-1); }
 }

/* ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ yprv.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ */

void orig1 (long double *a, long double *r, int lo)
 {
   int i;
   long double r0;
 /*  if ( lo==1 )
    { printf("\n\t orig1: r(p)=r_2/p+(r_0+r_1p)/(r_3+r_4p+r_5p^2)=\n");
      for (i=0;i<6;i++) printf("%12.3f",*(r+i)); PRN; }
  */
   for (i=0;i<6;i++) *(a+i)=*(r+i)/(*(r+5));
             /* F(p) -> f(t)  */
   // if ( lo==1 ) { for (i=0;i<6;i++) printf("%12.3f",*(a+i)); PRN; }
   *(a+4)/=2.;
   r0=*(a+4)*(*(a+4))-(*(a+3));
   *(a+3)=( r0 < 0 ) ? -1. : 1.;
   *(a+5)=sqrt(*(a+3)*r0);
   *(a+2)=*(a+1);
   *(a+1)=(*a-(*(a+4))*(*(a+2)))/(*(a+5));
   *a=*(r+2);
 //  for (i=0;i<6;i++) printf("%12.3f",*(a+i)); PRN;
   if ( lo==1 )
    {
      if ( *(a+3)==-1 )
       printf("\nF(t)=%8.3f+exp(-
%6.3f*t)*[%11.4f*sin(%6.3f*t)+%11.4f*cos(%6.3f*t)]",*a,*(a+4),*(a+1),*(a+5),
*(a+2),*(a+5));
      else
       if ( *(a+3)==1 )
       printf("\nF(t)=%8.3f+exp(-
%6.3f*t)*[%11.4f*sh(%6.3f*t)+%11.4f*ch(%6.3f*t)]",*a,*(a+4),*(a+1),*(a+5),*(
a+2),*(a+5));
      PRN;
    }
 }

int pslfn (long double *ay, long double *b, int nk, int lo)
```

```
{
  int i, k, ml, n1, n2, i3;
  long double *c, c0, r0;
  n2=6;
  n1=2*n2;
  ml=6*nk;
  c=(long double *)calloc(ml,sizeof(long double));
  if ( c!=NULL )
   {
     for (k=0,c0=0.;k<nk;k++)
       { i3=k*n2;
         c0+=*(b+i3+2);  }
 /*        *(b+i3+2)=0.;
         r0=*(b+i3+3);
         for (i=0;i<6;i++) *(b+i3+i)/=r0;
       }
     for (k=1;k<nk;k++)
       { i3=k*n2;
         if (fabs(*(b+i3)) < 0.0001 && fabs(*(b+i3+1)) < 0.0001 )
                { nk-=1;
                  for (i=0;i<n2;i++) *(b+i3+i)=*(b+i3+n2+i); }
       }
  */
     if ( lo==1 )
      { printf("\n\tpslfn input:  nk=%u\n",nk);
        for (i=0;i<ml;i++) printf("%9.4g%c",*(b+i), (i%6==5) ? '\n' : ' ');
}

     if ( nk > 1 )
      { for (i=0;i<n1;i++) *(c+i)=*(b+i);
        pslfn2 (ay,c,0);
        if ( nk > 2 )
         { for (k=0; k<nk-2; k++)
             { i3=n2*(k+2);
               for (i=0;i<n2;i++) *(c+i)=*(ay+i);
               for (i=0;i<n2;i++) *(c+i+n2)=*(b+i+i3);
               pslfn2 (ay,c,0); }
         }
      }
     else for (i=0;i<n2;i++) *(ay+i)=*(b+i);
   /*  r0=*(ay+5);
     for (i=0;i<6;i++) *(ay+i)/=r0;  */
      *(ay+2)=c0;
      if ( lo==1 )
       { printf("\n pslf(%u): ay=",nk);
         for (i=0;i<6;i++) printf("%10.4g",*(ay+i)); PRN; }
      free(c);
      return(nk);
   }
  else
   { printf("\n no memory for c in pslf\n");
     return(-1); }
}

int pslfn2 (long double *a, long double *b, int lo)
 {
```

```
    int i, i3, k, ml, n2;
    long double *c, *d, *f, c0, r1;
    n2=6;
    c=(long double *)calloc(20,sizeof(long double));
    d=(long double *)calloc(20,sizeof(long double));
    f=(long double *)calloc(20,sizeof(long double));
    if ( d!=NULL && c!=NULL && f!=NULL )
     {
       if ( lo==1 )
        { printf("\n pslf2 input b=\n");
          for (i=0;i<12;i++) printf("%9.4g%c",*(b+i), (i%n2==n2-1) ? '\n' :
' '); }
       for (k=0;k<12;k++) *(f+k)=*(b+k);
       for (k=0,c0=0.;k<2;k++)
         { i3=k*n2;
           c0+=*(f+i3+2);
           *(f+i3+2)=0.;
           r1=*(f+i3+3);
           for (i=0;i<6;i++) *(f+i3+i)/=r1; }
       for (i=0;i<12;i++) *(d+i)=*(c+i)=0.;
       for (k=0;k<3;k++) for (i=0;i<3;i++) *(c+k+i)+=*(f+3+k)*(*(f+9+i));
       for (k=0;k<2;k++)
           for (i=0;i<3;i++)
*(d+k+i)+=*(f+k)*(*(f+9+i))+(*(f+6+k))*(*(f+3+i));
       yprvn (c,d,a,5,2,0);
       *(a+2)=c0;
       if ( lo==1 )
        { printf("\n pslf2:  a=");
          for (i=0;i<6;i++) printf("%11.5g",*(a+i)); PRN; }
       free(f);
       free(d); free(c);
       return(1);
     }
    else
     { printf("\n no memory for c, d in pslf2\n");
       return(-1); }
  }

/* ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ pslf.c ÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ */

int pslf1 (long double *ay, long double *a, int nk, int lo)
 {
   int i, k, k2, ml, n1, n2, i3;
   long double *c, *d, *e, *f, *g, c0;
   n2=2*nk;
   if ( lo==1 )
    { printf("\n\tpslf1 input: sum_{k=1}^%u [a_{2k}/(p+a_{2k+1}]\n",nk);
      for (i=0;i<n2;i++) printf("%9.4g%c",*(a+i), (i%nk==nk-1) ? '\n' : '
');
      PRN; }
   c=(long double *)calloc(20,sizeof(long double));
   d=(long double *)calloc(20,sizeof(long double));
   e=(long double *)calloc(20,sizeof(long double));
   f=(long double *)calloc(20,sizeof(long double));
   g=(long double *)calloc(20,sizeof(long double));
   if ( g!=NULL && c!=NULL && d!=NULL && e!=NULL && f!=NULL )
```

```
    {
      for (i=0;i<20;i++) *(c+i)=*(d+i)=*(f+i)=0.;
      for (k=0;k<nk;k++)
        { k2=2*k;
          *(g+k2)=*(a+k2)/(*(a+k2+1));
          *(g+k2+1)=1/(*(a+k2+1)); }

      *f=*g+(*(g+2));
      *(f+1)=*g*(*(g+3))+(*(g+2))*(*(g+1));
      *(e+2)=*(g+3)*(*(g+1));
      *(e+1)=*(g+3)+(*(g+1));
      *e=1.;
      for (k=0;k<3;k++)
        { *(ay+k)=*(f+k);
          *(ay+k+3)=*(e+k); }
      if (nk > 2 )
       { for (k=2;k<nk;k++)
           { k2=2*k;
             *d=*f+(*e)*(*(g+k2));
             for (i=1;i<k+2;i++) *(d+i)=*(f+i)+(*(f+i-
1))*(*(g+k2+1))+(*(e+i))*(*(g+k2));
             *c=*e;
             for (i=1;i<k+2;i++) *(c+i)=*(e+i-1)*(*(g+k2+1))+(*(e+i));
             for (i=0;i<k+2;i++)
               { *(f+i)=*(d+i);
                 *(e+i)=*(c+i); }
           /*  printf("\n\t f(%u), e(%u)\n",k+1,k+1);
             for (i=0;i<k+2;i++) printf("%12.4g",*(f+i)); PRN;
             for (i=0;i<k+2;i++) printf("%12.4g",*(e+i)); PRN;
           */
           }
         yprvn (c,d,ay,nk+1,2,lo);
       }
      if ( lo==1 )
       { printf("\n pslf1(%u)=",nk);
         for (i=0;i<6;i++) printf("%11.4g",*(ay+i)); PRN; }
      free(g);
      free(f); free(e);
      free(d); free(c);
      return(nk);
    }
   else
    { printf("\n no memory for c in pslf1\n");
      return(-1); }
 }


 long double grafik3 (long double *bp, long double *sl, long double *h, int
mm, int nn, int m6)
  {
    /* bp(r,z,t) to res(r,z,t); m6 -- number of points for grafik;
       h=[ra,rb,z0,zl,t0, tl,h]; sl[2*mm] -- self-values for R_m and Z_k;
     */

    FILE *fcl;
    int i, j, k, n, m, m1, m2, nb, mb, n1, n2, k1, k2;
```

```
    long double *tn, *c, rr, zz, tt, dr, dz, dt, s0, r1, r2, rs, rc, f1, g,
tk,dlt;
    m1=m6*m6*m6+10;
    c=(long double *)calloc(10,sizeof(long double));
    tn=(long double *)calloc(m1,sizeof(long double));
    if ( tn==NULL || c==NULL )
     { printf("\n\t c, tn in grafik3==0\n");
       return(-1); }
    else
     {
       printf("\n\t ra,rb,z0,zl,t0,tl,h\n");
       for (j=0;j<7;j++) printf("%6.2f",*(h+j)); PRN;
       for (j=0;j<mm;j++) printf("%10.4g",*(sl+j)); PRN;
       for (j=0;j<nn;j++) printf("%10.4g",*(sl+mm+j)); PRN;
       dz=(*(h+3)-(*(h+2)))/m6;
       dt=(*(h+5)-(*(h+4)))/m6;
       dr=(*(h+1)-(*(h)))/m6;
       rr=*h+dr;
       g=*(h+6);
       printf("\n\t dr=%7.3f, dz=%7.3f, dt=%7.3f, rr=%7.3f\n",dr,dz,dt,rr);
       dlt=0;
       for (m1=0; m1<m6;m1++)
         {
           m2=m1*m6*m6;
           zz=0.;
           for (n1=0;n1<m6;n1++)
             { n2=n1*m6;
               tt=0;
               for (k1=0;k1<m6;k1++)
                 { *(tn+m2+n2+k1)=0.;
                   for (m=0;m<mm;m++)
                     { mb=6*m*mm;
                       r2=fxs ((sl+m),0.3,g,rr,0,1);
               //    printf("\n\t r2(%u)=%10.4g, n1=%u,
k1=%u\n",m,r2,n1,k1);
                       tk=0;
                       for (k=0;k<nn-1;k++)  // -1
                         { nb=6*k;
                           s0=*(sl+mm+k);
                           r1=g/s0*sin(s0*zz)+cos(s0*zz);
                           for (j=0;j<6;j++) *(c+j)=*(bp+mb+nb+j);
                           for (j=0;j<3;j++) *(c+j)*=r1;
                      // for (j=0;j<6;j++) printf("%10.4g",*(c+j)); PRN;
                           if (*(c+3) == -1 )
                            { rc=cos(*(c+5)*tt);
                              rs=sin(*(c+5)*tt); }
                           else
                            { rc=cosh(*(c+5)*tt);
                              rs=sinh(*(c+5)*tt); }
                           f1=(*c+exp(-
(*(c+4))*tt)*(*(c+1)*rs+(*(c+2))*rc))*r1;
                           tk+=f1;
                         //  printf("\n\t
tk(%u,%u,%6.3f,%6.3f)=%10.4g\n",m,k,zz,tt,tk);
                         }
```

```
                        *(tn+m2+n2+k1)+=(tk*r2); // fabs
                        dlt+=tk*r2;
                      //  printf("\n\t  tn(%u,%u,%u;m=%u; zz=%7.3f
tt=%7.3f)=%10.4g; mn=%u\n",m1,n1,k1,m,zz,tt,*(tn+m2+n2+k1),m2+n2+k1);

                      }
                    // printf("\n\t  tn(%u,%u,%u;rr=%7.3f zz=%7.3f
tt=%7.3f)=%10.4g; mnk=%u\n",m1,n1,k1,rr,zz,tt,*(tn+m2+n2+k1),m2+n2+k1);
                     tt+=dt;
                   }
                 zz+=dz;
               }
             rr+=dr;
           }
         n2=m6*m6*m6;
         printf("\n\t DT_p(%u;r,z,t); dlt=%12.4g\n",n2,dlt);
         for (m1=0; m1<m6;m1++)
           {
             m2=m1*m6*m6;
             for (n1=0;n1<m6;n1++)
               { n2=n1*m6;
                 for (j=0;j<m6;j++) printf("%9.4g",*(tn+m2+n2+j));
               PRN; }
           PRN; }
         free(tn);
         free(c);
         return (dlt);
      }
  }


 int grafik (long double *bp, long double *sl, long double *h, int mm, int
nn, int m6, int it, int lp)
  {
    /* bp(r,z,t) to res(r,z,t; m6 -- number of points for grafik;
       h=[ra,rb,z0,zl,t0, tl,h]; sl[2*mm] -- self-values for R_m and Z_k;
       lp=0 -- fixt r; - f(z,t): lp=1 -- fixt z - f(r,t); lp=2 -- fixt t -
f(r,z):
     */
    FILE *fcl;
    int i, j, k, n, m, m1, m2, nb, mb, n1, n2;
    long double *tn, *c, rr, zz, tt, dr, dz, dt, tl, s0, *r1, *r2, rs, rc,
f1, g, tk;
    mb=m6*mm*nn+1;
    c=(long double *)calloc(20,sizeof(long double));
    tn=(long double *)calloc(1601,sizeof(long double));
    r1=(long double *)calloc(10,sizeof(long double));
    r2=(long double *)calloc(10,sizeof(long double));
    if ( tn==NULL || c==NULL || r1==NULL || r2==NULL )
     { printf("\n\t c, tn, r1 or r2 in grafik==0\n");
       return(-1); }
    else
     {
     /*  printf("\n\t ra,rb,z0,zl,t0,tl,h\n");
       for (j=0;j<7;j++) printf("%6.2f",*(h+j)); PRN;
       for (j=0;j<mm;j++) printf("%10.4g",*(sl+j)); PRN;
```

```
        for (j=0;j<nn;j++) printf("%10.4g",*(sl+mm+j)); PRN;
      */
        tl=*(h+5);
        dz=(*(h+3)/2-(*(h+2)))/m6;
        dt=(*(h+5)-(*(h+4)))/m6;
        dr=(*(h+1)-(*(h)))/m6;
        rr=*h;
        tt=*(h+4);
        zz=*(h+2);
        g=*(h+6);
        if ( lp==0 )  rr=*(h+1);
        if ( lp==1 )  zz=*(h+3)/2.;
        if ( lp==2 )  tt=*(h+5);
        printf("\n\t lp=%u: dr=%7.3f, dz=%7.3f, dt=%7.3f, rr=%7.3f, zz=%7.3f,
zt=%7.3f\n",lp,dr,dz,dt,rr,zz,tt);
        n2=6*mm*nn;
        for (i=0;i<n2;i++)
            printf("%10.4g%c",*(bp+i), (i%6==5) ? '\n' : ' ');
        for (m1=0; m1<m6;m1++)
          {
            m2=m1*m6;
            if ( lp < 2 ) tt=0;
            if ( lp==2 ) zz=0;

            for (n1=0;n1<m6;n1++)
              {
                *(tn+m2+n1)=0.;
                for (m=0;m<mm;m++)
                  { mb=6*m*mm;
                    *(r2+m)=fxs ((sl+m),0.3,g,rr,0,1);
                   //   printf("\n\t r2(%u)=%10.4g, n1=%u\n",m,*(r2+m),n1);
                    tk=0;
                    for (k=0;k<nn;k++)
                      { nb=6*k;
                        s0=*(sl+mm+k);
                        *(r1+k)=g/s0*sin(s0*zz)+cos(s0*zz);
                        for (j=0;j<6;j++) *(c+j)=*(bp+mb+nb+j);
                        for (j=0;j<3;j++) *(c+j)*=(*(r1+k));
                        if (*(c+3) == -1 )
                         { rc=cos(*(c+5)*tt);
                           rs=sin(*(c+5)*tt); }
                        else
                         { rc=cosh(*(c+5)*tt);
                           rs=sinh(*(c+5)*tt); }
                        f1=(*c+exp(-
(*(c+4))*tt)*(*(c+1)*rs+(*(c+2))*rc))*(*(r1+k));
                        tk+=f1;
                      }
                    *(tn+m2+n1)+=fabs(tk*(*(r2+m)));  // fabs
                  }
                if (lp < 2) tt+=dt;
                else  zz+=dz;
              }
            if ( lp==0 ) zz+=dz;
            else rr+=dr;
          }
```

```
        n2=m6*m6;
        if ( lp==0 ) printf("\n\t T_p(%u: %u;z,t;r=rb/2)\n",it,n2);
        if ( lp==1 ) printf("\n\t T_p(%u: %u;r,t;z=zl/2)\n",it,n2);
        if ( lp==2 ) printf("\n\t T_p(%u: %u;r,z,t=tl/2)\n",it,n2);
        if ( it==0 )
         { if ( lp==0 ) fcl = fopen("pt0zt.dat","w");
           if ( lp==1 ) fcl = fopen("pt0rt.dat","w");
           if ( lp==2 ) fcl = fopen("pt0rz.dat","w"); }
        else
         {
           if ( it==1 )
            { if ( lp==0 ) fcl = fopen("pt1zt.dat","w");
              if ( lp==1 ) fcl = fopen("pt1rt.dat","w");
              if ( lp==2 ) fcl = fopen("pt1rz.dat","w"); }
           else
            {
              if ( it==2 )
               { if ( lp==0 ) fcl = fopen("pt2zt.dat","w");
                 if ( lp==1 ) fcl = fopen("pt2rt.dat","w");
                 if ( lp==2 ) fcl = fopen("pt2rz.dat","w"); }
              else
               { if ( lp==0 ) fcl = fopen("pt3zt.dat","w");
                 if ( lp==1 ) fcl = fopen("pt3rt.dat","w");
                 if ( lp==2 ) fcl = fopen("pt3rz.dat","w"); }
            }
         }
        for (k=0; k<n2; k++) fprintf(fcl,"%g,",*(tn+k));
        fclose(fcl);
        free(r2); free(r1);
        free(tn); free(c);
        return (1);
      }
  }

int ftgtp (long double *a, long double *b, long double *c, int lo)
 {
  /* f(t)=b_0+e^(-b_4*t)*[b_1*f1(b_5*t)+b_2*f2(b_5*t)].
     g(t)=c_0+e^(-c_4*t)*[c_1*f1(c_5*t)+c_2*f2(c_5*t)].
     (b_3=-1) f1=sin(b_5t), f2=cos(b_5t); b_6=1: f1=sh(b_5t); f2=ch(b_5t).
     exit -- V_2(p)=a_2/p+(a_0+a_1p)/(a_3+a_4p+a_5p^2).   */

  int i, k, nk, nk1, k1, k2;
 long  double *b1, *c1, *d, r1, zk, zp;
  b1=(long double *)calloc(10,sizeof(long double));
  c1=(long double *)calloc(30,sizeof(long double));
  d=(long double *)calloc(20,sizeof(long double));
  if ( b1==NULL || c1==NULL || d==NULL )
   { printf("\n\t arrays in ftgtp=0\n");
     return(-1);
   }
  else
   {
     if ( lo==1 )
      { printf("\n ftgtp: b=");
        for (k=0;k<6;k++) printf("%10.4g",*(b+k));
        printf("\n ftgtp: c=");
```

```
   for (k=0;k<6;k++) printf("%10.4g",*(c+k)); PRN;
 }
if ( ( *(b+3)==-1 && *(c+3)==-1 ) || ( *(b+3)==1 && *(c+3)==1 ) )
 { if ( *(b+3)==1 ) zp=-1;
   if ( *(b+3)==-1 ) zp=1;
//   printf("\n\tftgtp: zp=%7.2f\n",zp);
   zk=1;
   nk=nk1=4;
   *(b1+4)=*(b1+5)=*(b+4)+(*(c+4));
   for (k=0;k<2;k++)
     { k1=2*k;
       k2=4*k;
       *(b1+2+k2)=*(b+5-k);
       *(b1+3+k2)=*(c+5-k);
       *(d+k+6)=*c*(*(b+1+k));
       *(d+k+4)=*b*(*(c+1+k));
       *(d+k1)=(*(b+1)*(*(c+2))+zk*(*(b+2))*(*(c+1)))/2.;
       *(d+k1+1)=(*(b+2)*(*(c+2))-zp*zk*(*(b+1))*(*(c+1)))/2.;
       *(b1+k)=*(b+5)+zk*(*(c+5));
       zk=-zk;
     }
   for (k=0;k<4;k++)
     { k2=6*k;
       k1=2*k;
       *(c1+k2)=*(d+k1)*(*(b1+k))-(*(b1+4+k))*(*(d+k1+1));
       *(c1+k2+1)=*(d+k1+1);
       *(c1+k2+3)=*(b1+k+4)*(*(b1+k+4))+zp*(*(b1+k))*(*(b1+k));
       *(c1+k2+4)=2*(*(b1+k+4));
       *(c1+k2+5)=1;
     }
   if ( fabs(*(b1+1)) < 0.001 )
    {
      *b1=*(d+3);
      *(b1+1)=*(b1+5);
      yprv2 (a,b1,c1,1,0);
      for (i=0;i<6;i++) *(c1+i)=*(a+i);
      nk-=1;
      for (k=1;k<nk;k++)
        { k2=6*(k+1);
          k1=6*k;
          for (i=0;i<6;i++) *(c1+k1+i)=*(c1+k2+i); }
    }
   if ( fabs (*c)<0.0001 && fabs (*b)<0.0001 )  nk-=2;
   else
    {
      if ( fabs (*c)<0.0001 || fabs (*b)<0.0001 )
       { nk-=1;
         k2=6*nk;
         k1=6*(nk-1);
       //  printf("\n\t k1=%u; k2=%u; \n",k1,k2);
         for (i=0;i<6;i++) *(c1+k1+i)=*(c1+k2+i); }
    }
   if ( nk > 1 ) pslfn (a,c1,nk,0);
 }
else
 {
```

```
        if (*(c+3)==1 )
         { for (i=0;i<6;i++)
             { *(b1+i)=*(c+i);
               *(c+i)=*(b+i);
               *(b+i)=*(b1+i); }
         }
        for (k=0;k<6;k++) printf("%9.3g",*(b+k)); PRN;
        for (k=0;k<6;k++) printf("%9.3g",*(c+k)); PRN;
        *b1=0.5*(*(b+1)+(*(b+2)));
        *(b1+1)=0.5*(*(b+1)-(*(b+2)));
        *(b1+2)=*b;
        *d=*(c+4);
        *(d+1)=*(b+4)+(*(c+4))-(*(b+5));
        *(d+2)=*(b+4)+(*(c+4))+(*(b+5));

        for (k=0;k<3;k++)
          { k2=6*k;
            *(c1+k2)=*(b1+k)*(*(c+k)*(*(c+5))-(*(d+k))*(*(c+2)));
            *(c1+k2+1)=*(b1+k)*(*(c+2));
            *(c1+k2+2)=0;
            *(c1+k2+3)=*(d+k)*(*(d+k))+(*(c+5))*(*(c+5));
            *(c1+k2+4)=2*(*(d+k));
            *(c1+k2+5)=1;
          }
        nk= ( fabs (*b) <0.0001 ) ? 2 : 3;
        if ( fabs (*c) > 0.0001 )
         { k2=6*nk;
           *(c1+k2)=*c*(*(b+1)*(*(b+5))-(*(b+2))*(*(b+4)));
           *(c1+k2+1)=*c*(*(b+2));
           *(c1+k2+2)=0;
           *(c1+k2+3)=*(b+4)*(*(b+4))-(*(b+5))*(*(b+5));
           *(c1+k2+4)=2*(*(b+4));
           *(c1+k2+5)=1;
           nk+=1;
         }
        pslfn (a,c1,nk,0);
      }
     r1=*(a+5);
     for (k=0;k<6;k++) *(a+k)/=r1;
     *(a+2)=*b*(*c);
     if ( lo==1 )
      { printf("\n ftgtp  out:  a= ");
        for (k=0;k<6;k++) printf("%10.4g",*(a+k)); PRN; }
     free(d);
     free(c1); free(b1);
     return(nk);
   }
 }
```

Д 2.2 Програмна реалізація інтегрування добутків циліндричних функцій.

```
/*------------------------ neiman.c ------------------------*/
```

```
#include "bstr.h"

 void slfr (long double *, long double *, int, int);
 int pslf2 (long double *, long double *, int, int);
 void slfrb2 (long double *, long double *, int, int);
 long double BS2 (long double, long double, long double, int, int);
 void slfz (long double *, long double *, int);
 int gornd (long double *, long double *, long double *, int);
 int yprvd (long double *, long double *, long double *, int, int, int);
 int xalsd (long double *, long double *, int, int);
 int sppd (long double *, long double *, long double *, int, int);
 int svpd (long double *, long double *, int, int);
 long double inbsz (long double *, long double, long double, int, int);

 int pzsd2 (long double *, long double *, long double *, long double *, int
*);
 long double inzbsn_1 (long double *, long double, long double, long double,
int);
 long double inzbsn_2 (long double *, long double *, long double, long
double, int *);
 long double inzbsn_3 (long double *, long double *, long double, long
double, int *);
 long double intbs2_2 (long double *, int);
 long double intbzm (long double *, long double *, long double *, long
double *, long double, long double, int *, int *, int);

 int bsnd (long double *, long double *, long double *, long double *, int,
int);
 int lnsd (long double *, long double, int);
 int bnsd (long double *, long double *, int);
 int bs12 (long double *, long double *, long double *);
 int bs22 (long double *, int);
 long double bsd2 (long double *, long double *, long double, long double,
int);
 int ppspd (long double *, long double *, long double *, long double *, int,
int);
 int bszm (long double *, long double *, long double *, long double *, long
double, long double, int);
 int prdbc (float *, float *, int);

 int gskl (long double *, long double *, long double *, int, int, int, int);

main()
 {
   FILE *fcl;
   int i, j, k, l, m, m2, m6, ml, *nb, nj, mj=6, mv=6, k1, k2, k6, ki, *ns,
n, nl, n1, n2, n3, nk, np;
   long double  r0, r1, r2, cj, s1, s0, sr, x1, x2, sr1, d0, d1, d2, rm, rb,
ra, rln, cfi, cf, sf, zk, hz, gn;
   long double *c, *gs, *d, *u, *uz, *v, *cl, *cl2, *cnk, *sl, *hl, a, b,
bs2, bs3, bs0, bs1;
   long double *hg, *hgl, x, *h, *r, *bs, *znk, *rnk, *rn, *zn, um, cb;
   static int nbs[5], nsl[5];
   static long double ud[60], c1[50], g1[250], bv1[10], bv[10], cv[40],
rr[10], v1[20], cnk2[50],
```

```
                hg3[100], hg1[200], hl1[200], hl2[200], h1[50], znr[10],
znf[10], rnk1[100], rnk2[100],
   uin[]={5.,6.,10, .5,1, 25.,.25, 1.3, .75, 1.2, .76,.35, 0.5, .3,
.97,.685,0.12,0.23,0.,0.,0.,0.,0.},
               hh[]={1,16,36,16,1, -2.285, -11.0967, 13.2285, 22.8553,
3.02};
         //  b1[]={13,9,1,3.5,4,0}, dd1[]={2,3,1,1,2,0};

  /* uin=[0-L_0,1-L_1,2-L_k,3-R_0,4-R_k,5-w_0, 6-fi, 7-rho, 8-c_p, 9-u_r,
10-nu, 11-lbd_T, 12-mu, 13-T1_0, 14-u_wx,15-w_wx ] */

   c=c1; znk=znr;
   r=rr; h=h1;
   rn=rnk1; rnk=rnk2;
   zn=znf;  cnk=cnk2;
   hg=hg1; hl=hh;
   hgl=hg3; bs=hl2;
   gs=g1; v=v1;
   nb=nbs; ns=nsl;
   cl=bv; sl=cv;
   cl2=bv1; u=ud;
   uz=uin;
   rb=*(uz+4);
   ra=*(uz+3);
  //  slfr(cl,v,mj,mv);
   *v=0.4;
   *(v+1)=ra;
   *(v+2)=rb;
   *(v+3)=2;
   *(v+4)=1.;
   *(v+5)=0.5;
 /*
   k1=150;
   a=*(v+1);
   b=*(v+2);
   cb=a/b;
   for (m=0;m<3;m++)
     { x=1;
       printf("\n\t main:\t [h0=%6.2f, a=%6.2f, b=%6.2f]; g(x)=\n",*v,a,b);
       for (k=0;k<50;k++)
         { ki=m*50;
           bs2=BS2(x,cb,1,0);
           bs3=BS2(x,cb,1,1);
           bs0=BS2(x,cb,0,0);
           bs1=BS2(x,cb,0,1);
           r2=a/b*(*v)*bs0+(a/b*x/b)*bs1-(x/b+(*v)/x)*bs2-(1/b+(*v))*bs3;
           r1=x/b*bs3+(*v)*bs2;
           *(c+k+ki)=r1;
           printf("x=%7.2f; f(x)=%11.3g;\t df(x)=%11.3g: \t
dx=%12.4g\n",x,r1,r2,r1/r2);
           x+=0.5;
         }
       a+=0.2;
     }
   mj=1;
   fcl = fopen("gslpol0.dat","w");
```

```
   for (i=0; i<k1; i++) fprintf(fcl,"%g ",*(c+i));
    fclose(fcl);
  */

   slfrb2(cl2,v,mj,mv);

   for (i=0;i<mv;i++) *(cl2+i)/=2;
 /*  sr=*(cl2);
    *v=0.4;
   *(v+1)=1.;
   *(v+2)=.2;
   *(v+3)=1;
   *(v+4)=2.;
   *(v+5)=1.;
   slfz(sl,v,mv);
   np=5;
   x=10;
   np=lnsd (hl,x,nl);
   printf("\n\t main:\t np=%u\n",np);

   bsnd (hg, bs, hgl, hl, np, 0);

   bszm (cnk, hg, bs, hgl, sr, rb, 2);
   intbs2_2 (cnk, 2);

   n=mv;
   n2=n*n;

    for (k=0;k<1;k++)
      { k1=k*mv;
        s1=*(cl2+k);
        *nb=k;
        *(nb+1)=k;
        *(nb+2)=k;
        gn=yn(1,s1*ra)/jn(1,s1*ra);
        *(rn+k)=(yn(1,s1*rb)-gn*jn(1,s1*rb))*s1*rb;
        for (l=0;l<6;l++)
          { *ns=l;
            *(ns+1)=l;
            *(ns+2)=l;
            printf("\n\t  k=%u,  l=%u\n",k,l);

            printf("\n\t main: nb_0=%u, nb_1=%u\t ns_0=%u,  ns_1=%u;\n",*nb,
*(nb+1),*ns,*(ns+1));
            printf("\n\t main: Yn\n");
            for (i=0;i<21;i++) printf("%10.4g%c",*(hg+i), (i%7==6) ? '\n' :
' ');
                printf("\n\t main: hb\n");
            for (i=0;i<18;i++) printf("%10.4g%c",*(bs+i), (i%6==5) ? '\n' :
' ');
               printf("\n\t main: hl\n");
            for (i=0;i<18;i++) printf("%10.4g%c",*(hgl+i), (i%6==5) ? '\n' :
' ');

            *(rn+i)=intbzm (hg,hgl,bs,cl2,rb,ra,nb,ns,1);
            *(rnk+l)=intbzm (hg,hgl,bs,cl2,rb,ra,nb,ns,2);
```

```
            *(rn+l)/=*(rnk+k1+l);
            intbzm (hg,hgl,bs,cl2,rb,ra,nb,ns,3);
          }
      }


    k1=n*n;
    printf("\n\t int Z_n(s*rb) --  array rn(%u,%u)\n",n,n);
    for (j=0;j<n2;j++) printf("%8.3g%c",*(rn+j), ( j%n==n-1 ) ? '\n' : ' ');
    printf("\n\t normbes2 --  array rnk(%u,%u)\n",n,n);
    for (j=0;j<n2;j++) printf("%8.3g%c",*(rnk+j), ( j%n==n-1 ) ? '\n' : '
');
    printf("\n\t int Z^3_n(s*rb) --  array rnk(%u,%u)\n",n,n);
    for (j=0;j<n2;j++) printf("%8.3g%c",*(rnk+n2+j), ( j%n==n-1 ) ? '\n' : '
');

    cfi=*(uz+6);
    um=*(uz+12)/(*(uz+7));
    printf("\n\t CIKL0 --  fi=%9.3g; um=%9.3g; u(16)=\n",cfi,um);
    for (j=0;j<16;j++) printf("%8.3g%c",*(uz+j), ( j%8==7 || j==15 ) ? '\n'
: ' ');

    n2=n*n;
    n3=n*n2;
    hz=*(uz+2);
    for (k=0;k<n;k++)
      { zk=*(sl+k);
        r1=zk/(*(uz+10));
        *(znk+k)=sqrt( hz/2*(r1*r1+1)+(r1*r1-1)/(4*zk)*sin(2*zk*hz)+(1-
cos(2*zk*hz))/(2*(*(uz+10))));
        *(zn+k)=((1-cos(zk*hz))/zk-sin(zk*hz)/(*(uz+10)))/(*(znk+k)); }

    for (k=0;k<n;k++)
      { nk=k*n;
        *nb=k;
        cf=cfi*(1-cos(cfi*2*pi))/((cfi*cfi-k*k)*pi);
        sf=k*sin(cfi*2*pi)/((cfi*cfi-k*k)*pi);
        for (i=0;i<n;i++)
          { ki=(k*n+i)*n;
            *ns=i;
            r1=*(cl2+nk+i)*(*(cl2+nk+i));
            r2=*(rn+nk+i);
            for (j=0;j<n;j++)
              {
                *(gs+j+ki)=um*(*(sl+j)*(*(sl+j))+r1);
                *(cnk+j+ki)=*(zn+j)*r2*cf*(*(uz+5));
                *(cnk+j+ki+n3)=*(zn+j)*r2*sf*(*(uz+5));
              }
          }
      }
    printf("\n\t self values gs(%u,%u,%u)\n",n,n,n);
    k1=n2*n;
    for (j=0;j<k1;j++) printf("%8.3g%c",*(gs+j), ( j%n==n-1 ) ? '\n' : ' ');

    printf("\n\t self values zs(%u)\n",n);
    for (j=0;j<n;j++) printf("%9.4g",*(sl+j)); PRN;
```

```
    fcl = fopen("gslcikl.dat","w");
    for (i=0; i<k1; i++) fprintf(fcl,"%g,",*(gs+i));
    fclose(fcl);
    fcl = fopen("nrmcikz.dat","w");
    for (i=0; i<n; i++) fprintf(fcl,"%g,",*(znk+i));
    fclose(fcl);
    fcl = fopen("slfcikz.dat","w");
    for (i=0; i<n; i++) fprintf(fcl,"%g,",*(sl+i));
    fclose(fcl);

    k2=2*k1;
    fcl = fopen("lincikz.dat","w");
    for (i=0; i<k2; i++) fprintf(fcl,"%g,",*(hgl+i));
    fclose(fcl);
   */
 }

void slfrb2 (long double *z, long double *h, int m, int n)
 {
   /* h=[h, r, R, x0, al_0, al_1]; m - order of Bessel; n - number of self-
values.
      z[m,n] -- array of self-values; x0=initial value of x;  h=al_1;
      J_m(lb*r)*Y_m(lb*R)-Y_m(lb*r)*J_m(lb*R)=0.
      f(x)= al_0*x*[jn(m,x*a)*d[yn(m,x*b)]/dx-yn(m,x*a)*d[jn(m,x*b)]/dx]
            +al_1*[jn(m,x*a)*yn(m,x*b)-yn(m,x*a)*jn(m,x*b)]=0.
                output: array z:  slfr2.dat (m,n)  */

   FILE *fcl;
   int i, ii, k, ki, km;
   long double a, b, c, h0, a0, a1, x, dx, x1, r1, f0, f1, bs0, bs1, ys0,
gnp, gx;
   km=m*n;
   printf("\n\t slfr2:\t h=%7.2f,  a=%7.2f, b=%7.2f,
x1=%7.2f\n",*h,*(h+1),*(h+2),*(h+3));
   h0=*h;
   x1=1;
   a=*(h+1);
   b=*(h+2);
   c=a/b;
   n=6;
        x=2;
      r1=x*c;
              bs0=yn(0,x)-yn(1,r1)/jn(1,r1)*jn(0,x);
              bs1=yn(1,x)-yn(1,r1)/jn(1,r1)*jn(1,x);
              ys0=yn(0,r1)*jn(1,r1)-jn(0,r1)*yn(1,r1);
              gnp=c*ys0/(jn(1,r1)*jn(1,r1));
              gx=x*bs1-h0*b*bs0;
              printf("\n\t gx(%8.3g)=%9.3g\n",x,gx);
   for (k=0; k<1; k++)
     { ki=k*n;
       for (i=0; i<n; i++)
         { x=x1;
         dx=1.0;
         printf("\n\t x(%u)=%10.3g,  dx=%10.3g\n",i,x,dx);
         while (  fabs(dx) > .02 )
```

```
          { r1=x*c;
              bs0=yn(0,x)-yn(1,r1)/jn(1,r1)*jn(0,x);
              bs1=yn(1,x)-yn(1,r1)/jn(1,r1)*jn(1,x);
              ys0=yn(0,r1)*jn(1,r1)-jn(0,r1)*yn(1,r1);
              gnp=c*ys0/(jn(1,r1)*jn(1,r1));
               printf("\n j0=%8.3g; y0=%8.3g; bs0=%9.4g; bs1=%9.4g;
ys0=%9.4g;  gnp=%9.4g\n",jn(0,x),yn(0,x),x*bs0,h0*bs1,ys0,gnp);
              f0=x*bs1-h0*b*bs0;
              f1=x*bs0+h0*b*bs1+gnp*(h0*b*jn(0,x)-x*jn(1,x));

            dx=f0/f1;
              x-=dx;
              printf("\n  f0=%9.3g, f1=%9.3g, dx=%10.4g;
x=%9.3g\n",f0,f1,dx,x);
             }
           printf("\n x1=%10.4g; x(%u)=%9.3g\n",x1,i,x);
           *(z+i)=x/b;
          // *(z+i+n)=(yn(1,x)-jn(1,x))*x;
          // r1=jn(k,*(z+i)*a)*jn(k,*(z+i)*a);
          // *(z+i+n+n)=(r1-jn(k,x)*jn(k,x))/(pi*(*(z+i))*r1);
             x1=pi+x;

           // printf("\n \t x1=%10.4g; k=%u, i=%u;\t x=%10.4g\n",x1,k,i,x);
        }
     }
   printf("\n lb[J_m(lb*a)Y'_m(lb*b)-
Y_m(lb*a)J'_m(lb*b)]+h_k[J_m(lb*a)Y_m(lb*b)-Y_m(lb*a)J_m(lb*b)]=0.\n");
   printf("\n\t self-vlues for bessel2(h_k=%6.2f: r_0=%7.2f,r_1=%7.2f):
n=%u\n",*h,*(h+1),*(h+2),n);
   for (i=0;i<n;i++) printf("%10.5g%c",*(z+i), (i%n==n-1) ? '\n' : ' ');
   printf("\n\t\int[Z_k]: h_k=%6.2f: r_0=%7.2f,r_1=%7.2f):
\n",*h,*(h+1),*(h+2));
   for (i=0;i<n;i++) printf("%10.5g%c",*(z+i+n), (i%n==n-1) ? '\n' : ' ');
   /*
   fcl = fopen("slfrp.dat","w");
   for (i=0; i<n+n; i++) fprintf(fcl,"%g,",*(z+i));
   fclose(fcl);
   */
 }

 int bszm (long double *cz, long double *hz, long double *hy, long double
*hg, long double sr, long double ra, int mb)
  {
   /* mb=2:
                                          cz[n,6k+3] + cz[n,6k+4]*r^2
     Zn(bt*r)=Yn(bt*r)-Gn*Jn(bt*r)=----------------------------------------
------(k=0,3)
                                    cz[n,6k] + cz[n,6k+1]*r^2 +
cz[n,6k+2]*r^4

                                             cz[n,20+6k+3] + cz[n,20+6k+4]*r
                                  + ----------------------------------------
------------(k=0,3);
                                    cz[n,20+6k] + cz[n,20+6k+1]*r +
cz[n,20+6k+2]*r^2
    mb=1:
```

```
                    cz[n,6k+3] + cz[n,6k+4]*r^2
 Jn(bt*r)=-----------------------------------------(k=0,3)
          cz[n,6k] + cz[n,6k+1]*r^2 + cz[n,6k+2]*r^4
*/
FILE *fcl;
int i, j, k, k6, kz, ks;
long double s2, gn, r1, r2, *b, *c;
b=(long double *)calloc(50,sizeof(long double));
c=(long double *)calloc(50,sizeof(long double));
if ( b!=NULL && c!=NULL )
 {
/*
   printf("\n\t bszm:   sr=%9.4g: rb=%7.3f; hz(x)\n",sr,ra);
   for ( k=0;k<3;k++)
     { ks=6*k;
       for (i=0;i<6;i++) printf("%12.4g",*(hz+ks+i));  PRN; }

   printf("\n\t   hg(x)\n");
   for ( k=0;k<3;k++)
     { ks=6*k;
       for (i=0;i<6;i++) printf("%12.4g",*(hg+ks+i));  PRN; }
   printf("\n\t  hy(x)\n");
   for ( k=0;k<3;k++)
     { ks=6*k;
       for (i=0;i<6;i++) printf("%12.4g",*(hy+ks+i));  PRN; }
*/
  s2=sr*sr;    // beta^2;
  r2=s2*s2;    // beta^4;
  gn=yn(1,sr*ra)/jn(1,sr*ra);
  if ( mb==1 )
   {
     for ( k=0;k<3;k++)
       { k6=6*k;
         *(cz+k6)=*(hg+k6)/r2;
         *(cz+k6+1)=*(hg+k6+1)/s2;
         *(cz+k6+2)=1;
         *(cz+k6+3)=*(hg+k6+3)/r2;
         *(cz+k6+4)=*(hg+k6+4)/s2;
         *(cz+k6+5)=0;
       }
     printf("\n\t output pszm:  Jn(%8.3g*r)=\n",gn,sr);
     for ( k=0;k<3;k++)
       { ks=6*k;
         for (i=0;i<6;i++) printf("%12.5g",*(cz+ks+i));  PRN; }

     fcl = fopen("bssr1.dat","w");
     for (i=0; i<18; i++) fprintf(fcl,"%g,",*(cz+i));
     fclose(fcl);
   }
  else
   { // Zn(bt*r)=Yn(bt*r)-Gn*Jn(bt*r)
     for ( k=0;k<3;k++)
       { k6=6*k;
         kz=6*k+18;
         *(c+3)=*(hg+k6)/r2;
         *(c+4)=*(hg+k6+1)/s2;
```

```
            *(c+5)=1;
            *(c)=-gn*(*(hg+k6+3))/r2;
            *(c+1)=-gn*(*(hg+k6+4))/s2;
            *(c+2)=0;
            *(c+9)=*(hy+k6)/r2;
            *(c+10)=*(hy+k6+1)/s2;
            *(c+11)=1.;
            *(c+6)=*(hy+3+k6)/r2;
            *(c+7)=*(hy+4+k6)/s2;
            *(c+8)=0.;
            pslf2 (b,c,3,0);

            for ( i=0;i<3;i++)
              { r1=*(b+5);
                *(cz+k6+i)=*(b+i+3)/r1;
                *(cz+k6+i+3)=*(b+i)/r1; }
            *(cz+kz+3)=*(hz+k6+3)/s2;
            *(cz+kz+4)=*(hz+k6+4)/sr;
            *(cz+kz+5)=0.;
            *(cz+kz)=*(hz+k6)/s2;
            *(cz+kz+1)=*(hz+k6+1)/sr;
            *(cz+kz+2)=1;
          }
       }
      printf("\n\t pszm: gn(%6.2f)=%11.4g:  Zn(%8.3g*r)=\n",ra,gn,sr);
      for ( k=0;k<3;k++)
        { ks=6*k;
          for (i=0;i<6;i++) printf("%12.5g",*(cz+ks+i));  PRN; }
      PRN;
      for ( k=0;k<3;k++)
        { ks=6*k;
          for (i=0;i<6;i++) printf("%12.5g",*(cz+ks+18+i));  PRN; }

      fcl = fopen("bssr2.dat","w");
      for (i=0; i<36; i++) fprintf(fcl,"%g,",*(cz+i));
      fclose(fcl);

      free(c); free(b);
    }
   else
    { printf("\n\t no memory for b in bszm\n");
      return(0); }
  }

/*------------------------ bsnd.c -------------------------*/

int bsnd (long double *hz, long double *hy, long double *hg, long double
*hl, int nl, int n)
 {
  /*                 hg[n,6k+3] + hg[n,6k+4]*(z^2/2)
     Jn(z^2/2)=-----------------------------------------------;k=0,2;
           hg[n,6k] + hg[n,6k+1]*(z^2/2) + hg[n,6k+2]*(z^2/2)^2
                  hz[n,6k+3] + hz[n,6k+4]*(z/2)
     Yn(z/2)=-----------------------------------------------;k=0,2;
           hz[n,6k] + hz[n,6k+1]*(z/2) + hz[n,6k+2]*(z/2)^2
     mj (output) -- number of chanes; m -- number of functions.
```

```
         bsds1.dat  - approx. J_ms(x);  bsds2.dat  - approx. Y_ms(x);
         bsdz.dat  - approx. Z_ms(x);
     */
   FILE *fcl;
   int i, k, k1, k2, nn, jp, jq;
   long double *a, *b, *c, *d, *e, *g, *t, r1, r2, x, c0;
   a=(long double *)calloc(50,sizeof(long double));
   g=(long double *)calloc(50,sizeof(long double));
   b=(long double *)calloc(50,sizeof(long double));
   t=(long double *)calloc(50,sizeof(long double));
   c=(long double *)calloc(50,sizeof(long double));
   d=(long double *)calloc(50,sizeof(long double));
   e=(long double *)calloc(50,sizeof(long double));
   if (a!=NULL && g!=NULL && b!=NULL && t!=NULL && d!=NULL && c!=NULL  &&
e!=NULL )
    {
      nn=20;
      c0=1.;
      printf("\n\t bsnd:  Ln(x)=hl_[i+%u]/hl_i\n",nl);
      for (i=0;i<nl;i++) printf("%12.4g",*(hl+nl+i));  PRN;
      for (i=0;i<nl;i++) printf("%12.4g",*(hl+i));  PRN;
      for (i=0;i<30;i++) *(a+i)=*(b+i)=*(e+i)=0.;
      *a=*b=1.;
      if ( n > 0 ) for (i=1;i<=n;i++) (*b)/=(i*1.);
      for (i=1;i<nn;i++) *(b+i)=-(*(b+i-1))/(i*(i+n)*1.);
      yprvn (a,b,g,nn,6,0);
      // k2=13;
      for (k=0;k<=6;k++)
        { k1=2*k;
          *(t+k1)=*(g+k+7);
          *(e+k1)=*(g+k); }
      jq=6;
      x=3;
      r1=r2=0;
      for (i=0;i<=jq;i++)
        { r1=r1*x*x+(*(g+jq-i));
          r2=*(g+jq+jq+1-i)+r2*x*x; }
      printf("\n\t bsnd: Jn(%7.2f) =%10.4g;
Jnp=%10.4g\n",x,jn(0,2*x),r1/r2);

      bnsd (hg,g,6);

      printf("\n\t bsnd:  ¯¯à. Js%u(18,x)=\n\n",n);
      for (i=0;i<18;i++) printf("%12.5g%c",*(hg+i), (i%6==5) ? '\n' : ' ');
PRN;
      k2=2*nl;

      for (i=0;i<nl;i++) *(b+i)=*(hl+nl+i);
      jp=sppdp (c,t,hl,13,5);
      sppdp (d,e,b,12,5);
      jq=6;                        //8
      yprvn (c,d,b,jp,jq,0);

      x=2;
      r1=r2=0;
      for (i=0;i<=jq;i++)
```

```
        { r1=r1*x+(*(b+jq-i));
          r2=*(b+jq+jq+1-i)+r2*x;
        }
      printf("\n\t bsnd:  Yn(%4.2f)=Jn(x)*ln (x) =%10.4g; Ynp=%10.4g;
hz(%u,x):\n",x,jn(0,2*x)*log (2*x),r1/r2,jq/2);
      bnsd (hz,b,jq);

      printf("\n\t Z%u(x)=Yn(x)+[Y2(x^2)-gn*Jn(x^2)]=hz+hy-gn*hg\n",n);
      k2=3*jq;

      printf("\n\t Yn(x)=\n");
      for (i=0;i<k2;i++) printf("%12.5g%c",*(hz+i), (i%6==5) ? '\n' : ' ');
PRN;

      jp=bs22 (c,n);
      bnsd (hy,c,jp);

      printf("\n\t  hy(x)\n");
      for (i=0;i<18;i++) printf("%12.5g%c",*(hy+i), (i%6==5) ? '\n' : ' ');
PRN;

      printf("\n\t hg(x)=Jn(x)\n");
      for (i=0;i<18;i++) printf("%12.5g%c",*(hg+i), (i%6==5) ? '\n' : ' ');
PRN;

    /*
      fcl = fopen("bsdjn.dat","w");
      for (i=0; i<18; i++) fprintf(fcl,"%g,",*(hg+i));
      fclose(fcl);

      fcl = fopen("bsdy2.dat","w");
      for (i=0; i<14; i++) fprintf(fcl,"%g,",*(hy+i));
      fclose(fcl);

      fcl = fopen("bsdyn.dat","w");
      for (i=0; i<21; i++) fprintf(fcl,"%g,",*(hz+i));
       fclose(fcl);
      */

      free(e); free(d);
      free(c); free(t);
      free(b); free(g);
      free(a);
      return(1);
    }
   else
    { printf("\n\t no memory for a, t or b in bsnd\n");
      return(-1); }
 }

/*------------------------- bsnd.c -------------------------*/

  int yprbzm (long double *, long double *, long double *, int, int);

 long double  intbzm (long double *hz, long double *hl, long double *hb,
long double *s, long double rb, long double ra, int *nb, int *ns, int mi)
```

```
  {
     /* cg=int (ra,rb: P(k=1:mi)Z_(nb_k)(*s_k*r)dr
        mm - number of functions;
        ns(mi)= s_{nb_0,ns_0},s_{nb_1,ns_1},s_{nb_2,ns_2},   */
    int i, k, m, l, k1, k2, k3, kl, km, ms, mv=6, na, nc, ny;
    long double *zb, *zbm, *a, *b, *c, *sr, cj, s1;
    sr=(long double *)calloc(5,sizeof(long double));
    zb=(long double *)calloc(120,sizeof(long double));
    zbm=(long double *)calloc(120,sizeof(long double));
    a=(long double *)calloc(10,sizeof(long double));
    b=(long double *)calloc(10,sizeof(long double));
    c=(long double *)calloc(20,sizeof(long double));
    if (a!=NULL && b!=NULL  && c!=NULL && sr!=NULL && zb!=NULL  && zbm!=NULL
)
     {
    /*   printf("\n\t intbsm: nb_0=%u, nb_1=%u\t ns_0=%u,  ns_1=%u;
mi=%u\n",*nb, *(nb+1),*ns,*(ns+1),mi);
       printf("\n\t intbsm: Yn\n");
           for (i=0;i<21;i++) printf("%10.4g%c",*(hz+i), (i%7==6) ? '\n' : '
');
       printf("\n\t intbsm: hl\n");
           for (i=0;i<18;i++) printf("%10.4g%c",*(hl+i), (i%6==5) ? '\n' : '
');
       printf("\n\t intbsm: hb\n");
           for (i=0;i<18;i++) printf("%10.4g%c",*(hb+i), (i%6==5) ? '\n' : '
');
     */
       // ms=mm*mv;
       printf("\n\t intbsm: s-values for bes2 (r_0=%7.2f,r_1=%7.2f):  mv=%u;
mi=%u\n",ra,rb,mv,mi);
       for (i=0;i<mv;i++) printf("%10.4g%c",*(s+i), (i%mv==mv-1) ? '\n' : '
');
       k1=*nb*mv+(*ns);
       s1=*(s+k1);
       *sr=s1;
       k2=*(nb+1)*mv+(*(ns+1));
       *(sr+1)=*(s+k2);
       k3=*(nb+2)*mv+(*(ns+2));
       *(sr+2)=*(s+k3);
    /*
       for (m=0;m<mi;m++)
         { s1=*(sr+m);
           bszm (zb, hz, hb, hl, s1, ra);
           k1=42*m;
           for (i=0;i<42;i++) *(zbm+k1+i)=*(zb+i);
         }
       printf("\n\t intbsm: Zn(%8.3g,%u)=\n",s1,mi);
       for (i=0;i<84;i++) printf("%12.4g%c",*(zbm+i), (i%7==6) ? '\n' : '
');
       cj=0;
       for (m=0;m<1;m++)
         { km=7*m;
           k1=km+7;
           if ( m<3 )
            { for (i=0;i<7;i++) *(a+i)=*(zbm+km+i);
              na=7; }
```

```
            else
             { for (i=0;i<6;i++) *(a+i)=*(zbm+km+i);
               na=6; }
            for (k=0;k<1;k++)
             { kl=7*k+42;
               k1=kl+7;
               if ( k<3 )
                { for (i=0;i<7;i++) *(b+i)=*(zbm+kl+i);
                  nc=7; }
               else
                { for (i=0;i<6;i++) *(b+i)=*(zbm+kl+i);
                  nc=6; }
              printf("\n\t intbsm: Zny(%u,%u)=\n",m,k);
              ny=yprbzm (c,a,b, na, nc);
              cj+=inbsz (c,rb,ra,ny,1);
             }
          }
       printf("\n\t intbzm: sr_0(%u)=%7.3f,  sr_1(%u)=%7.3f,
sr_2(%u)=%7.3f: cj=%12.5g\n",k1,s1,k2,*(sr+1),k3,*(sr+2),cj);

       if ( mi==1 ) cj=inzbsn_1 (zb,s1,rb,ra,*nb);
       if ( mi==2 ) cj=inzbsn_2 (zb,sr,rb,ra,nb);
       if ( mi==3 ) cj=inzbsn_3 (zb,sr,rb,ra,nb);
       printf("\n\t  intbzm(%u)=%10.5g\n",mi,cj);
     */
       free(c); free(b);
       free(a); free(zbm);
       free(zb); free(sr);
     }
    else
     { printf("\n\t no memory for b in indsm\n");
       return(0); }
  }


 int sppd (long double *, long double *, long double *, int, int);

 int yprbzm (long double *c, long double *a, long double *b, int m, int l)
  {
   int i, k, nb, na, ny, jq, k2, lp;
   long double *c1, *c2, *d1, *d2, *d3, *d4, r1, r2;
   c1=(long double *)calloc(20,sizeof(long double));
   c2=(long double *)calloc(20,sizeof(long double));
   d1=(long double *)calloc(20,sizeof(long double));
   d2=(long double *)calloc(20,sizeof(long double));
   d3=(long double *)calloc(20,sizeof(long double));
   d4=(long double *)calloc(20,sizeof(long double));
   if ( c1!=NULL && c2!=NULL && d1!=NULL && d2!=NULL && d3!=NULL && d4!=NULL
)
    {
      printf("\n\t yprbzm: input: a(%u), b(%u)\n",m,l);
      for (i=0;i<m;i++) printf("%12.4g",*(a+i)); PRN;
      for (i=0;i<l;i++) printf("%12.4g",*(b+i)); PRN;
       /*
         for (i=0;i<3;i++) *(d1+i+i)=*(a+i);
         for (i=0;i<4;i++) *(d2+i)=*(a+i+3);
```

```
          yprvn (d1,d2,c1,5,2,1);  */
        if ( m==7 )
         {
          for (i=0;i<3;i++) *(d1+i+i)=*(a+i);
          for (i=0;i<4;i++) *(d2+i)=*(a+i+3);
          na=5; }
        else
         { for (i=0;i<3;i++) *(d1+i)=*(a+i);
          for (i=0;i<3;i++) *(d2+i)=*(a+i+3);
          na=3; }
        if ( l==7 )
         {
          for (i=0;i<3;i++) *(d3+i+i)=*(b+i);
          for (i=0;i<4;i++) *(d4+i)=*(b+i+3);
          nb=5; }
        else
         { for (i=0;i<3;i++) *(d3+i)=*(b+i);
          for (i=0;i<3;i++) *(d4+i)=*(b+i+3);
          nb=3; }
        jq=6;
        ny=sppd (c1,d1,d3,na,nb);
        sppd (c2,d2,d4,na,nb);
      /*  printf("\n\t yprbzm: input for yprvd: c2(%u), c1(%u)\n",ny,ny);
        for (i=0;i<ny;i++) printf("%12.4g",*(c2+i)); PRN;
        for (i=0;i<ny;i++) printf("%12.4g",*(c1+i)); PRN; */
        if (na==5 || nb==5 )
         { jq=6; }
        else jq=4;
        if (jq==6 )
         {  yprvn (c1,c2,d1,ny,jq,0);
          bnsd (c,d1,jq);
         }
        else
         {
          *c1=*(a+3)*(*(b+3));
          *(c1+2)=*(a+4)*(*(b+4));
          *(c1+1)=*(a+4)*(*(b+3))+(*(a+3))*(*(b+4));
          *(c1+3)=0;
          lp=0;
          for (i=0;i<3;i++)
            { *(c+i)=*(a+i);
              *(c+6+i)=*(b+i); }
          if ( fabs(*a-(*b))<0.01 ) lp=1;   // && fabs(*(a+1)-(*(b+1)))>0.01
)
          for (i=0;i<20;i++) *(d4+i)=0;
          if (lp==0 )
           { for (k=0;k<3;k++)
               { k2=5*k;
                 *(d4+k2)=*(b+k);
                 *(d4+k2+6)=*(b+k);
                 *(d4+k2+2)=*(a+k);
                 *(d4+k2+8)=*(a+k);
                 *(d4+k2+4)=*(c1+k); }
             xalsd (d3,d4,4,0);
             *(c+3)=*d3;
             *(c+4)=*(d3+1);
```

```
            *(c+9)=*(d3+2);
            *(c+10)=*(d3+3);
            *(c+5)=0;
            *(c+11)=0;
           }
          else
           { *(c+3)=*(c1+2)/(*(a+2));
            *(c+4)=*(c+5)=0;
            *(c+9)=*c1-(*a)*(*(c+3));
            *(c+10)=*(c1+1)-(*(a+1))*(*(c+3));
            *(c+11)=1;
           }

        }
      /*
       k2=3*jq;
       for (i=0;i<k2;i++) printf("%12.5g%c",*(c+i), (i%6==5) ? '\n' : ' ');
PRN;
      */
       free(d4); free(d3);
       free(d2); free(d1);
       free(c2); free(c1);
       return(jq/2);
     }
    else
     { printf("\n no memory for d, c, b or a in yprbsm\n");
       return(-1); }
 }


 long double inzbsn_1 (long double *c, long double sr, long double rr, long
double x, int n)
 {
   /*  Z_n(sr*x)=Y_n(sr*a)*j_n(sr*x)-j_n(sr*a)*Y_n(sr*x);
      =sum_{k=1,3}[u^k_3+u^k_4*x]/[u^k_0+u^k_1*x+u^k_2*x^2]*x^[n].  */

   int i, k, m, m6;
   long double r0, r1, cj1, *b, *h, *r, *u, rm;
   b=(long double *)calloc(20,sizeof(long double));
   u=(long double *)calloc(20,sizeof(long double));
   h=(long double *)calloc(10,sizeof(long double));
   r=(long double *)calloc(10,sizeof(long double));
   if ( b!=NULL && u!=NULL && h!=NULL && r!=NULL )
    {
      rm=bsd2 (u,c,sr,rr,n);
      printf("\n\t inzbsn_1: int{sum[k=0;%u]x^k*Z_%u(%9.4g*r)}=\n",n,n,sr);
      for (i=0;i<18; i++) printf("%10.4g%c",*(u+i), (i%6==5) ? '\n' : ' ');
PRN;
      for (i=0;i<10;i++) *(r+i)=0.;
      for (m=0;m<3;m++)
        { m6=6*m;
          for (i=0;i<6;i++) *(b+i)=*(u+m6+i);
          *(h+n)=*(b+4)/(*(b+2));
          if ( n > 0 )
           { *(h+n-1)=(*(b+3)-(*(b+1))*(*(h+n)))/(*(b+2));
             if ( n > 1 )
```

```
                for (k=0;k<n-1;k++)
                    *(h+n-2-k)=-(*b*(*(h+n-k))+(*(b+1))*(*(h+n-1-
k)))/(*(b+2));
           }
          *(u+3+m6)=-(*b)*(*h);
          if (n==0 ) *(u+4+m6)=*(b+3)-(*(b+1))*(*h);
          else *(u+4+m6)=-(*b)*(*(h+1))-(*(b+1))*(*h);
          for (i=0;i<=n;i++)
            { *(r+i)+=*(h+i);
              *(h+i)=0.; }
        }
      r0=r1=0.;
      for (i=0;i<=n;i++) r0=rr*r0+(*(r+n-i))/(n+1-i);
      r0*=rr;
      for (i=0;i<=n;i++) r1=x*r1+(*(r+n-i))/(n+1-i);
      r0-=r1*x;
      printf("\n\t inzbsn_1: output: r1=%8.3g, cj(%u)=%10.4g;  s_0/2=%7.3f;
array u, r\n",r1*x,n,r0,sr/2);
      for (i=0;i<=n;i++) printf("%10.4g",*(r+i)); PRN;
      for (i=0;i<18; i++) printf("%11.6g%c",*(u+i), (i%6==5) ? '\n' : ' ');
PRN;
      cj1=inbsz (u,rr,x,3,2)+r0;
      printf("\n  inzbsn_1:
int(Z_%u(%7.3f*r)(%7.3f;%7.3f)==%10.5g.\n",n,sr,x,rr,cj1);
      cj1*=rm;
      printf("\n  inzbsn_1:
int(Z_%u(%7.3f*r)(%7.3f;%7.3f)==%10.5g.\n",n,sr,x,rr,cj1);
      free(r);
      free(h); free(u);
      free(b);
      return(cj1);

    }
   else
    { printf("\n no memory for d, c, b or a in jijk_bs\n");
      return(-1); }
 }

long double bsd2 (long double *f, long double *a, long double sr, long
double rr, int n)
 {
  /* Y_n(s*b)*jn(n,s*x)-J_n(s*b)*Yn(n,s*x)=(f_3+f_4x)/(f_0+f_1x+f_2x^2). */

   int k, i, m, k2, ns;
   long double s1, s2, sp, *c, *d, *e, sy, sj, s, rm;
   c=(long double *)calloc(20,sizeof(long double));
   d=(long double *)calloc(20,sizeof(long double));
   e=(long double *)calloc(20,sizeof(long double));
   if ( c!=NULL && d!=NULL && e!=NULL )
    {
      s1=sr/2.;
      sp=1.;
      if ( n > 0 ) for (i=0;i<n;i++) sp*=s1;
      printf("\n\t bsd2: sr=%9.4g,  sp(%u)=%9.4g\n",sr,n,sp);
      for (i=0;i<36;i++) printf("%10.4g%c",*(a+i), (i%6==5) ? '\n' : ' ');
PRN;
```

```
      rm=1.;
      for (m=0;m<3;m++)
        { k2=12*m;
          for (i=0;i<12;i++)  *(c+i)=*(a+k2+i);
          bs12(e,c,d);
         /* if (m==0) rm=*(e+4);
          *(e+3)=*(e+3)/rm;
          *(e+4)=*(e+4)/rm; */
          for (k=0;k<6;k++) *(f+k2+k)=*(e+k);
        }
      rm*=sp;
      printf("\n bsd2: sr=%9.4g;  sp(%u)=%9.4g; rm=%10.5g
f(18):\n",sr,n,sp,rm);
      for (i=0;i<18;i++) printf("%10.4g%c",*(f+i), (i%6==5) ? '\n' : ' ');
PRN;
      free(e);
      free(d); free(c);
      return(rm);
    }
   else
    { printf("\n\t no memory for b in bsd2\n");
      return(-1); }
  }

int bs12 (long double *f, long double *a, long double *b)
 {
   int k, k2;
   long double s1, s2, *c, *d;
   c=(long double *)calloc(10,sizeof(long double));
   d=(long double *)calloc(10,sizeof(long double));
   if ( c!=NULL && d!=NULL )
    {  for (k=0;k<6;k++) *(c+k)=*(d+k)=0.;
       printf("\n\t input: bs12: a, b\n");
       for (k=0;k<6;k++) printf("%9.3g",*(a+k)); PRN;
       for (k=0;k<6;k++) printf("%9.3g",*(b+k)); PRN;
      for (k=0;k<3;k++)
        { k2=2*k;
          *(c+k)=*a*(*(b+k));
          *(d+k2)=*(a+k)*(*(b+3));
          *(d+k2+1)=*(a+k)*(*(b+4)); }
      for (k=0;k<3;k++)
        { *(d+k)+=*(b+k)*(*(a+3));
          *(d+k+2)+=*(b+k)*(*(a+4));
          *(c+k+2)+=*(a+1)*(*(b+k));
          *(c+k+4)+=*(a+2)*(*(b+k)); }
      yprvd (c,d,a,7,2,0);
      s1=*(a+5);
      for (k=0;k<6;k++) *(a+k)/=s1;
      for (k=0;k<3;k++)
        { *(f+3+k)=*(a+k);
          *(f+k)=*(a+k+3); }
      printf("\n bs12: output f=");
      for (k=0;k<6;k++) printf("%9.3g",*(f+k));
      free(d); free(c);
      return(1);
    }
```

```
  else
   { printf("\n\t no memory for b in indsm\n");
     return(-1); }
 }

int bs22 (long double *b, int mm)
 {
   /*  2-nd part for Y_mm    */
   int i, k, mp, np;
   long double *a, *d, r1, rn, pn, zn, cn, bn, rp;
   a=(long double *)calloc(30,sizeof(long double));
   d=(long double *)calloc(30,sizeof(long double));
   if ( a!=NULL && d!=NULL )
    {
      rn=1.;
      cn=0.;
      *a=1.;
      for (i=1;i<15;i++) *(a+i)=0.;
      if ( mm > 0 )
         for (k=1;k<=mm;k++)
           { rn*=k*1.;
             cn+=1./k; }
      bn=0.;
      pn=1.;
      zn=1.;
      if ( mm==0 )
       { *b=1.;
         for (k=1;k<14;k++)
           { pn*=k*1.;
             bn+=1./k;
             *(b+k-1)=(bn)*zn/(pn*pn);
             zn=-zn; }
       }
      else
       { for (k=1;k<14;k++)
           {
             *(b+k-1)=(cn+bn)*zn/(pn*rn);
             bn+=1./k;
             pn*=k*1.;
             rn*=(k+mm)*1.;
             cn+=1./(k+mm);
             zn=-zn;
           }
       }
      np=6;
      rp=(mm==0) ? 2./pi : -1./pi;
      yprvn (a,b,d,14,np,0);
      rn=0; cn=0;
      for (i=0;i<=np;i++)
        { *(b+i)=*(d+np+1+i);
          rn+=*(b+i);
          cn+=*(d+i)*rp;
        }
       bn=cn/rn;
      for (i=0;i<=np;i++) *(b+i+np+1)=*(d+i)*rp;
      for (i=0;i<=np;i++) *(b+i)=*(d+i)*rp;
```

```
      for (i=0;i<=np;i++) *(b+i+np+1)=*(d+i+np+1);
    /*  printf("\n bs22: bn=%10.4g: b_%u/b_0(%u)=\n",bn,np+1,mm);
      for (i=0;i<=np;i++) printf("%11.4g",*(b+i)); PRN;
      for (i=0;i<=np;i++) printf("%11.4g",*(b+np+1+i)); PRN;
    */
      free(d); free(a);
      return(np);
     }
    else
     { printf("\n\t no memory for a or d in bs22\n");
      return(-1); }
  }

void slfr (long double *z, long double *h, int m, int n)
 {
   /* h=[1, h, R, x0]; m - order of Bessel; n - number of self-values.
      z[m,n] -- array of self-values
      djm/dr+h*jm=0.               */
    /* output: slfr1.dat (m,n); slfr0.dat (m,n)(h=0); nrmr1.dat (m,n);
nrmr0.dat (m,n).  */
   FILE *fcl;
   register int i, k, ki, km;
   long double x, dx, x1, bm, bm1, bm2, r1, r2, hr;
   hr=*(h+1)*(*(h+2));
   km=m*n;
   for (k=0; k<m; k++)
     { ki=k*n;
       if (fabs(*(h+1) >.001 )) x1=*(h+3)+k;
       else x1=*(h+3)+k*pi/2.;
       for (i=0; i<n; i++)
         { x=x1;
           dx=1.0;
           while ( fabs(dx) > .005 )
            { bm=jn(k,x);
              bm1=jn(k+1,x);
              if (fabs(*(h+1) >.001 ))
               { r1=(k+hr)*bm-x*bm1;
                 r2=(k*(k+hr)/x-x)*bm-hr*bm1; }
              else
               { r1=bm;
                 r2=k/x*bm-bm1; }
              dx=r1/r2;
              x-=dx;
            }
           *(z+i+ki)=x/(*(h+2));
           x1=pi+x;
           if (fabs(*(h+1) >.001 ))
            { bm=jn(k,x);
              r2=*(h+2)*(*(h+2));
              *(z+i+ki+km)=r2/(2*x*x)*(hr*hr+x*x-k*k)*bm*bm; }
           else
            { bm=jn(k+1,x);
              *(z+i+ki+km)=bm*bm/2; }
         }
     }
```

```
   printf("\n\t self-vlues for bessel(h=%7.2f,R=%7.2f): m=%u,
n=%u\n",*(h+1),*(h+2),m,n);
   for (i=0;i<km;i++) printf("%9.3g%c",*(z+i), (i%n==n-1) ? '\n' : ' ');
   printf("\n\tnorms for bessel: m=%u, n=%u\n",m,n);
   for (i=0;i<km;i++) printf("%9.3g%c",*(z+i+km), (i%n==n-1) ? '\n' : ' ');
   if (fabs(*(h+1) >.001 ))
    { fcl = fopen("slfr1.dat","w");
      for (i=0; i<km; i++) fprintf(fcl,"%g,",*(z+i));
      fclose(fcl);
      fcl = fopen("nrmr1.dat","w");
      for (i=0; i<km; i++) fprintf(fcl,"%g,",*(z+i+km));
      fclose(fcl); }
   else
    { fcl = fopen("slfr0.dat","w");
      for (i=0; i<km; i++) fprintf(fcl,"%g,",*(z+i));
      fclose(fcl);
      fcl = fopen("nrmr0.dat","w");
      for (i=0; i<km; i++) fprintf(fcl,"%g,",*(z+i+km));
      fclose(fcl); }
 }

void slfz (long double *z, long double *h, int n)
 {
   /* h=[al0, bl0, al1, bl1, Lz, x0]; n - number of self-values.
      z[n] -- array of self-values
      [al0*dZ/dz+bl0*Z]_(z=0)=0; [al1*dZ/dz+bl1*Z]_(z=Lz)=0;
      f(x)=d1*x*sin(x)+(d0*x^2-L_z*bt^1)*cos(x)=0.    x=al_z*L_z.  */
     /* output - slfz.dat: alz_n; int Z_n(z)dz; ||Z_n(h)||^2.    */
   register int i, n1;
   long double x, dx, x1, bm, bm1, r1, r2, r3, r4, c0, d1, d0;
   FILE *fcl;
   n1=3*n;
   x1=*(h+5);
   c0=*h/(*(h+1));
   d0=c0*(*(h+2))/(*(h+4));
   d1=*(h+2)+(*(h+3))*c0;
   for (i=0; i<n; i++)
     { x=x1;
       dx=1.0;
       while ( fabs(dx) > .001 )
        { bm=cos(x);
          bm1=sin(x);
          r1=(d0*x*x-(*(h+3))*(*(h+4)))*bm+d1*x*bm1;
          r2=(2*d0+d1)*x*bm+(d1-d0*x*x+(*(h+3))*(*(h+4)))*bm1;
          dx=r1/r2;
          x-=dx;
        }
       *(z+i)=x/(*(h+4));
       x1=pi+x;
       r3=*h/(*(h+1))*(*(z+i));
       r2=2*r3*(cos(2*x)-1);
       r4=(1-r3*r3)*sin(2*x);
       *(z+i+n+n)=((1+r3*r3)*2*x+r2+r4)/(4*(*(z+i)));
       *(z+i+n)=(sin(x)/(*(z+i))+*h/(*(h+1))*(cos(x)-1))/(*(z+i+n+n));
     }
   printf("\tself-vlues for z: n=%u\n",n);
```

```
   for (i=0;i<n;i++) printf("%9.3g",*(z+i)); PRN;
   printf("\t transorms for z:  n=%u\n",n);
   for (i=0;i<n;i++) printf("%9.3g",*(z+i+n)); PRN;
   printf("\tnorms for z:  n=%u\n",n);
   for (i=0;i<n;i++) printf("%9.3g",*(z+i+n+n)); PRN;
   fcl = fopen("slfz.dat","w");
   for (i=0; i<n1; i++) fprintf(fcl,"%g,",*(z+i));
   fclose(fcl);
 }

/*------------------------ slfr.c -------------------------*/

long double inbsz (long double *a, long double rb, long double ra, int np,
int lx)
 {
   /*  C= int[ra,rb] Z_m(sr*x)xdx, k=1,np.
       Z_m(sr*x)=Y_m(sr*b)*j_m(sr*x)-j_m(sr*b)*Y_m(sr*x);
       np -- number of chains;  2*ns -- sum of order of J.
       lx=1; 2 order;   lx=2; 4 order of chains - x=r^2.
   */
   int i, k, k2;
   long double r1, xr0, xr1, d, a1, a2, a3, a4, at, cc, c1, *b, rc, rd;
   b=(long double *)calloc(40,sizeof( long double));
   if ( b!=NULL )
    { k2=6*np;
      printf("\n\t inbsz: input: np=%u  - array a(%u)\n",np,k2);
      for (k=0;k<k2;k++)  printf("%10.3g%c",*(a+k), (k%6==5) ? '\n' : ' ');
      if ( lx==1 )
       { rc=ra; rd=rb; }
      if ( lx==2 )
       { rc=ra*ra; rd=rb*rb; }

      cc=0.;
      for (k=0;k<np;k++)
        { k2=6*k;
          for (i=0;i<6;i++) *(b+i)=*(a+i+k2);
          r1=(*b)*(*(b+2))-(*(b+1))*(*(b+1))/4;
          d=sqrt(fabs(r1));
          xr1=(*(b+2)*rd+(*(b+1)))*rd+(*b);
          xr0=(*(b+2)*rc+(*(b+1)))*rc+(*b);
       if ( r1 > 0 )
            at=(atan((rd*(*(b+2))+(*(b+1))/2)/d)-
atan((rc*(*(b+2))+(*(b+1))/2)/d))/d;
          else
           { a1=(*(b+2)*rd+(*(b+1))/2-d)/(*(b+2)*rd+(*(b+1))/2+d);
             a2=(*(b+2)*rc+(*(b+1))/2-d)/(*(b+2)*rc+(*(b+1))/2+d);
             at=log(fabs(a1/a2))/(2*d); }
          a3=(*(b+3)-(*(b+1))*(*(b+4)))/2;
          a4=*(b+1)/2*(*(b+4)*(*(b+1))-(*(b+3)))-(*b)*(*(b+4));
          if ( lx==1 )
             cc+=*(b+4)/(*(b+2))*(rd-rc);
          if ( *(b+5) == 0 )
           { c1=a3*log(fabs(xr1/xr0))+a4*at;
            cc+=c1;
           }
          else
```

```
           {
             if ( *(b+5)==1. )
              { c1=((rb*a3+a4)/xr1-(ra*a3+a4)/xr0+a3*at)/r1;
            cc+=c1;
              }
             else
              { a1=((a3*rb+a4)/(xr1*xr1)-(a3*ra+a4)/(xr0*xr0))/2;
                a2=3*a3/r1*((rb+(*b)/2.)/xr1-(ra+(*b)/2.)/xr0+at);
                c1=(a1+a2)/r1;
                cc+=c1;
                printf("\n\tk=%u b_5=%8.2g: a1=%8.3g; a2=%8.3g;  c1=%8.3g
c=%8.3g\n",k,*(b+5),a1,a2,c1,cc);
              }
           }
        }
      printf("\n\t p/p inbsm: output: np=%u; int-l=%11.5g\n",np,cc);
      free(b);
      return(cc);
    }
   else
    { printf("\n\t no memory for b in indsm\n");
      return(-1); }
 }


int bnsd (long double *g, long double *d, int nn)
 {
  /*       sum[0,nn]d[n,k+nn+1]*z^k       g[n,6k+3] + g[n,6k+4]*z
     Ln(x)=_____________________ =_____________________________________;
k=0,mj;
            sum[0,nn]d[n,k]*z^k   g[n,6k] + g[n,6k+1]*z+g[n,6k+2]*z^2
     nn -- order of Fn(x).  mj=nn/2.   ¥§ã«ìâ â -- ä ©« pnapr.dat
*/

   int i, k, l, n, mj, ki, nk, n2, n6;
   long double *a, *b, *c, *ay, *t, r1, r2, r3, r4, zn;
   mj=(nn+1)/2;
   c=(long double *)calloc(100,sizeof(long double));
   b=(long double *)calloc(30,sizeof(long double));
   t=(long double *)calloc(30,sizeof(long double));
   a=(long double *)calloc(30,sizeof(long double));
   ay=(long double *)calloc(20,sizeof(long double));
   if ( a!=NULL && b!=NULL && ay!=NULL && c!=NULL && t!=NULL )
    {
      n2=2*nn;
  /*    printf("\n\t bnsd(%u): \n",nn);
        for (i=0;i<=nn;i++) printf("%9.3g",*(d+i));  PRN;
        for (i=0;i<=nn;i++) printf("%9.3g",*(d+i+nn+1));  PRN;
    */
      r1=*(d+n2+1);
      for (i=0;i<=nn;i++)
        { ki=2*i;
          *(b+i)=*(d+i)/r1;
          *(a+i)=*(d+n2+1-i)/r1; }
    /*  printf("\n\t bnsd(%u): r1=%10.4g \n",nn,r1);
        for (i=0;i<=nn;i++) printf("%9.3g",*(b+i));  PRN;
```

```
    for (i=0;i<=nn;i++) printf("%9.3g",*(a+i));  PRN;
  */
   gornd(a,ay,t,nn);

   for (k=0;k<nn;k++)
     { nk=(nn+1)*k+nn;
       *(c+nk)=*(b+k); }

   if ( mj > 2 )
    { for (n=0;n<mj;n++)
        { ki=0;
          for (k=0;k<mj; k++)
            { if ( k!=n )
               { nk=3*k;
                 for (i=0;i<3;i++)
                   { *(ay+ki)=*(t+nk+i);
                     ki+=1; }
               }
            }
       //   for (i=0;i<3*(mj-1);i++) printf("%9.3g",*(ay+i));  PRN;
          svpd (b,ay,3,mj-1);
        //  for (i=0;i<nn-1;i++) printf("%9.3g",*(b+i));  PRN;
          for (k=0;k<nn-1;k++)
            { nk=(nn+1)*k+2*n;
              *(c+nk)=*(b+k);
              *(c+nk+nn+2)=*(b+k); }
        }
    }
   else
    { for (k=0;k<3;k++)
        { nk=(nn+1)*k;
          *(c+nk)=*(t+k+3);
          *(c+nk+nn+2)=*(t+k+3);
          *(c+nk+2)=*(t+k);
          *(c+nk+nn+4)=*(t+k); }
    }
   xalsd (b,c,nn,0);
   for (n=0;n<mj;n++)
     { n6=6*n;
       n2=2*n;
       nk=3*n;
       *(g+n6+3)=*(b+n2);
       *(g+n6+4)=*(b+n2+1);
       *(g+n6+5)=0.;
       for (k=0;k<3;k++) *(g+n6+k)=*(t+nk+k);
     }
   n2=6*mj;
 //  printf("\n\t p/p bnsd:  Š®íää¨æ¨¥-âë à §«®¦¥-¨ï Fns(%u,x)=\n\n",ki);
 //  for (k=0;k<n2;k++) printf("%9.3g%c",*(g+k), (k%6==5) ? '\n' : ' ');
   free(ay); free(a);
   free(t); free(b);
   free(c);
   return(mj);
 }
else
 { printf("\n\t no memory for ay, a, t or b in lnsd\n");
```

```
      return(-1); }
 }

int lnsd (long double *h, long double x, int nl)
 {
  /*                 g[n,6k+3] + g[n,6k+4]*x
     Ln(x)=-----------------------------------------------;k=0,mj;
           g[n,6k] + g[n,6k+1]*x + g[n,6k+2]*x^2
     jq -- order of Ln(x).  mj=jq/2.
      ¥§ã«ìâ â -- ä ©« lnapr.dat  */

   FILE *fcl;
   int i, k, k2, n2, mj, np;
   long double *c, *b, *t, *d, cg, r1, r2, r3, r4, r5, rln;
  // mj=(jq+1)/2;
   b=(long double *)calloc(20,sizeof(long double));
   d=(long double *)calloc(20,sizeof(long double));
   t=(long double *)calloc(20,sizeof(long double));
   c=(long double *)calloc(20,sizeof(long double));
   if ( d!=NULL && c!=NULL && b!=NULL && t!=NULL )
    {
      cg=0.5772157;
      for (k=0;k<20;k++) *(b+k)=*(c+k)=*(d+k)=0.;
      *t=*(t+2)=1.;
      *(t+1)=2.;
      *b=*(b+2)=1./(2.*nl+1);
      *(b+1)=-2.*(*b);
      for (k=0;k<nl-1;k++)
        { k2=2*k+2;
          r1=2.*(nl-k)-1.;
          for (i=0;i<=k2;i++) *(d+i)=*(b+i)+(*(t+i))/r1;
          *b=*d;
          *(b+1)=*(d+1)-2.*(*d);
          *c=*t;
          *(c+1)=*(t+1)+2.*(*t);
          for (i=0;i<=k2+1;i++)
            { *(b+2+i)=*(d+i)-2.*(*(d+i+1))+(*(d+i+2));
              *(c+2+i)=*(t+i)+2.*(*(t+i+1))+(*(t+i+2)); }
          *(b+k2+2)=*(d+k2);
          *(c+k2+2)=*(t+k2);
          for (i=0;i<=(k2+2);i++) *(t+i)=*(c+i);
        }
      n2=2*nl;
      for (k=0;k<=n2;k++) *(b+k)=*(b+k)+(*(t+k));
      *d=-(*b);  /* (x-1)/(x+1)*[b/c]  */
      *c=*t;
      for (k=0;k<n2;k++)
        { *(d+k+1)=*(b+k)-(*(b+k+1));
          *(c+k+1)=*(t+k+1)+(*(t+k)); }
      *(d+n2+1)=*(b+n2);
      *(c+n2+1)=*(t+n2);
      for (k=0;k<=n2+1;k++)
        { *(h+k)=*(c+k);
          *(h+k+n2+2)=(*(d+k)*2.+cg*(*(c+k)))*2./pi; }
         // *(d+k+n2+2); }
      k2=2*n2+3;
```

```
      np=(k2+1)/2;
      printf("\n\t p/p lnsd: np=%u:  h(%u,%u) & h(0,%u)\n",np,n2+2,k2,np-1);
      for (i=0;i<=n2+1;i++) printf("%9.4g%c",*(h+n2+2+i), (i%9==8) ? '\n' :
' '); PRN;
      for (i=0;i<=n2+1;i++) printf("%9.4g%c",*(c+i), (i%9==8) ? '\n' : ' ');
PRN;
      for (k=0,r1=0,r2=0;k<=n2+1;k++)
        { r1=*(h+k2-k)+x/2*r1;
          r2=*(h+n2+1-k)+x/2*r2; }
      rln=r1/r2;
      r1=(x/2-1)/(x/2+1);
      r2=r1*r1;
      for (k=0,r3=0;k<=6;k++) r3=1./(2.*(6-k)+1)+r3*r2;
      r3*=2.*r1;

      printf("\n\t np=%u;\t log(%6.2f)=%10.5g; lnap(%6.2f)=%10.5g;
ln(%6.2f)=%10.5g;\n",np,x/2,(cg+r3)*2/pi,x/2,rln,x/2,(cg+log(x/2))*2/pi);
      for (k=0;k<n2+2;k++)
        { *(t+k)=*(h+k);
          *(c+k)=*(h+n2+2+k); }
      /* yprvd (t,c,b,n2+2,6,1);
      fcl = fopen("lnap.dat","w");
      for (i=0; i<=k2; i++) fprintf(fcl,"%g,",*(h+i));
      fclose(fcl);
      */
      free(c); free(t);
      free(d); free(b);
      return(np);
    }
   else
    { printf("\n\t no memory for ay, a, t or b in lnsd\n");
      return(-1); }
 }

/*_____________________________ bsnd.c ___________________________________*/


long double inzbsn_2 (long double *cc, long double *sr, long double rr, long
double x, int *nb)
 {
   /* *nb -- order of Jn(sr*x);   Z_m(sr*x)=Y_m(sr*b)*j_m(sr*x)-
j_m(sr*b)*Y_m(sr*x);
      Z_m0(sr0*x)*Z_m1(sr1*x)= Z_m0(sr0*x)+Z_m1(sr1*x)=
      =sum_{k=1,6}[u^k_3+u^k_4*x]/[u^k_0+u^k_1*x+u^k_2*x^2]*x^[m0+m1].  */

   register int i, k, m, m2, mp, m6, l5, k2, k6, ns;
   long double r1, r2, *a, *b, *c, *d, *e, *h, *r, *u, r0, cj2, rm, rmp;
   u=(long double *)calloc(40,sizeof(long double));
   a=(long double *)calloc(22,sizeof(long double));
   b=(long double *)calloc(22,sizeof(long double));
   c=(long double *)calloc(22,sizeof(long double));
   d=(long double *)calloc(22,sizeof(long double));
   e=(long double *)calloc(22,sizeof(long double));
   h=(long double *)calloc(22,sizeof(long double));
   r=(long double *)calloc(10,sizeof(long double));
```

```
   if ( u!=NULL && a!=NULL && b!=NULL && c!=NULL && d!=NULL && e!=NULL &&
h!=NULL && r!=NULL )
    {
      ns=(*nb+(*(nb+1)));
      rmp=1.;
      for (m=0;m<2;m++)
        {
          m6=18*m;
          for (k=0;k<18;k++)
             *(b+k)=*(cc+m6+k);

          rm=bsd2 (h,b,*(sr+m),rr,*(nb+m));
          for (k=0;k<18;k++) *(u+k+m6)=*(h+k);
          rmp*=rm;
        }
      printf("\n\t
x^[%u]*Z_%u(s*x)*Z_%u(s*x)=x^[%u]*[Z_%u(s*x)+Z_%u(s*x)]\n",ns,*(nb),*(nb+1),
ns,*(nb),*(nb+1));
      for (i=0;i<10;i++) *(r+i)=0.;
      for (m=0;m<3;m++)
        { m6=6*m;
          for (i=0;i<6;i++) *(b+i)=*(u+m6+i);
          m2=2*m;
          for (i=0;i<22;i++) *(h+i)=0.;

          for (k=0;k<3;k++)
            { k2=2*k;
              k6=6*k+18;
              for (i=0;i<6;i++) *(c+i)=*(u+k6+i);
              l5=pzsd2 (a,h,b,c,nb);
              if ( l5==1 && m==k ) mp=2;
              *(d+m2)+=*a;
              *(d+m2+1)+=*(a+1);
              *(e+m6+k2)=*(a+2);
              *(e+m6+k2+1)=*(a+3);
              if ( ns > 0 ) for (i=0;i<ns;i++) *(r+i)+=*(h+i);
            }
        }
       if ( l5 == 0 ) for (i=0;i<6;i++)
*(d+6+i)=*(e+i)+(*(e+6+i))+(*(e+12+i));
       else
        { for (k=0;k<2;k++)
            { *(d+k)+=*(e+6+k)+(*(e+12+k));
              *(d+8+k)=*(e+4+k)+(*(e+10+k))+(*(d+4+k));
              *(d+k+4)=*(e+k+2)+(*(e+14+k))+(*(d+k+2));
              *(d+2+k)=*(e+k);
              *(d+k+6)=*(e+k+8);
              *(d+k+10)=*(e+k+16); }
        }
      for (m=0;m<6;m++)
        { m2=2*m;
          m6=6*m;
          for (i=0;i<2; i++) *(u+i+3+m6)=*(d+i+m2); }
      if ( mp==2 )
       { for (i=0;i<3;i++)
           { *(u+i+24)=*(u+i+30)=*(u+i+12);
```

```
            *(u+i+12)=*(u+i+18)=*(u+6+i);
            *(u+i+6)=*(u+i); }
          *(u+11)=*(u+23)=*(u+35)=1.;
        }
      r0=r1=0.;
      if ( ns > 0 )
       { for (i=0;i<ns;i++) r0=rr*r0+(*(r+ns-1-i))/(ns-i);
       r0*=rr;
         for (i=0;i<ns;i++) r1=x*r1+(*(r+ns-1-i))/(ns-i);
       r0-=r1*x;
       }
      printf("\n\t inzbsn_2: output: cj=%7.3g;  s_0/2=%7.3f; s_1/2=%7.3f:
array u, r\n",r0,*sr/2,*(sr+1)/2);
      for (i=0;i<36; i++) printf("%10.4g%c",*(u+i), (i%6==5) ? '\n' : ' ');
PRN;
      if ( ns > 0 ) for (i=0;i<ns;i++) printf("%10.4g",*(r+i)); PRN;
      cj2=rmp*inbsz (u,rr,x,6,2)+r0;
      printf("\n\t inzbs_2: int(Z_%u*Z_%u)(%7.2f,%7.2f)=%10.5g;  s_0=%7.3f;
s_1=%7.3f\n",*nb,*(nb+1),x,rr,cj2,*sr,*(sr+1));
      free(r);
      free(h); free(e);
      free(d); free(c);
      free(b); free(a);
      free(u);
      return(cj2);
    }
   else
    { printf("\n no memory for d, c, b or a in inzbs_2\n");
      return(-1); }
 }

/* -------------------------- jijk_bsd.c -------------------------- */

int pzsd2 (long double *e, long double *d, long double *b, long double *c,
int *nb)
 {
  /*  [b_3+b_4*x]/[b_0+b_1*x+b_2*x^2]*[c_3+c_4*x]/[c_0+c_1*x+c_2*x^2]=
      [e_0+e_1*x]/[b_0+b_1*x+b_2*x^2]+[e_2+e_3*x][c_0+c_1*x+c_2*x^2];
      =[e_0+e_1*x]/[b_0+b_1*x+b_2*x^2]+[e_2+e_3*x]/[b_0+b_1*x+b_2*x^2]^2. */

   int k, i, i1, l1, np, nx, ns, mp, mk;
   long double *f, *g, *h, r1;
   f=(long double *)calloc(12,sizeof(long double));
   g=(long double *)calloc(12,sizeof(long double));
   h=(long double *)calloc(12,sizeof(long double));
   if ( f!=NULL && g!=NULL && h!=NULL )
    {
      for (i=0;i<12;i++) *(g+i)=*(e+i)=*(d+i)=0.;
      *f=*(b+3)*(*(c+3));
      *(f+1)=*(b+4)*(*(c+3))+(*(c+4))*(*(b+3));
      *(f+2)=*(b+4)*(*(c+4));
      *(f+3)=0.;
      mp=*nb+(*(nb+1));
      printf("\n\tp/p pzsd2: input: nb=%u: array b, nb_1=%u: b,
c\n",*nb,*(nb+1));
      for (i=0;i<6;i++)  printf("%10.3g",*(b+i));  PRN;
```

```
      for (i=0;i<6;i++)  printf("%10.3g",*(c+i));  PRN;
      for (i=0;i<15;i++) *(d+i)=0.;
      if ( mp==0 )
       { *g=0.;
         for (i=0;i<3;i++) *(g+i+1)=*(f+i); }
      if ( mp > 0 )
       {
         *e=*c*(*b);
         *(e+1)=*c*(*(b+1))+(*(c+1))*(*b);
         *(e+2)=*(c+1)*(*(b+1))+(*(c+2))*(*b)+(*c)*(*(b+2));
         *(e+3)=*(c+2)*(*(b+1))+(*(c+1))*(*(b+2));
         *(e+4)=*(c+2)*(*(b+2));
         *(d+mp-1)=*(f+2)/(*(e+4));
     /* printf("\n\tp/p pzsd2:  mp=%u:  array f, e\n",mp);
      for (i=0;i<6;i++)  printf("%10.3g",*(f+i));  PRN;
      for (i=0;i<6;i++)  printf("%10.3g",*(e+i));  PRN;  */
         if ( mp==1 )
          { *(g+1)=-(*(e+1))*(*d);
            *(g+2)=*f-(*(e+2))*(*d);
            *(g+3)=*(f+1)-(*(e+3))*(*d); }
         if ( mp > 1 )
          { *(d+mp-2)=(*(f+1)-(*(e+3))*(*(d+mp-1)))/(*(e+4));
            if ( mp==2 )
             { *(g+2)=-(*(e+1))*(*(d+1))-(*(e+2))*(*d);
               *(g+3)=*f-(*(e+2))*(*(d+1))-(*(e+3))*(*d); }
            if ( mp > 2)
             { *(d+mp-3)=(*f-(*(e+2))*(*(d+mp-1))-(*(e+3))*(*(d+mp-
2)))/(*(e+4));
               if ( mp > 3)
                  *(d+mp-4)=-(*(e+1)*(*(d+mp-1))+(*(e+2))*(*(d+mp-
2))+(*(e+3))*(*(d+mp-3)))/(*(e+4));
               if ( mp > 4 )
                { mk=mp-4;
                  for (k=1;k<=mk;k++)
                    { for (i=0,r1=0;i<3;i++) r1+=*(e+i)*(*(d+mp-k-i));
                      *(d+mk-k)=-r1/(*(e+4)); }
                }
               *(g+2)=-(*e)*(*(d+2))-(*(e+1))*(*(d+1))-(*(e+2))*(*d);
               *(g+3)=-(*e)*(*(d+3))-(*(e+1))*(*(d+2))-(*(e+2))*(*(d+1))-
(*(e+3))*(*d);
             }
          }
         *g=-(*e)*(*d);
         *(g+1)=-(*e)*(*(d+1))-(*(e+1))*(*d);
       }
      printf("\n\tp/p pzsd2: g(4) & d(%u)=",mp);
      for (i=0;i<4;i++)  printf("%10.3g",*(g+i));  PRN;
      if ( mp > 0 ) for (i=0;i<mp;i++) printf("%10.3g",*(d+i)); PRN;
      l1=psz2 (e,b,c,g);
      free(h);
      free(g); free(f);
      return(l1);
    }
   else
    { printf("\n\tno memory for g, f in pzs2d\n");
      return(-1); }
```

```
}


int pzszab (long double *c, long double *a, long double *b, int lp)
 {
    // Z_a(bt*x) * Z_b (bt*x)=sum c_j;
   int i,k, ki, n;
   long double *d;
   d=(long double *)calloc(60,sizeof(long double));
   if ( d!=NULL )
    {
      n=6;
      for (k=0;k<4;k++)
        { for (i=0;i<3;i++)
            { ki=8*k+7*i;
              *(d+ki)=*(b+i); }
        }
      for (k=0;k<2;k++)
        { for (i=0;i<3;i++)
            { ki=8*k+14*i+4;
              *(d+ki)=*(a+i); }
        }
      *(d+6)=*(a+3)*(*(b+3));
      *(d+13)=*(a+3)*(*(b+4));
      *(d+20)=*(a+4)*(*(b+3));
      *(d+27)=*(a+4)*(*(b+4));
      xalsd (c,d,n,lp);
    /*
      printf("\n\tp/p  pzszab:  d(%u)=",n);
      for (i=0;i<n;i++)  printf("%10.3g",*(c+i));  */
      free(d);
      return(n);
    }
   else
    { printf("\n\tno memory for d in pzs2\n");
      return(-1);
    }
 }

int pzszab (long double *, long double *, long double *, int);
int pzsza (long double *, long double *, long double *, int);

long double intbs2_2 (long double *z, int lp)
 {
  /* ||Z_n (x)||=int^r_1_r_0: Z^2_n(x)dx.*/
   int i, k, ki, m, n, nk, mk, nn, nz;
   long double *d, *a, *b, *c, *e, *g, r1;
   a=(long double *)calloc(10,sizeof(long double));
   b=(long double *)calloc(10,sizeof(long double));
   c=(long double *)calloc(10,sizeof(long double));
   d=(long double *)calloc(60,sizeof(long double));
   e=(long double *)calloc(60,sizeof(long double));
   g=(long double *)calloc(60,sizeof(long double));

   if (a!=NULL && b!=NULL && c!=NULL && d!=NULL && e!=NULL && g!=NULL )
    {
```

```
        for (i=0;i<60;i++) *(d+i)=0.;
        for (nn=0;nn<2;nn++)
          { nz=18*nn;
            for (n=0;n<3;n++)
              { nk=6*n+nz;
                for (i=0;i<6;i++) *(a+i)=*(z+nk+i);
                for (m=n;m<3;m++)
                  { mk=6*m+nz;
                    for (i=0;i<6;i++) *(b+i)=*(z+mk+i);
                    for (i=0;i<10;i++) *(c+i)=0.;
                    pzsza (c,a,b,lp);
                    if ( m==n )
                     { *(e+nk+3)=*(c+1);
                       *(e+nk+4)=*(c+2);
                       r1=*c;
                       for (i=0;i<3;i++) *(e+nk+i)=*(a+i);
                     }
                       // (e_3+e_4*x^2)/ a(x)^2.
                    else
                     {
                       *(d+nk+3)+=2*(*c);
                       *(d+nk+4)+=2*(*(c+1));
                       for (i=0;i<3;i++) *(d+nk+i)=*(a+i);
                       *(d+mk+3)+=2*(*(c+2));
                       *(d+mk+4)+=2*(*(c+3));
                       for (i=0;i<3;i++) *(d+mk+i)=*(b+i);
                   /*    printf("\n\t intbs2_2:n=%u, m=%u;
d(%u)=\n",n,m,nk);
                       for (i=0;i<6;i++) printf("%12.4g",*(d+nk+i));
                       printf("\n\t intbs2_2:n=%u, m=%u;  d(%u)=\n",n,m,mk);
                       for (i=0;i<6;i++) printf("%12.4g",*(d+mk+i));
                    */
                       }
                  }
                *(d+nk+3)+=r1;
              }
            printf("\n\t intbs2_2:  d(%u)=\n",nz);
            for (k=0;k<3;k++)
              { nk=6*k+nz;
                for (i=0;i<6;i++) printf("%12.4g",*(d+nk+i));
                PRN; }
            printf("\n\t intbs2_2:  e(%u)=\n",nz);
            for (i=0;i<18;i++) printf("%12.4g%c",*(e+nz+i), (i%6==5) ? '\n' :
' ');
          }
        nz=18;
        for (n=0;n<3;n++)
          { nk=6*n;
            for (i=0;i<6;i++) *(a+i)=*(z+nk+i);
            for (m=n;m<3;m++)
              { mk=6*m+nz;
                for (i=0;i<6;i++) *(b+i)=*(z+mk+i);
                for (i=0;i<10;i++) *(c+i)=0.;
                pzszab (c,a,b,0);
                *(g+nk+3)+=2*(*c);
                *(g+nk+4)+=2*(*(c+1));
```

```
                *(g+nk+5)+=2*(*(c+2));
                 //   *(d+nk+6)+=2*(*(c+3));
                for (i=0;i<3;i++) *(g+nk+i)=*(a+i);
                *(g+mk+3)+=2*(*(c+4));
                *(g+mk+4)+=2*(*(c+5));
                for (i=0;i<3;i++) *(g+mk+i)=*(b+i);
              /*       printf("\n\t intbs2_2:n=%u, m=%u;  g(%u)=\n",n,m,nk);
                     for (i=0;i<6;i++) printf("%12.4g",*(g+nk+i));
                     printf("\n\t intbs2_2:n=%u, m=%u;  g(%u)=\n",n,m,mk);
                     for (i=0;i<6;i++) printf("%12.4g",*(g+mk+i));    */
             }
          }
        for (k=0;k<3;k++)
          { nk=6*k;
            *(g+nk+21)+=*(d+nk+21);
            *(g+nk+22)+=*(d+nk+22);
            *(g+nk+3)+=*(d+nk+3);
            *(g+nk+5)+=*(d+nk+4); }
       printf("\n intbs2_2: ||Z_n(bt*x)||=int [ra,rb]\sum
kr=1,3][Za_k(x)+Zb_k(x)+Za^2(x)+Zb^2(x)]\n");
       /* printf("\n\t intbs2_2:  g=\n");
        for (k=0;k<3;k++)
          { nk=6*k;
            for (i=0;i<6;i++) printf("%12.4g",*(g+nk+i));
            PRN; }  */
        printf("\n\t intbs2_2:  g=\n");
        for (i=0;i<36;i++) printf("%12.4g%c",*(g+i), (i%6==5) ? '\n' : ' ');
        printf("\n\t intbs2_2:  e=\n");
        for (i=0;i<36;i++) printf("%12.4g%c",*(e+i), (i%6==5) ? '\n' : ' ');
        free(g);
        free(e); free(d);
        free(c); free(b);
        free(a);
        return(r1);
      }
     else
      { printf("\n\tno memory for d in intbs2_2\n");
        return(-1); }
  }


 int pzsza (long double *c, long double *a, long double *b, int lp)
  {
     /* Z_m(bt*x) * Z_l (bt*x)=sum c_j; (m=0,2; l=0,2) v (m=3,5, l=3,5). */
              /* lp=2 -- print */
    int i,k, ki, n;
    long double *d, r1;
    d=(long double *)calloc(60,sizeof(long double));
    if ( d!=NULL )
     {
       if ( (*a) != (*b) )
        { n=4;
          for (i=0;i<3;i++)
            { ki=5*i;
              *(d+ki)=*(d+ki+6)=*(b+i);
              *(d+ki+2)=*(d+ki+8)=*(a+i); }
```

```
        *(d+4)=*(a+3)*(*(b+3));
        *(d+9)=*(a+3)*(*(b+4))+*(a+4)*(*(b+3));
        *(d+14)=*(a+4)*(*(b+3));
        xalsd (c,d,n,0);           // lp
       }
      else
       {  // l == m
        r1=*(a+4)*(*(a+4))/(*(a+2)); // A_0
        (*c)=r1;
        *(c+1)=*(a+3)*(*(a+3))-(*a)*r1; // A_1
        *(c+2)=*(a+3)*(*(a+4))*2-(*(a+1))*r1; // A_2
          // [c_11+c_12*x^2]/[a(x)^k]^2
       }
    // printf("\n\tp/p pzsza:  c=");
    // for (i=0;i<5;i++)  printf("%12.4g",*(c+i));  PRN;
     free(d);
     return(n);
   }
  else
   { printf("\n\tno memory for d in pzsza\n");
     return(-1); }
 }

int psz2 (long double *a, long double *b, long double *c, long double *g)
 {
   int i, k, k2, l5;
   long double *d;
   printf("\n\tp/p psz2: input:  array g=");
   for (i=0;i<4;i++)  printf("%10.3g",*(g+i));  PRN;
   d=(long double *)calloc(30,sizeof(long double));
   if ( d!=NULL )
    {
      if ( fabs (*b-(*c)) || fabs (*(b+1)-(*(c+1))) > 0.1 )
       { l5=0;
         for (i=0;i<30;i++) *(d+i)=0.;
         for (k=0;k<3;k++)
           { k2=5*k;
             *(d+k2)=*(c+k);
             *(d+k2+6)=*(c+k);
             *(d+k2+2)=*(b+k);
             *(d+k2+8)=*(b+k);
             *(d+k2+4)=*(g+k); }
         *(d+19)=*(g+3);
         xalsd (a,d,4,0);
       }
      else
       { l5=1;
         *(a+1)=*(g+3)/(*(b+2));
         *a=(*(g+2)-(*(b+1))*(*(g+3)))/(*(b+2));
         *(a+2)=*(g)-(*a)*(*b);
         *(a+3)=*(g+1)-(*(g+3))*(*b)-(*(b+1))*(*a);
       }
   /*   printf("\n\tp/p psz2: output: l5=%u; array a=",l5);
      for (i=0;i<4;i++)  printf("%10.3g",*(a+i));  PRN;  */
     free(d);
     return(l5);
```

```
    }
   else
    { printf("\n\tno memory for d in pzs2\n");
      return(-1); }
  }

/*------------------------- pzs2.c -------------------------*/

 int pzsd3 (long double *, long double *, long double *, int *, int);

long double inzbsn_3 (long double *cc, long double *sr, long double rr, long
double x, int *nb)
 {
  /* [b_3+b_4*x]/[b_0+b_1*x+b_2*x^2]*[c_3+c_4*x]/[c_0+c_1*x+c_2*x^2]=
     [e_0+e_1*x]/[b_0+b_1*x+b_2*x^2]+[e_2+e_3*x][c_0+c_1*x+c_2*x^2];
     =[e_0+e_1*x]/[b_0+b_1*x+b_2*x^2]+[e_2+e_3*x]/[b_0+b_1*x+b_2*x^2]^2. */

   register int k, i, j, i1, i2, i3, l, m, k2, k3, k6, l2, l3, m1, m2, m6,
nj, ns, *lp, np, mj;
   long double *a, *b, *c, *d, *e, *f, *h, *c0, *f1, *u, r0, r1, s1, r2, r3,
rb, cj, cj3, rm, rmp;
   lp=(int *)calloc(5,sizeof(int));
   u=(long double *)calloc(60,sizeof(long double));
   a=(long double *)calloc(20,sizeof(long double));
   b=(long double *)calloc(20,sizeof(long double));
   c=(long double *)calloc(60,sizeof(long double));
   d=(long double *)calloc(10,sizeof(long double));
   f=(long double *)calloc(70,sizeof(long double));
   h=(long double *)calloc(24,sizeof(long double));
   c0=(long double *)calloc(60,sizeof(long double));
   f1=(long double *)calloc(24,sizeof(long double));
   if (f1!=NULL && a!=NULL && b!=NULL && c!=NULL && d!=NULL && c0!=NULL &&
f!=NULL  && h!=NULL && lp!=NULL )
    {
      ns=0;
      for (k=0;k<3;k++) ns+=*(nb+k);
      rb=x;
      rm=1;
      rmp=1;
      for (m=0;m<3;m++)
        { m6=18*m;
          for (k=0;k<36;k++) *(b+k)=*(cc+m6+k);

          rm=bsd2 (h,b,*(sr+m),rr,*(nb+m));
          for (k=0;k<18;k++) *(u+k+m6)=*(h+k);
          printf("\n\t inzbsn_3:  u(%u,%u):\n",m,m6);
          for (i=0;i<18;i++) printf("%10.3g%c",*(u+i+m6), (i%6==5) ? '\n' :
' ');
          rmp*=rm;
        }
     /*
      *u=2;
      *(u+1)=2.5;
      *(u+2)=1.;
      *(u+3)=1;
      *(u+4)=3.;
```

```
      *(u+5)=0.;
      *(u+6)=1;
      *(u+7)=2.5;
      *(u+8)=1.;
      *(u+9)=2;
      *(u+10)=1.;
      *(u+11)=0.;
      *(u+12)=4.;
      *(u+13)=3.;
      *(u+14)=1.;
      *(u+15)=3.;
      *(u+16)=2.;
      *(u+17)=0.;
      for (i=0;i<18;i++)
        { *(u+18+i)=*(u+i);
          *(u+36+i)=*(u+i); }
      */
      for (i=0;i<10;i++) *(d+i)=0.;
      printf("\n inzbsn_3: input:
J%u(%6.3g)*J%u(%6.3g)*J%u(%6.3g)\n",*nb,*sr,*(nb+1),*(sr+1),*(nb+2),*(sr+2))
;
      for (i=0;i<54;i++) printf("%10.3g%c",*(u+i), (i%6==5) ? '\n' : ' ');
      for (i=0;i<10;i++) *(b+i)=0.;
      for (i=0;i<4;i++) *(lp+i)=0;
      cj=0.;
      for (m=0;m<3;m++)
        { m6=6*m;
          m2=2*m;
          m1=18*m;
          for (i=0;i<6;i++) *(a+i)=*(u+i+m6);
          for (l=0;l<3;l++)
            {
              l2=2*l;
              l3=6*l;
              for (i=0;i<6;i++) *(a+6+i)=*(u+i+l3+18);
              for (k=0;k<3;k++)
                { k6=6*k+36;
                  k2=2*k;
                  for (i=0;i<6;i++) *(a+12+i)=*(u+k6+i);
             pzsd3 (h,f1,a,lp,ns);
                  *(d+m2)+=*h;
                  *(d+m2+1)+=*(h+1);
                  *(f+l2+m6)+=*(h+2);
                  *(f+l2+m6+1)+=*(h+3);
                  i1=k2+l3+18*m;
                  *(c0+i1)=*(h+4);
                  *(c0+i1+1)=*(h+5);
                  if ( ns > 1 ) for (i=0;i<ns-1;i++) *(b+i)+=*(f1+i);
            }      /*  k  */
             }
      /*    printf("\n\t inzbsn_3: m=%u  arrays d(6), f(%u),
c0(%u)\n",m,m6,m1);
          for (i=0;i<2;i++) printf("%10.3g",*(d+i+m2)); PRN; PRN;
          for (i=0;i<6;i++) printf("%10.3g%c",*(f+i+m6), (i%6==5) ? '\n' : '
');  PRN;
```

```
          for (i=0;i<18;i++) printf("%10.3g%c",*(c0+i+m1), (i%6==5) ? '\n' :
' ');  */
      }
      r0=r1=0.;
      if ( ns > 1 )
       { for (i=0;i<ns-1;i++) r0=rr*r0+(*(b+ns-2-i))/(ns-1-i);
         r0*=rr;
      for (i=0;i<ns-1;i++) r1=x*r1+(*(b+ns-2-i))/(ns-1-i);
         r0-=r1*x;
      }
      cj=r0;
   /*   printf("\n\t p/p jikl: cj=%10.3g:  lp, arrays d(6), f(18),
c0(54)\n",cj);
      for (i=0;i<4;i++) printf("%6.1d",*(lp+i)); PRN; PRN;
      for (i=0;i<6;i++) printf("%10.3g",*(d+i)); PRN; PRN;
      for (i=0;i<18;i++) printf("%10.3g%c",*(f+i), (i%6==5) ? '\n' : ' ');
PRN;
      for (i=0;i<54;i++) printf("%10.3g%c",*(c0+i), (i%6==5) ? '\n' : ' ');
*/

      for (i=0;i<54;i++) *(c+i)=*(u+i);
      if ( *(lp+3) == 0 )  /* A*B*C  */
       {
         for (k=0;k<3;k++)
           { k2=2*k;
             k6=6*k;
             *(u+k6+3)=*(d+k2);
             *(u+k6+4)=*(d+k2+1);
             *(u+k6+5)=0.;
             for (l=0;l<2;l++)
               { r1=0.;
                 for (i=0;i<3;i++)
                   { i1=6*i;
                     r1+=*(f+k2+i1+l); }
                 *(u+k6+21+l)=r1;
                 r1=0.;
                 for (j=0;j<3;j++)
                   { for (i=0;i<3;i++)
                       { i1=6*j+18*i;
                         r1+=*(c0+k2+i1+l); }
                   }
                 *(u+k6+39+l)=r1;
               }
           }
       }
      else
       {
         if ( *(lp+3)==1 )
          {
            if ( *lp==2 )  /* A*A*C  */
             {
               for (l=0;l<2;l++)
                 {
                   *(u+3+l)=*(d+l)+(*(f+6+l))+(*(f+12+l));
                   *(u+15+l)=*(d+l+2)+(*(f+2+l))+(*(f+14+l));
                   *(u+27+l)=*(d+l+4)+(*(f+4+l))+(*(f+10+l));
```

```
        r1=0.;
        for (j=0;j<3;j++)
          { i1=12*j+9;
            m2=8*j;
            *(u+i1+l)=*(f+m2+l);
            *(u+i1+2)=1.; }
        for (k=0;k<3;k++)
          { k2=2*k;
            k3=6*k+3;
            for (j=0,r1=0;j<3;j++)
              { m2=18*j;
                for (i=0;i<3;i++)
                  { i1=6*i;
                    r1+=*(c0+k2+i1+m2+l); }
              }
            *(u+k3+36+l)=r1;
          }
      }
    for (k=0;k<3;k++)
      { k2=12*k;
        k3=6*k;
        for (i=0;i<3;i++) *(u+k2+i)=*(u+k2+i+6)=*(c+i+k3); }
  }
 if ( *(lp+1)==2 )    /* A*B*A  */
  {
    for (l=0;l<2;l++)
      {
        for (i=0,r1=0,r2=0,r3=0;i<3;i++)
          { i1=6*i;
            r1+=*(c0+i1+18+l)+(*(c0+i1+36+l));
            r2+=*(c0+i1+2+l)+(*(c0+i1+38+l));
            r3+=*(c0+i1+4+l)+(*(c0+i1+22+l)); }
        *(u+3+l)=*(d+l)+r1;
        *(u+15+l)=*(d+2+l)+r2;
        *(u+27+l)=*(d+4+l)+r3;
        for (j=0;j<3;j++)
          { i2=20*j;
            l2=2*j;
            i3=12*j;
            for (i=0,r1=0,r2=0;i<3;i++)
              { i1=6*i;
                r2+=*(c0+i1+i2+l);
                r1+=*(f+i1+l2+l); }
            i1=6*j;
            *(u+39+i1+l)=r1;
            *(u+9+i3+l)=r2;
          }
      }
    for (k=0;k<3;k++)
      { k2=12*k;
        k3=6*k;
        for (i=0;i<3;i++)
          { *(u+k2+i)=*(u+k2+6+i)=*(c+k3+i);
            *(u+k3+36+i)=*(c+k3+18+i); }
        *(u+k2+11)=1.; }
  }
```

```
            if ( *(lp+2)==2 )    /* A*B*B  */
             {
               for (l=0;l<2;l++)
                 {
                   for (i=0,r1=0,r2=0,r3=0;i<3;i++)
                     { i1=6*i;
                       i2=18*i;
                       r1+=*(f+i1+l)+(*(c0+i2+6+l))+(*(c0+i2+12+l));
                       r2+=*(f+i1+2+l)+(*(c0+i2+2+l))+(*(c0+i2+14+l));
                       r3+=*(f+i1+4+l)+(*(c0+i2+4+l))+(*(c0+i2+10+l)); }
                   *(u+21+l)=r1;
                   *(u+33+l)=r2;
                   *(u+45+l)=r3;
                   for (k=0;k<3;k++)
                     { k2=8*k;
                       k3=12*k;
                       for (i=0,r1=0;i<3;i++)
                         { i2=18*i;
                           r1+=*(c0+i2+k2+l); }
                       *(u+k3+27+l)=r1;
                     }
                 }
               for (k=0;k<3;k++)
                 { k2=6*k;
                   k3=12*k;
                   for (i=0;i<3;i++) *(u+k3+18+i)=*(u+k3+24+i)=*(c+i+18+k2);
                   *(u+k3+29)=1.; }
             }
          }
        else  /*  A*A*A   */
         {
           for (l=0;l<2;l++)
             { r1=*(c0+24+l)+(*(c0+30+l))+(*(c0+42+l))+(*(c0+48+l));
               r2=*(c0+2+l)+(*(c0+14+l))+(*(c0+38+l))+(*(c0+50+l));
               *(u+3+l)=*(d+l)+(*(f+6+l))+(*(f+12+l))+r1;
               *(u+21+l)=*(d+2+l)+(*(f+2+l))+(*(f+14+l))+r2;
               for (i=0,r1=0;i<2;i++)
                 { i1=18*i;
                   r1+=*(c0+i1+4+l)+(*(c0+i1+10+l)); }
               *(u+39+l)=*(d+4+l)+(*(f+4+l))+(*(f+10+l))+r1;

*(u+9+l)=*(f+l)+(*(c0+6+l))+(*(c0+12+l))+(*(c0+18+l))+(*(c0+36+l));
               *(u+15+l)=*(c0+l);

*(u+27+l)=*(f+8+l)+(*(c0+8+l))+(*(c0+20+l))+(*(c0+32+l))+(*(c0+44+l));
               *(u+33+l)=*(c0+26+l);

*(u+45+l)=*(c0+16+l)+(*(f+16+l))+(*(c0+34+l))+(*(c0+40+l))+(*(c0+46+l));
               *(u+51+l)=*(c0+52+l);
             }
           for (k=0;k<3;k++)
             { i1=6*k;
               m6=18*k;
               *(u+m6+11)=1.;
               *(u+m6+17)=2.;
```

```
                for (i=0;i<3;i++)
*(u+m6+i)=*(u+m6+6+i)=*(u+m6+12+i)=*(c+i+i1);
              }
           }
        }
      printf("\n inzbsn_3: output: cj=%7.3g:
J%u(%8.3g)+J%u(%8.3g)+J%u(%8.3g)\n",cj,*nb,*sr,*(nb+1),*(sr+1),*(nb+2),*(sr+
2));
      for (i=0;i<54;i++) printf("%10.3g%c",*(u+i), (i%6==5) ? '\n' : ' ');
      if ( ns > 0 ) for (i=0;i<ns;i++) printf("%10.4g",*(b+i)); PRN;
      cj3=inbsz (u,rr,x,9,2)+cj;
      printf("\n inzbsn_3:
int(Z_%u(%7.3f*r)*Z_%u(%7.3f*r)*Z_%u(%7.3f*r)=%10.5g\n",*nb,*sr,*(nb+1),*(sr
+1),*(nb+2),*(sr+2),cj3);
      free(u); free(f1);
      free(c0);
      free(h); free(f);
      free(d); free(c);
      free(b); free(a);
      free(lp);
      return(cj3);
    }
   else
    { printf("\n\tno memory for d, c, b or a in jikl_bs\n");
      return(-1); }
 }

 int pzsd3 (long double *d, long double *h, long double *a, int *lp, int ns)
  {
    /* [(a_3+a_4*x)/(a_0+a_1*x+a_2*x^2)]*[(a_9+a_10*x)/(a_6+a_7*x+a_8*x^2)]*
                  *[(a_15+a_16*x)/(a_12+a_13*x+a_14*x^2)]=
          = d_0+d_1*x+d_2*x^2+...d_(nj-1)*x^(nj-
1)+[(h_0+h_1*x)/(a_0+a_1*x+a_2*x^2)]+

+[(h_2+h_3*x)/(a_6+a_7*x+a_8*x^2)]+[(h_4+h_5*x)/(a_12+a_13*x+a_14*x^2)].  */
      /* ns=nb_0+nb_1+nb_2    */
   int i, k, i1, i2, j, j1, n, nj, nk;
   long double *e, *c, *q, *f, *g, r0, r1, r2;
   e=(long double *)calloc(10,sizeof(long double));
   c=(long double *)calloc(30,sizeof(long double));
   q=(long double *)calloc(20,sizeof(long double));
   g=(long double *)calloc(20,sizeof(long double));
   f=(long double *)calloc(60,sizeof(long double));
   if ( e!=NULL && f!=NULL && c!=NULL && g!=NULL && q!=NULL )
    {
      printf("\n\t p/p pzsd3: input  array  a(18)\n");
      for (i=0;i<18;i++) printf("%10.4g%c",*(a+i), (i%6==5) ? '\n' : ' ');
PRN;
      r0=*(a+3)*(*(a+9));
      r1=*(a+3)*(*(a+10))+(*(a+4))*(*(a+9));
      r2=*(a+4)*(*(a+10));
      for (i=0;i<20;i++) *(q+i)=0.;
      *(q+ns+1)=r0*(*(a+15));
      *(q+ns+2)=r0*(*(a+16))+r1*(*(a+15));
      *(q+ns+3)=r1*(*(a+16))+r2*(*(a+15));
      *(q+ns+4)=r2*(*(a+16));
```

```
      for (i=0;i<30;i++) *(c+i)=0.;
      *c=*(a+6)*(*(a+12));
      *(c+1)=*(a+6)*(*(a+13))+(*(a+7))*(*(a+12));
      *(c+2)=*(a+6)*(*(a+14))+(*(a+7))*(*(a+13))+(*(a+8))*(*(a+12));
      *(c+3)=*(a+7)*(*(a+14))+(*(a+8))*(*(a+13));
      *(c+4)=*(a+8)*(*(a+14));
      *e=*c*(*a);
      *(e+1)=*c*(*(a+1))+(*(c+1))*(*a);
      *(e+6)=*(c+4)*(*(a+2));
      for (i=0;i<4;i++)
*(e+2+i)=*(c+i)*(*(a+2))+(*(c+i+1))*(*(a+1))+(*(c+i+2))*(*a);
    /*  printf("\n pzsd3 -  ns=%u -- array q, e\n",ns);
      for (i=0;i<ns+4;i++) printf("%10.3g",*(q+i)); PRN;
      for (i=0;i<7;i++) printf("%11.4g",*(e+i)); PRN;   */
      for (i=0;i<20;i++) *(h+i)=0.;
      if ( ns < 2 ) for (i=0;i<6;i++) *(g+i)=*(q+i);
      else
       { n=ns-2;
         *(h+n)=*(q+ns+4)/(*(e+6));
         if ( n==0 ) for (k=0;k<6;k++) *(g+k)=*(q+k)-(*h)*(*(e+k));
         if ( n > 0 )
          { nk=( n<6 ) ? n : 5;
            for (k=0;k<nk;k++)
              { for (i=0,r0=0.;i<=k;i++) r0+=*(e+5-k+i)*(*(h+n-i));
                *(h+n-1-k)=(*(q+ns+3-k)-r0)/(*(e+6)); }
            if (n > 5 )
             { nk=n-5;
               for (k=0;k<nk;k++)
                 { for (i=0,r0=0.;i<6;i++) r0+=*(e+5-k+i)*(*(h+6+k-i));
                   *(h+k)=(*(q+6+k)-r0)/(*(e+6)); }
             }
            for (k=0;k<6;k++)
              { for (i=0,r0=0.;i<=k;i++) r0+=*(e+i)*(*(h+k-i));
                *(g+k)=*(q+k)-r0; }
          }
       }
      *(c+10)=*a*(*(a+12));
      *(c+11)=*a*(*(a+13))+(*(a+1))*(*(a+12));
      *(c+12)=*a*(*(a+14))+(*(a+1))*(*(a+13))+(*(a+2))*(*(a+12));
      *(c+13)=*(a+1)*(*(a+14))+(*(a+2))*(*(a+13));
      *(c+14)=*(a+2)*(*(a+14));
      *(c+20)=*a*(*(a+6));
      *(c+21)=*a*(*(a+7))+(*(a+1))*(*(a+6));
      *(c+22)=*a*(*(a+8))+(*(a+1))*(*(a+7))+(*(a+2))*(*(a+6));
      *(c+23)=*(a+1)*(*(a+8))+(*(a+2))*(*(a+7));
      *(c+24)=*(a+2)*(*(a+8));
      if (fabs(*(a+6)-(*(a+12))) <0.001 && fabs(*a-(*(a+12)))<0.001 &&
fabs(*a-(*(a+6)))<0.001 ) /* && ( kk==ll && ll==mm ) ) */
       {
         *(d+1)=*(g+5)/(*(c+4));
         *d=(*(g+4)-(*(c+3))*(*(d+1)))/(*(c+4));
         *(d+3)=(*(g+3)-(*(c+3))*(*d)-(*(c+2))*(*(d+1)))/(*(a+2));
         *(d+2)=(*(g+2)-(*(c+2))*(*d)-(*(c+1))*(*(d+1))-
(*(a+1))*(*(d+3)))/(*(a+2));
         *(d+4)=*g-(*c)*(*d)-(*a)*(*(d+2));
```

```
         *(d+5)=*(g+1)-(*(c+1))*(*d)-(*c)*(*(d+1))-(*(a+1))*(*(d+2))-
(*a)*(*(d+3));
      /*   for (i=0;i<3;i++) *(c+10+i)=*(a+i);
         *(c+13)=*(c+14)=0.;
         for (i=1;i<5;i++) *(c+20+i)=0.;
         *(c+20)=1; */
         *(lp+3)=2; }
      else
       { if (fabs(*a-(*(a+12))) <0.001 ) /* && kk==mm )  */
          { for (i=0;i<3;i++) *(c+20+i)=*(a+i+6);
            *(c+23)=*(c+24)=0.;
            *(lp+1)=2;
            *(lp+3)=1; }
         if (fabs(*a-(*(a+6))) <0.001 ) /* && ll==mm )  */
          { for (i=0;i<3;i++) *(c+10+i)=*(a+i+12);
            *(c+13)=*(c+14)=0.;
            *lp=2;
            *(lp+3)=1; }
         if (fabs(*(a+6)-(*(a+12))) <0.001 ) /* && ll==kk )  */
          { for (i=0;i<3;i++) *(c+20+i)=*(a+i);
            *(c+23)=*(c+24)=0.;
            *(lp+2)=2;
            *(lp+3)=1; }
         for (i=0;i<60;i++) *(f+i)=0.;
   /*   printf("\n pzsd3 - lp(%u)=",ns);
      for (i=0;i<4;i++) printf("%6.1d",*(lp+i)); PRN;  */
         for (i=0;i<5;i++)
           { i1=7*i;
             *(f+i1)=*(f+i1+8)=*(c+i);
             *(f+i1+2)=*(f+i1+10)=*(c+i+10);
             *(f+i1+4)=*(f+i1+12)=*(c+i+20); }
         for (i=0;i<6;i++)
           { i1=7*i+6;
             *(f+i1)=*(g+i); }
         xalsd (d,f,6,0);
       }
      printf("\n pzsd3: output -- d=");
      for (i=0;i<6;i++) printf("%10.4g",*(d+i)); PRN;
      for (i=0;i<6;i++) printf("%10.4g",*(g+i)); PRN;
      for (i=0;i<6;i++) printf("%10.4g",*(c+i)); PRN;
      free(f); free(g);
      free(q); free(c);
      free(e);
      return(1);
    }
   else
    { printf("\n\tno memory for e, c, q or f in pzsd3\n");
      return(-1); }
 }


 float umlk0 (float *sl, float *zn, float *un, float *gs, float nu, float t,
float z, float x, int n)
  {
    /* U^(0)(x,z,t)=Z_U(z)X_U(x)*U_n_{m,l}e^(-gs_{m,l}*t)   */
    int m, k, m1, j;
```

```
  float zz, xx, r2, uu, uux;
  uu=0.;

  for (m=0;m<n;m++)
    { m1=m*n;
    zz=sin(*(sl+n+m)*z)-(*(sl+n+m))/nu*cos(*(sl+n+m)*z)/(*(zn+n+m));
      uux=0.;
      for (k=0;k<n;k++)
        { xx=sin(*(sl+k)*x)/(*(zn+k));
          r2=*(un+m1+k)*(1-exp(-(*(gs+k+m1))*t));
          uux+=r2*xx; }
      uu+=uux*zz;
    }
  return(uu);
}
```