Институт по металознание, съоръжения и технологии с център по хидро- и аеродинамика „Акад. А. Балевски“ — БАН

Магистър по изчислителна математика и математическо моделиране
Людмил Владимиров Йовков

МАТЕМАТИЧЕСКО МОДЕЛИРАНЕ НА КРИСТАЛИЗАЦИЯТА НА МЕТАЛНИ СПЛАВИ

ДИСЕРТАЦИОНЕН ТРУД

за присъждане на образователна и научна степен „доктор“
по докторска програма „Металознание и термична обработка на металите“,
шифър 02.09.01

Научни ръководители:

Доц. д-р Валентин Манолов

Проф. д-р Татяна Черногорова

София, 2020

Настоящият дисертационен труд се посвещава на всички мои ученици, колеги и приятели, които ме подкрепяха безусловно до самия край, и на Антония Лафчийска, която си отиде без време от този свят.

Съдържание

1 Анализ на научната литература **10**

1.1 Исторически бележки. Класическа и обобщена задача на Стефан . . . 10

1.2 Класификация на числените методи за приближено решаване на задачи с фазови преходи . . . 14

2 Цели и задачи на дисертационния труд **22**

3 Едномерен математически модел за кристализация на метални сплави **26**

3.1 Постановка на задачата . . . 26

3.2 Диференчна схема и числени симулации . . . 29

3.2.1 Влияние на центровете на кристализация върху процеса на затвърдяване . . . 34

4 Двумерен математически модел за кристализация на метална сплав **38**

4.1 Постановка на задачата . . . 38

4.2 Относителен дял на твърдата фаза $\psi(u)$ . . . 45

4.3 Диференчна схема . . . 48

4.3.1 Диференчна схема в направление r . . . 51

4.3.2 Диференчна схема в направление z . . . 61

4.4 Изглаждане на делта-функцията . . . 69

4.5 Пресмятане на интегралните коефициенти $c_{i,j}$ . . . 71

4.6 Методически изследвания . . . 78

4.6.1 Тестов пример в направление r . . . 78

4.6.2 Тестов пример в направление z . . . 82

4.6.3 Резултати от методическите изследвания . . . 85

4.7 Валидиране на математическия модел на база експериментални данни . . . 86

4.8 Параметрично изследване . . . 90

5 Създаване на графичен потребителски интерфейс чрез средствата на софтуера MATLAB 95
5.1 Графичен потребителски интерфейс 95
5.2 Класът *handles* 98
5.3 Основни графични обекти в GUI 99
5.4 Създаване на графичен потребителски интерфейс за решаване на двумерна кристализационна задача с фазов преход в цилиндрични координати 100

6 Изводи от работата. Приноси. Апробация 109
6.1 Изводи от работата по дисертацията 109
6.2 Приноси на дисертационния труд 110
6.2.1 Научни приноси на дисертационния труд 110
6.2.2 Научно-приложни приноси на дисретационния труд 111
6.3 Апробация на резултатите 111

Благодарности 113

Декларация за оригиналност 114

Библиография 115

Приложение 123

Списък на фигурите

1.1 *Класическа задача на Стефан с две фази* 11
1.2 *Обобщена задача на Стефан в бинарна среда* 13
1.3 *Схематично представяне на половината от осното сечение на многослойна кокила:* D_1 *— област, заета от металната сплав,* D_2 *— основна част на кокилата,* D_3 *— втулка,* D_4 *— мъртва глава,* D_5 *— тънко термично съпротивление (газова междина, обмазка)* 19

3.1 *Разпределение на температурата като функция на времето, алуминиева сплав* ***AlSi7Mg*** 33
3.2 *Разпределение на температурата като функция на времето, сив чугун* 34
3.3 *Влияние на броя на кристализационните центрове върху преохлаждането, алуминиева сплав* ***AlSi7Mg*** 35
3.4 *Влияние на броя на кристализационните центрове върху преохлаждането, сив чугун* 36
3.5 *Валидиране на едномерния математически модел* (3.7) — (3.12) *въз основа на експериментални данни за алуминиева сплав* ***AlSi7Mg*** 37

4.1 *Осно сечение на цилиндрична форма* 38
4.2 *Схематично представяне на фазова диаграма за аналитично определяне на функцията* $\psi(u)$ 45
4.3 *Графика на функцията* $\psi(u)$*, алуминиева сплав* ***AlSi7Mg*** . 48
4.4 *Неравномерна отместена мрежа* $\widehat{\omega}^*_{h^r}$ *в направление* r . . . 49
4.5 *Неравномерна неотместена мрежа* $\widehat{\omega}_{h^z}$ *в направление* z . . 50
4.6 *Равномерна неотместена мрежа* $\overline{\omega}_\tau$ *в направление* t 50
4.7 *Пресмятане на параметъра* Δ *в случая на намаляваща мрежова функция, когато:* ***а)*** $d_{i-1} < d_i$*;* ***б)*** $d_{i-1} > d_i$ 69
4.8 *Пресмятане на параметъра* Δ *в случая на растяща мрежова функция, когато:* ***а)*** $d_{j-1} < d_j$*;* ***б)*** $d_{j-1} > d_j$ 70

4.9 *Разположение на полуцелите възли* $r_{i\pm1/2}$ *спрямо точките* $r_{S\pm\Delta}$ *и* r_L 72
4.10 *Разположение на полуцелите възли* $z_{j\pm1/2}$ *спрямо точките* $z_{S\pm\Delta}$ *и* z_L 73
4.11 *Схематично представяне на областта* $\overline{D} = \overline{D}_1 \cup \overline{D}_2$ 78
4.12 *Схематично представяне на областта* $\overline{D} = \overline{D}_1 \cup \overline{D}_2$ 82
4.13 *Сравнение между точното решение* $u(r,t)$*, даващо се с формулата* (4.84)*, и приближеното решение, получено с метода на крайните разлики* 86
4.14 *Сравнение между точното решение* $u(z,t)$*, даващо се с формулата* (4.90)*, и приближеното решение, получено с метода на крайните разлики* 87
4.15 *Схематично представяне на формата за отливане на отливката. Термодвойката е разположена в точката с координати* $(r^*;\, z^*) = (0;\, 0.0565)$ m 88
4.16 *Валидиране на теоретичния модел* (4.2) — (4.9) *чрез сравнение с експериментални резултати, получени в лабораторни условия* 89
4.17 *Изотермични линии на солидуса в обема на отливката при отсъствие на топлинни загуби* ($\overline{\alpha} = 0$) 89
4.18 *Изотермични линии на солидуса в обема на отливката при наличие на топлинни загуби* ($\overline{\alpha} \neq 0$) 90

5.1 *Стандартен изглед на графичния прозорец в средата* ***guide*** 99
5.2 *Примерна схема на* ***GUI*** *за решаване на двумерна кристализационна задача с фазов преход в цилиндрична координатна система* 101
5.3 *Меню* ***Save*** 103
5.4 *Меню* ***Plot*** 104
5.5 *Меню* ***Interpolation*** 105
5.6 *Билинейна интерполация по точките* Q_{11}, Q_{12}, Q_{21} *и* Q_{22} . 106
5.7 *Функционална зависимост на температурата от времето в целия обем на отливката. Подменюто* ***Plot as a function of time*** *е свързано с диалогово въвеждане на координатите на точката* $(r;\, z)$*. По подразбиране е заложено* $(r;\, z) = (0.010;\, 0.010)$*. Представената симулация е извършена именно за тази точка.* 106
5.8 *Диалогов прозорец, изискващ въвеждането на данни за билинейна интерполация* 107
5.9 *Числен резултат от билинейна интерполация* 107

5.10 *Кубична интерполация (в радиално направление) за получаване стойностите на приближеното решение в точки, които не принадлежат на мрежата* 108

Списък на таблиците

1.1 *Сравнителна таблица за предимствата и недостатъците на двата най-често използвани числени метода за решаване на задачата на Стефан* 16

3.1 *Стойности на параметрите на модела за два вида сплави: алуминиева сплав* ***AlSi7Mg*** *[82] и сив чугун [23], [83]* . 30

4.1 *Стойности на параметрите* u_L, u_S, $c^{(1)}$, $c^{(2)}$, a_1 *и* a_2, *които определят функцията* $\psi(u)$ *по формулата* (4.24) . . . 48

4.2 *Пресмятане на коефициента* $\int_{r_{i-1/2}}^{r_{i+1/2}} c(u)\,\mathrm{d}\,r$ *при намаляваща функция* 74

4.3 *Пресмятане на коефициента* $\int_{z_{j-1/2}}^{z_{j+1/2}} c(u)\,\mathrm{d}\,z$ *при растяща функция* 76

4.4 *Геометрични параметри и топлофизични характеристики на едномерната в направление* r *задача* (4.73) — (4.77) . 79

4.5 *Стойности на допълнителните параметри* a_m, b_m, d_m $(m = 1, 2)$ *и на коефициентите* $\alpha_k^{(1)}$, α *за едномерната в направление* r *задача* (4.73) — (4.77) 81

4.6 *Геометрични параметри и топлофизични характеристики на едномерната в направление* z *задача* (4.85) — (4.89) . 83

4.7 *Стойности на коефициентите* α_k, $\widetilde{\alpha}$ *и* $\overline{\alpha}$ *за едномерната в направление* z *задача* (4.85) — (4.89) 84

4.8 *Числени резултати за сходимост на едномерната задача* (4.73) — (4.77) *в направление* r 85

4.9 *Числени резултати за сходимост на едномерната задача* (4.85) — (4.89) *в направление* z 85

4.10 *Неравномерна мрежа* $\widehat{\omega}^*_{h_i^r}$ 86

4.11 *Неравномерна мрежа* $\widehat{\omega}_{h_j^z}$ 86
4.12 *Температурен профил на цилиндрична отливка от сплав* ***AlSi7Mg***, $t = 100$ s 90
4.13 *Температурен профил на цилиндрична отливка от сплав* ***AlSi7Mg***, $t = 400$ s 91
4.14 *Степен на влияние на параметъра* α 92
4.15 *Степен на влияние на параметъра* $\widetilde{\alpha}$ 93
4.16 *Степен на влияние на параметъра* $\overline{\alpha}$ 93
4.17 *Степен на влияние на параметъра* α_k 94
4.18 *Степен на влияние на параметъра* κ 94

Глава 1

Анализ на научната литература

1.1 Исторически бележки. Класическа и обобщена задача на Стефан

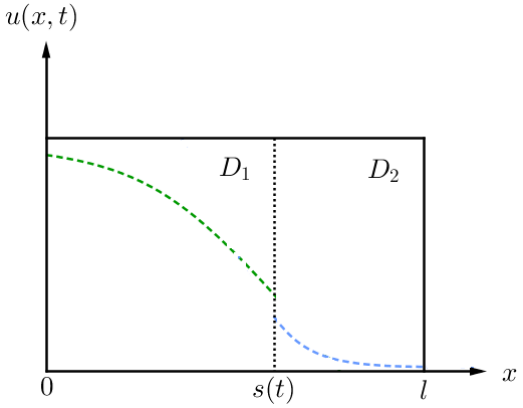
Съвременните достижения в изчислителната математика и математическото моделиране са основа за теоретичното изследване на голяма част от процесите в природата. Освен обичайните начално-гранични задачи, с които се занимават частните диференциални уравнения, съществува и друга, специална диференциална задача, която отчита промяната в агрегатните състояния на веществата. В литературата тя е известна като *задача на Стефан*.

По своята същност задачата на Стефан е особен тип гранична задача за уравнението на топлопроводността, която изисква определянето не само на температурното поле, но и разположението на неизвестната фазова граница между състоянията за всеки момент от време. Наличието на такава фазова граница и нейното локализиране чрез задаване на допълнително гранично условие е характерна особеност на задачата на Стефан.

Счита се, че първата научна работа, посветена на изучаването на задачи с фазов преход, принадлежи на Габриел Ламе и Беноа Клапейрон: *Memoire sur la solidification par refroidissment d'un globe liquide* (1831 г.) [58], [80]. Те установяват, че дебелината на твърдата фаза, която се образува при затвърдяването на еднородна смес, е пропорционална на $\sqrt{t}$, обаче не успяват да изчислят коефициента на пропорционалност. Едва по-късно, през 1889 г., Йозеф Стефан публикува четири статии за четири различни задачи, свързани с фазов преход [52], и успява да постави гранично условие върху границата на раздела на фазите. Така е дадено началото на нов клас физико-математически задачи, които представля-

ват значителен интерес и в днешно време. Типични примери за задача на Стефан са процесите на замръзване на водата, на преразпределение на концентрацията на вещество при дифузия, на образуване на метален слитък, на неизотермично изпарение на идеални смеси, на лазерно излъчване и др.

На фиг. 1.1 в декартова координатна система схематично е представена едномерна двуфазна система с подвижна фазова граница $x = s(t)$. Съществено е, че фазовото превръщане се извършва при постоянна тем-



Фигура 1.1: *Класическа задача на Стефан с две фази*

пература на фазов преход u^*. Например, ако разглеждаме задачата за кристализация на метална стопилка, отлята във форма, то D_1 представлява течната фаза, D_2 — твърдата фаза, $u(x,t)$ — температурното разпределение като функция на пространствената променлива и времето, а границата между течната и твърдата фаза е линията $u = u_S$. Тук u_S е температурата солидус.

По традиция решаването на класическата задача на Стефан води до получаване на съответни температурни полета. Нека да разгледаме най-простата едномерна задача на Стефан с две фазови състояния (вж. фиг. 1.1) за кристализацията на метална стопилка. Във всички вътрешни точки на областта $D = D_1 \cup D_2$ разпределението на топлината се описва с уравнението на топлопроводността

$$c(u)\rho\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\lambda(u)\frac{\partial u}{\partial x}\right),\ 0 < x \leq l,\ 0 < t \leq T. \tag{1.1}$$

Величините c, ρ и λ имат обичайния смисъл на топлофизичните характеристики топлоемкост, плътност и коефициент на топлопроводност. За тях предполагаме, че са на части гладки функции. Да означим с индекс

1 онези от тях, които се отнасят за течната фаза, и с индекс 2 — тези за твърдата. Тогава можем да запишем

$$c,\ \rho,\ \lambda = \begin{cases} c_1,\ \rho_1,\ \lambda_1, & x \in D_1, \\ c_2,\ \rho_2,\ \lambda_2, & x \in D_2. \end{cases}$$

По-нататък задаваме началното условие

$$u(x,0) = u_0(x),\ 0 \le x \le l, \tag{1.2}$$

както и граничните условия

$$u(0,t) = \mu_1(t),\ u(l,t) = \mu_2(t),\ 0 < t \le T, \tag{1.3}$$

върху външните граници на D.

При фазовия преход от течно в твърдо агрегатно състояние се отделя т. нар. *скрита топлина на кристализация* κ. Граничното условие върху вътрешната подвижна фазова граница $x = s(t)$, което моделира това явление, се поставя така:

$$\lambda \frac{\partial u}{\partial x}(s-0,t) - \lambda \frac{\partial u}{\partial x}(s+0,t) = \kappa\rho \frac{\mathrm{d}\, s}{\mathrm{d}\, t}. \tag{1.4}$$

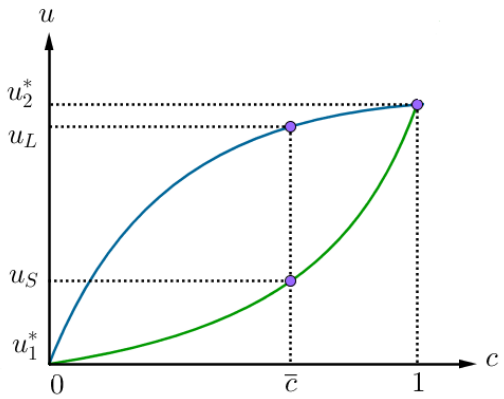
Физически то означава, че разликата от топлинните потоци отляво и отдясно на линията $x = s(t)$ е пропорционална на скоростта, с която се придвижва във времето фазовата граница. При това коефициентът на пропорционалност е равен на $\kappa\rho$.

Както беше казано по-горе, температурата u^* на фазовия преход остава постоянна, ето защо трябва да поискаме още по границата $s(t)$ да е изпълнено равенството

$$u(s(t),t) = u^*. \tag{1.5}$$

И така, уравненията (1.1) — (1.5) дават пълния математически модел на най-простата едномерна двуфазна задача на Стефан. На пръв поглед нейната експлицитна формулировка е достатъчна за решаване на широк клас задачи с фазови преходи. Теоретичните изследвания и реалните експерименти в лабораторни условия обаче показват, че основните ѝ предположения са валидни само при промяна в агрегатните състояния на чисти вещества без примеси. За първи път през 80-те години на XX в. класическата задача на Стефан претърпява модификации на идейно ниво. Тези нови течения в досегашните разглеждания постепенно разширяват кръгозора, в който задачата се е възприемала до момента, и водят до т. нар. *обобщена задача на Стефан* [6], [8], [9].

Обобщената задача на Стефан запазва основните черти на класическата такава, но е напълно адаптирана към физични процеси, които протичат с фазови преръщания на съставни вещества от едно агрегатно състояние в друго. Фундаменталните ѝ концепции ще илюстрираме върху следния пример. Да разгледаме хода на кристализационния процес при сплав $\mathbb{S}$, съставена от две компоненти. Да означим с индекс 1 всички физични величини, които се отнасят за първата компонента, и с индекс 2 — тези за втората. Поради комплексната химична структура на сплавта $\mathbb{S}$ промяната в агрегатните състояния на двете ѝ компоненти ще протича съответно при различни температури на фазов преход u_1^* и u_2^*. С течение на времето се установява цял температурен интервал, т. нар. *двуфазна зона*, съдържащ едновременно кристали и течна фаза [8], [33], [36].



Фигура 1.2: *Обобщена задача на Стефан в бинарна среда*

На фиг. 1.2 схематично сме показали фазовата диаграма за конкретната сплав. Очевидно ако положим $c = 0$ или $c = 1$, в обема на отливката ще присъства точно една от съставляващите компоненти [6].

Нека $\overline{c}$ е междинна концентрация от интервала $(0; 1)$. Сега равновесната температура ликвидус u_L ще бележи началото на кристализацията, а равновесната температура солидус u_S — края. Отделянето на скритата топлина на кристализация в двуфазната зона се отчита чрез задаване на ефективната топлоемкост $c_{\text{ef.}}(u)$ по формулата [23], [28], [29], [30]

$$c_{\text{ef.}}(u) = c_S - \kappa \frac{\mathrm{d}\,\psi(u)}{\mathrm{d}\,u},$$

където c_S е коефициентът на топлоемкост на чистия метал в твърдо състояние, а $\psi(u)$ е обемната част на твърдата фаза. Обичайно функцията

$\psi(u)$ се определя с помощта на диаграмата на състоянията и както ще бъде показано по-нататък в изложението, зависимостта ѝ от температурата е дробно-линейна. Следователно познаването на идеята за двуфазната зона в интервала $(u_S; u_L)$ при многокомпонентните сплави притежава голяма практическа стойност в изследването на фазовите превръщания.

1.2 Класификация на числените методи за приближено решаване на задачи с фазови преходи

В течение на годините е публикуван голям брой научни трудове, които са посветени на изучаването на задачи с фазови преходи. Освен строгия физичен смисъл, вложен в тях, авторите вземат предвид и математическия смисъл, а именно как да се намери аналитично решение. Ако такова решение въобще съществува, то трябва да бъде единствено. Единствеността изисква, например, да бъде изпълнено условието на Липшиц в някаква ограничена и затворена област. Поради нелинейното условие за спрягане (1.4), (1.5), което трябва да е в сила по дължината на подвижната граница, ще е налице прекъсване на предполагаемото решение. Освен това разглежданите задачи търпят логично обобщение до случая, когато пространствените измерения са p на брой, $p \geq 2$.

Изложените факти недвусмислено означават, че диференциалната задача на Стефан трудно би се поддала на обичайните техники за намиране на интегрални криви в явен вид дори и при опростени предположения на математическия модел (едномерност по пространството, пренебрегване влиянието на въздуха, триенето и т. н.). Например в [16] задачата за ламинарно разпространение на топлина вследствие на горене притежава автомоделно решение от тип „бягаща“ вълна [5], ако числото на Люис, Le, е равно на нула. Ако $Le \neq 0$, то автомоделното решение не съществува.

Най-широко използваният числен метод за приближено решаване на задачата на Стефан си остава методът на крайните разлики. Той е предпочитан благодарение на своята простота, логичност и сравнително лесна компютърна реализация. Гореспоменатото, разбира се, не отхвърля възможността в определени кръгове на природните науки и техниката успешно да се прилага мощният и доста по-сложен метод на крайните елементи. Например в статиите [64], [65], [68] са представени конструирането и изследването на числена схема по метода на крайните елементи за едномерната двуфазна задача на Стефан. В [64], [65] авторът използва

линейни базисни функции. Чрез серия от числени симулации той установява реда на сходимост както за неизвестната температура $u(x,t)$, така и за подвижната фазова граница $s(t)$. Директно е показано, че редът и в двата случая е от порядъка на $1/n$, където n е броят на точките по пространството. В [68] пък е представена схема с втори ред на точност при използване на квадратични крайни елементи.

Както вече беше казано, съществените особености при решаването на задачата на Стефан са наличието на фазова граница между агрегатните състояния на веществата и предварително неизвестното ѝ разположение в пространството. В традиционния запис на метода на крайните разлики връзките (1.4), (1.5) са заложени като самостоятелни гранични условия. През 1965 г. А. А. Самарский и Б. Д. Моисеенко публикуват своята знаменита статия *„Экономичная схема сквозного счета для многомерной задачи Стефана“* [4], която систематизира и обобщава тяхната работа върху задачата на Стефан в многомерния случай. Те доказват по неоспорим начин, че модификацията на коефициента на топлоемкост $c(u)$ чрез въвеждането на т. нар. „изгладена“ или още приблизително делтаобразна функция [1], *без явното отделяне на фазовата граница*, е еквивалентна на изпълнението на условията (1.4), (1.5). Така приложението на обичайните хомогенни и консервативни диференчни схеми [3] към задачата (1.1) — (1.5) вече е очевидно и ще доведе до система от линейни алгебрични уравнения с тридиагонална матрица, която може да се реши по метода на Томас (на прогонката) [49], [50], [51]. По-късно ние ще изложим подробно методологията на прехода от експлицитно задаване на гранично условие върху подвижната фазова граница към неявното му включване в коефициентите на диференциалното уравнение [6].

Оказва се обаче, че така представената на концептуално ниво диференчна схема притежава един недостатък — липса на добра точност при определяне положението на линията $s(t)$ дори и в случая на еднотипни формули за диференчните коефициенти. Както е посочено в [16], при устремяване на параметъра на „изглаждането“ към нула и сгъстяване на дискретната мрежа намалява и грешката, която се допуска при локализиране на фазовата граница. Всичко това води до повишение в броя на аритметичните и логическите операции.

Да разгледаме друг тип числени методи от групата на крайните разлики за приближено решаване на задачата на Стефан — т. нар. *метод на адаптивните мрежи* [17], [19], [37], [38]. В теорията на апроксимациите

[1] Всъщност Самарский и Моисеенко „изглаждат“ и коефициента на топлопроводност λ, предполагайки, че е функция на температурата u. Тъй като в разглежданията, които ще проведем до края на настоящия дисертационен труд, ние ще считаме $\lambda = \text{const}$, то няма повече да се спираме на въпроса за неговото „изглаждане“.

за *адаптивна* се приема такава мрежа, която отговаря на следните две условия:

1. броят на възлите ѝ поначало не е фиксиран и варира в зависимост от момента от време;

2. разположението на възлите ѝ не е хаотично, произволно, а се определя на ниво диференциално уравнение.

Последното означава, че всяко поредно конструиране на мрежата е свързано с интегрирането на специално диференциално уравнение, чието решение представлява точно търсената мрежа. Този прийом, въпреки своята сложна алгоритмична реализация, напоследък се предпочита все по-често. Съответният адаптивен алгоритъм *отделя явно границата на раздела на фазите* и се отличава със следните достойнства:

- ☐ чрез динамична пренастройка на мрежата се постига освобождаване от основната трудност при изследванията, а именно подвижната граница;

- ☐ установява се висок ред на сходимост на приближеното решение към точното, който се обуславя от разпределението на мрежовите възли, а не от увеличението в техния брой;

- ☐ локализирането на фазовата граница притежава много по-висока точност в сравнение с метода, предложен от Самарский и Моисеенко.

В таблица 1.1 сме обобщили предимствата и недостатъците на двата начина за приближено решаване на задачата на Стефан — метода на **х**омогенните **к**онсервативни **д**иференчни **с**хеми (Х. К. Д. С.) и метода на **а**даптивните **м**режи (А. М.).

Числен метод	**Точност при опр. на** $s(t)$	**Алг. сложност**
Х. К. Д. С.	ниска	средна
А. М.	висока	висока

Таблица 1.1: *Сравнителна таблица за предимствата и недостатъците на двата най-често използвани числени метода за решаване на задачата на Стефан*

Въпросът за това, кой от двата метода е по-удобен, няма еднозначен отговор. Практиката показва, че при неголеми изисквания към точността, с която локализираме фазовата граница, за предпочитане е методът на Х. К. Д. С., в противен случай — методът на А. М.

Методът на Х. К. Д. С., макар и с по-ниска степен на точност по отношение определянето на положението на линията $s(t)$ за всеки момент от време, се счита за по-универсален. След публикуването на статията [4] много автори се позовават на основните ѝ идеи и ги прилагат в различни задачи за пренос на топлина. Те се оказват най-ефикасни при многомерни задачи на математическата физика, както се вижда от [16], [30], [57]. Двумерната задача за топенето на цинк в работата [16] използва подхода на Самарский и Моисеенко за „изглаждане“ на коефициентите. Авторите подчертават, че резултатите от изчислителния процес зависят едновременно от параметъра на делтаобразната функция и от начина, по който се „изглажда“ енталпията на фазовия преход. В повечето трудове, посветени на този въпрос (вж. напр. [4], [57], [60]), се прилага най-простата линейна интерполация или интерполация с полином от по-висока степен [13]. Допълнително в [16] авторите споменават следния важен факт. Известен е аналитичен метод [33], [34] за приближено решаване на едномерната задача на Стефан, при който температурното разпределение по пространствените координати удовлетворява уравнението на Лаплас, а границата на областта се изменя по условието на Стефан. Приближеното решение в този случай ще клони към точното асимптотически.

Трябва да се вземе предвид следният факт: в редица случаи „изглаждането“ на коефициента на топлоемкост $c(u)$ е процедура, чувствителна към параметъра на делтаобразната функция. Ето защо е естествено към този подход да се пристъпва с известно внимание.

Нюанси на класическата задача на Стефан присъстват в значителна част от научните изследвания, посветени на многомерните кристализационни задачи, задачите от газова динамика и задачите за топлопроводност. И тук предпочитан подход за приближено решаване си остава методът на крайните разлики, разпределен почти равномерно в своите две основни направления — метода на адаптивните мрежи (А. М.) и метода на хомогенните консервативни диференчни схеми (Х. К. Д. С). Докато в работи [17], [19], [21], [37], [38] и [39], например, можем да забележим предпочитанията към метода на А. М. за числено изследване на задачи от газовата динамика и топлопреноса, то обратно — в работи [5], [7], [10], [16], [26], [28], [30], [45], [70], [71] са демонстрирани предимствата на метода на Х. К. Д. С. В [50] и [51] са използвани устойчиви неявни схеми за квазилинейни диференциални задачи. Допълнително в [17] авторите излагат на преден план възможността споменатите методи да се използват успоредно.

Да разгледаме някои конкретни аспекти на метода на Х. К. Д. С. за приближено решаване на кристализационни задачи с наличие на фазов преход. От класическата теория на мрежовите методи [3], [44] е добре

известно, че явната двуслойна схема

$$c(y^n)\rho\frac{y_i^{n+1}-y_i^n}{\tau}=\frac{1}{h}\left[\lambda(y^n)\frac{y_{i+1}^n-y_i^n}{h}-\lambda(y^n)\frac{y_i^n-y_{i-1}^n}{h}\right]$$

за едномерното уравнение (1.1) има ограничено приложение поради силните условия за устойчивост. Обикновено се залага на итерационна или линеаризирана неявна диференчна схема с тегло $\sigma\in[0;\,1]$.

Итерационната неявна схема, както е казано в [3], води до по-малък обем на изчисленията. Скоростта ѝ на сходимост е $O\big(h^2+\tau\big)$ при $\sigma\neq 1/2$ и $O\big(h^2+\tau^2\big)$ при $\sigma=1/2$. Неин недостатък е удвояването на количеството заемана памет. Наистина — алгоритмичното изпълнение на един итерационен процес изисква съхраняване на информация за две последователни приближения $\overset{(s)}{y}$ и $\overset{(s+1)}{y}$, с помощта на която се проверява съответното условие за прекратяване на итерациите.

От линеаризираните схеми най-често се избира тази със $\sigma=1$ — тогава се получава триточковата чисто неявна схема

$$c(y^n)\rho\frac{y_i^{n+1}-y_i^n}{\tau}=\frac{1}{h}\left[\lambda(y^n)\frac{y_{i+1}^{n+1}-y_i^{n+1}}{h}-\lambda(y^n)\frac{y_i^{n+1}-y_{i-1}^{n+1}}{h}\right],$$

която е абсолютно устойчива и сходяща с ред на сходимост $O\big(h^2+\tau\big)$. Така например в [45] е разгледана едномерна двуфазна задача за кристализационния процес на стоманен слитък в цилиндрична кокила при различни технологични условия. Избраната диференчна схема е споменатата вече чисто неявна линеаризирана схема.

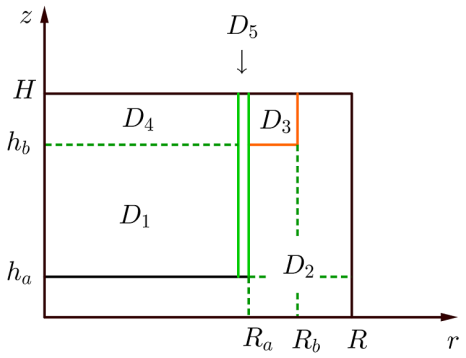
Обобщението на разглежданията до задача с размерност $p\geq 2$ води до *метода на сумарната апроксимация* (т. нар. локално едномерна схема или Л. Е. С.). В [3], [44] е показано, че този метод не зависи нито от броя на пространствените измерения, нито от геометричната форма на областта, нито от непрекъснатостта на коефициентите. Освен това той удовлетворява изключително важното изискване за икономичност по отношение на броя на аритметичните операции.

С метода на сумарната апроксимация се срещаме в трудовете [20], [27], [29], [41], [42], [43], [47], [78], където той е приложен за числено решаване на двумерна кристализационна задача в цилиндрична координатна система при наличие на фазово превръщане и топлообмен по закона на Стефан — Болцман. Линеаризираната Л. Е. С. е предпочетена в [43], [71], а итерационната — в [29], [41]. В [28] и [70] са разгледани тримерни задачи в декартови координати за многомерното уравнение на топлопроводността, които са решени с неявна линеаризирана Л. Е. С.

Тук предпочитанията към линеаризирана схема отдаваме на очевидния факт, че се решава голям брой нелинейни мрежови уравнения.

В [25] е извършено детайлно сравнение между метода на А. М. и метода на Х. К. Д. С., приложени върху най-простата едномерна двуфазна задача на Стефан (1.1) — (1.5). При метода на А. М. е използвана явна диференчна схема с условие за устойчивост, а при метода на Х. К. Д. С. — чисто неявна итерационна схема. Тъй като авторката е избрала топлофизичните характеристики да бъдат прекъснати функции константи, тя е успяла да намери и съответното автомоделно решение на задача (1.1) — (1.5). На база числени симулации е установено, че методът на А. М. генерира много по-добри резултати при локализирането на подвижната фазова граница в сравнение с метода на Х. К. Д. С., обаче значително усложнява числения алгоритъм и програмната реализация.

Реалните кристализационни задачи се отличават по своята трудност не само с наличието на подвижна фазова граница между агрегатните състояния на веществата, но и със специфична структура на избраната кокила. В металургичната практика е обичайно явление кокилите да са композирани от различни по брой и химически състав слоеве. Възприето е убеждението, че употребата на многослойни кокили води до получаването на слитъци с по-добро качество при различни технологични условия: добавяне на втулка и мъртва глава, нанасяне на обмазка, подгряване с екзотермични смеси и др. На фиг. 1.3 схематично е представена половината от осното сечение на цилиндрична кокила, принадлежаща към споменатия вид. Както може да се предположи, гранични условия



Фигура 1.3: *Схематично представяне на половината от осното сечение на многослойна кокила: D_1 — област, заета от металната сплав, D_2 — основна част на кокилата, D_3 — втулка, D_4 — мъртва глава, D_5 — тънко термично съпротивление (газова междина, обмазка)*

сега се поставят не само по външните граници на областта

$$D = D_1 \cup D_2 \cup D_3 \cup D_4 \cup D_5,$$

но и по всяка една от вътрешните граници $r = R_a$, $r = R_b$, $z = h_a$ и $z = h_b$. В литературата тези гранични условия са известни като *условия за спрягане*. Реалният им смисъл се изразява в наличието на контакт между две тела с различни начални температури и различни топлофизични характеристики. При апроксимацията им по метода на баланса (интегро-интерполационния метод) обикновено се препоръчва точката на контакт от пространствената мрежа да се заложи като „двойна“. Концепцията на този прийом е пределно ясна и се счита за общоприета. Намерила е израз в научните трудове [42], [43], [47], [48], [70], [71], където чрез построяване на чисто неявни Л. Е. С. върху неравномерни мрежи с „двойни“ точки е изследван процесът на затвърдяване в различни типове многослойни кокили.

Понякога в условията за спрягане се отчита не само молекулярната топлопроводност, която се подчинява на закона на Нютон, но и топлообменът чрез излъчване, който се осъществява по закона на Стефан — Болцман [45], [47], [71]. В този случай съответното контактно условие ще бъде нелинейно по отношение на неизвестната температура, тъй като топлинният поток q ще зависи по степенен закон от нея. Наистина — известно е, че след образуването на газова междина с ширина $\delta_{\text{м.}}$ стойността на потока q в междината между слитъка и кокилата се дава с израза (вж. фиг. 1.3)

$$q = \frac{\lambda_{\text{в.}}}{\delta_{\text{м.}}}\left(u\Big|_{r=R_a} - u_p\right) + \sigma_s\left[\left(u\Big|_{r=R_a}\right)^4 - u_p^4\right]. \tag{1.6}$$

Тук $\lambda_{\text{в.}}$ е коефициентът на топлопроводност на въздуха, u_p е температурата по вътрешната повърхност на слоя обмазка, а σ_s е константата на Стефан — Болцман. Ако представим второто събираемо в дясната страна на (1.6) по следния начин:

$$\left(u\Big|_{r=R_a}\right)^4 - u_p^4 = \left[\left(u\Big|_{r=R_a}\right)^2 + u_p^2\right]\left(u\Big|_{r=R_a} + u_p\right)\left(u\Big|_{r=R_a} - u_p\right),$$

то при построяването на линеаризирана чисто неявна диференчна схема можем да вземем изразите от първата и втората скоба на долния слой n по времето, а този от третата — на горния $n+1$. При итерационна схема изчисляваме изразите от всяка скоба на слоя $n+1$, като този от третата вземаме на $(s+1)$-вата итерация, а останалите — на s-тата. Този конкретен избор няма да се отрази на реда на точност.

И така, ние разгледахме важния клас физико-математически задачи, които отчитат промяната в агрегатните състояния на веществата. В синтезиран вид класифицирахме различните начини за приближеното решаване на тези задачи и детайлизирахме силните и слабите им страни. Направеният анализ създава обща представа за значимостта и широката приложимост на задачата на Стефан в различни аспекти на науката и експерименталните изследвания. Той не претендира за пълна изчерпателност на изложените факти, а следва да се приеме като база за осъществяване на бъдещи проучвания в областта на топлофизиката, термохимията, физикохимията и металургията чрез средствата на изчислителната математика и математическото моделиране.

Глава 2

Цели и задачи на дисертационния труд

В металургията и леярството изследването на кристализационните процеси е база за получаване на предварителна информация за отливките: температурно разпределение в обема на отливката, фазови криви, области с дефекти в микроструктурата, размери на зърната и др. Само по себе си едно чисто експериментално изследване изисква подходящи лабораторни условия и съответна апаратура, разходи за електроенергия, суровини, материали и т. н. Това силно ограничава осъществяването на регулярни и задълбочени анализи.

С течение на годините в практиката се внедри един своеобразен аналог на реалния експеримент — числената симулация въз основа на математически модел. Като се спазват фундаменталните природни закони, в научните среди се конструират специални математически модели, които са отражение на конкретна ситуация от действителността. Примери за последното са процесите на делене на клетки, разпадането на радиоактивните химични елементи, движението на небесните тела по техните стационарни орбити, взаимодействието между две популации от типа хищник — жертва (т. нар. модели на Лотка — Волтера), топлопреносът между две тела, осъществяващи идеален контакт, колебателните движения на струна и др. Числените симулации се извършват изцяло в интегрирана софтуерна среда, не са притиснати от финансови и времеви рамки и допускат многократно изпълнение при различни параметри на модела. След серия от опити се установяват определени числови оценки. Естествено тези оценки са само приближени, но ако съответният числен метод е сходящ и с висока степен на точност, без ограничение на общността може да се приеме, че резултатите от него са достоверни. Доколко тези резултати отговарят на очакванията, зависи от общоизвестни науч-

ни факти, фундаментални природни принципи или данни, събрани по експериментален път.

Сравнението с експериментални данни по подразбиране е един от най-сигурните качествени признаци за това, дали даден математически модел е адекватен, и ако да, то до каква степен. Позовавайки се на научните изследвания в областта на кристализацията, можем да твърдим, че даден модел се счита за добър, ако неговото сравнение с реалния експеримент дава относително отклонение, непревишаващо 10%. Разбира се, този консенсус между теория и практика относно акуратността на модела не е валиден, изобщо казано, по индукция и за останалите науки. Възможно е там да са приети други критерии, според които да се съди и да се оформят изводи.

Да преминем сега към синтезирано представяне на конкретните цели и задачи, които си поставяме в настоящия дисертационен труд, както и на съответната методология, с която ще ги постигнем.

I. Обект на изследването

Дисертационният труд е посветен на численото изследване на кристализационния процес при металните сплави. В математическата постановка ние ще разгледаме два модела.

1. Едномерен математически модел за кристализация на метална сплав с отчитане на центровете на кристализация.

2. Двумерен математически модел за кристализация на метална сплав, отлята в цилиндрична форма.

II. Цели на изследването

В рамките на настоящия дисертационен труд ще бъдат реализирани следните фундаментални цели.

1. Разработване на компютърен алгоритъм за числено симулиране на кристализацията на споменатите сплави чрез използване на едномерния математически модел и с отчитане центровете на кристализация.

2. Разработване на компютърен алгоритъм за числено симулиране на кристализацията чрез използване на двумерния математически модел и с отчитане топлината на кристализация на сплавите.

3. Проектиране, реализиране и тестване на собствен графичен потребителски интерфейс за решаване на двумерната кристализационна задача в диалогов режим.

4. Числено изследване на кристализационния процес с използване на получените числени алгоритми.

III. Задачи, които трябва да се решат, за да се постигнат целите на изследването

Численото изследване на даден математически модел изисква предварителна информация за аналитичното решение, отговарящо на зададените начални и гранични условия. В болшинството от случаите това решение или не се изразява в квадратури, или намирането му с познатите алгебрични техники е невъзможно. Изборът на адекватен числен метод и непосредственото му прилагане преодоляват тази трудност. За постигането на желаните резултати формулираме следните основни задачи и подзадачи.

■ **Едномерен математически модел**

1. *Конструиране на диференчна схема*:
   (а) дискретизация на областта, избор на мрежа;
   (б) построяване на явна диференчна схема, реализираща метода на Ойлер;
   (в) оценка на грешката от апроксимация.
2. *Оценка за влиянието на броя на кристализационните центрове върху преохлаждането.*
3. *Получаване на числени резултати за температурата като функция на времето.*
4. *Представяне на резултатите в графичен вид.*

■ **Двумерен математически модел**

1. *Конструиране на диференчна схема*:
   (а) избор на пространствена и времева мрежа;
   (б) построяване по метода на баланса на линеаризирана чисто неявна локално едномерна схема;
   (в) пресмятане на нелинейните коефициенти на диференчната схема в зависимост от температурата;
   (г) решаване на получената тридиагонална система линейни алгебрични уравнения с икономичен директен метод;
   (д) оценка на грешката от апроксимация;

(е) конструиране на тестови примери, едномерни по всяко от пространствените направления;

(ж) сравнения с така конструираните тестови примери и доказване реда на сходимост и коректността на диференчната схема по правилото на Рунге;

(з) изготвяне на графични резултати.

2. *Оценка на параметрите*:

   (а) избор на базов модел и на потенциални за изследването параметри;

   (б) съставяне на разчетни таблици за степента на влияние на всеки параметър;

   (в) систематизиране на резултатите за оценките на параметрите с най-слабо и най-силно влияние спрямо базовия модел.

3. *Получаване на числени резултати за температурата като функция на времето.*

4. *Представяне на резултатите в графичен вид.*

5. *Конструиране на графичен потребителски интерфейс в средата на софтуерната платформа* ***MATLAB***.

Глава 3

Едномерен математически модел за кристализация на метални сплави

3.1 Постановка на задачата

Кристализацията е процес на формиране и растеж на кристали, които са стопилки от чисти метали, сплави, течности или газови фази. Преходът от течно към твърдо агрегатно състояние се нарича първична кристализация. В условия на равновесие кристализацията се осъществява при постоянна температура или в температурен интервал. При чистите метали е възможен само първият случай, докато при многокомпонентните сплави — и двата.

Процесът започва с отливането на стопилката във формата. Началната температура u_0 на стопилката зависи естествено от нейния състав, от геометрията и размерите на формата. По време на кристализационния процес в отливката съществуват три зони: течна зона, двуфазна зона и зона на твърдата фаза [14], [69].

Течната зона в дадена точка има температура, изменяща се във времето от температурата на заливане u_0 до температурата на ликвидуса u_L. Известно е, че кристализацията не започва точно при достигане на u_L, а при друга стойност u, по-ниска от нея. Величината $\Delta u = u_L - u$ се нарича преохлаждане.

Двуфазната зона представлява слоят в обема на сплавта, за който температурата се изменя в интервала между температурата на ликвидуса u_L и температурата на евтектиката u_E. При температури, по-ниски от u_E, съществува само твърда фаза.

Разглежданият процес не протича линейно. Това се дължи на сложното температурно изменение в двуфазната зона. Можем да считаме, че единствено топлообменът в течната фаза има линеен характер.

Да предположим, че имаме достатъчно малък обем на стопилката, за който температурата u зависи само от времето t. При неизотермична обемна кристализация условието за баланс на топлината се записва във вида

$$c\rho V_0 \frac{\mathrm{d}\,u}{\mathrm{d}\,t} = \kappa\rho \frac{\mathrm{d}\,V}{\mathrm{d}\,t} - \alpha F\,(u - u_F)\,, \tag{3.1}$$

където c е специфичният топлинен капацитет, ρ е относителната плътност, V_0 е обемът на отливката, κ е топлината на кристализация, V е обемът на твърдата фаза в отливката, α е коефициент на топлообмен, F е площта на отливката и u_F е температурата на формата.

Съгласно статистическата теория за обема на кристализация при металите, създадена от Колмогоров [53], обемът V може да бъде дефиниран в зависимост от скоростта на растеж на кристалите така:

$$V = V_0\left(1 - e^{-\omega}\right). \tag{3.2}$$

Тук величината ω се пресмята по формулата

$$\omega = \varphi N \left(\int_{t_L}^{t} K_V \Delta u \,\mathrm{d}\,s \right)^3, \tag{3.3}$$

където φ е геометричен параметър, който за сферични кристали има стойност $\varphi = \dfrac{4\pi}{3}$, N е броят на центровете на кристализация, а K_V е кристализационна константа, предварително известна. Както е посочено в [35], в случая на двукомпонентна сплав величината на преохлаждането се определя така:

$$\Delta u = u_L - u = u_A - u - \beta_0 c_0 f_L^{k-1}. \tag{3.4}$$

Тук u_A е температурата на топене на чистия метал, β_0 е модулът на наклона на линията на ликвидуса върху диаграмата на състоянията, c_0 е концентрацията на легиращия елемент в сплавта, $f_L = e^{-\omega}$ е съдържанието на течната фаза в двуфазната зона, а k е коефициент на разпределение. От равенството (3.2) имаме

$$\frac{V}{V_0} = 1 - f_L = 1 - e^{-\omega}. \tag{3.5}$$

Накрая, използвайки зависимостите (3.1) — (3.5), достигаме до нелинейното уравнение

$$\frac{\mathrm{d}\,\Delta u}{\mathrm{d}\,t} = \frac{\alpha}{R\rho_i c_i}\left[u_A - u_F - \Delta u - \beta_0 c_0 f_L^{k-1}\right] - \left[\frac{\kappa}{c_i} + \beta_0 c_0 (1-k) f_L^{k-2}\right]\frac{\mathrm{d}\,f_L}{\mathrm{d}\,t}. \quad (3.6)$$

Тук $R = \dfrac{V_0}{F}$, а индексът i приема стойностите 1, 2, 3, като при това $i = 1$ отговаря на течната фаза, $i = 2$ — на двуфазната зона и $i = 3$ — на твърдата фаза.

Вече сме готови да формулираме математическия модел, който описва кристализационния процес във всяка от споменатите три зони [14], [15], [55], [59]. Навсякъде по-надолу в тази глава с t_L и t_E сме означили моментите от време, за които в обема на отливката се достигат съответно температурите u_L и u_E.

Охлаждането на сплавта в **течната фаза** започва при начална температура u_0 и приключва при достигане на температурата на ликвидуса u_L. Да означим началния момент от време с t_0, по традиция $t_0 = 0$ s. Температурната зависимост от времето в интервала $[t_0;\, t_L]$ се описва от линейното диференциално уравнение

$$\frac{\mathrm{d}\,u}{\mathrm{d}\,t} = -\frac{\alpha_1}{R\rho_1 c_1}(u - u_F) \quad (3.7)$$

с начално условие

$$u\Big|_{t=t_0} = u_0. \quad (3.8)$$

По-нататък, в **двуфазната зона** температурата се понижава, настъпва охлаждане, отделя се скритата топлина на кристализация и в стопилката се появяват първите кристали на твърдата фаза. В интервала $[t_L;\, t_E]$ е в сила едномерното уравнение

$$\frac{\mathrm{d}\,\Delta u}{\mathrm{d}\,t} = \frac{\alpha_2}{R\rho_2 c_2}\left[u_A - u_F - \Delta u - \beta_0 c_0 (e^{-\omega})^{k-1}\right] - \left[\frac{\kappa}{c_2} + \beta_0 c_0 (1-k)(e^{-\omega})^{k-2}\right]\cdot e^{-\omega}\cdot\frac{\mathrm{d}\,\omega}{\mathrm{d}\,t}. \quad (3.9)$$

Началното условие е

$$\Delta u\Big|_{t=t_L} = u_A - u_L - \beta_0 c_0, \quad (3.10)$$

където сме взели предвид, че от равенството (3.3) при $t = t_L$ имаме $\omega = 0$ и съответно $f_L = 1$.

След достигане на температурата на евтектиката u_E се осъществява кристализация на евтектичната стопилка при постоянна температура u_E. Разпределението на температурата в обема на отливката се определя от уравнението

$$c_3\rho_3\frac{\mathrm{d}\,u}{\mathrm{d}\,t} = -\kappa\rho_3\frac{\mathrm{d}\,f_E}{\mathrm{d}\,t} - \frac{\alpha_3}{R}(u - u_F), \tag{3.11}$$

чието начално условие е

$$u\Big|_{t=t_E} = u_E. \tag{3.12}$$

Тук $f_E = e^{-sK_E}$, където $s = \int\limits_{t_E}^{t}(u_E - u)\,\mathrm{d}\,t$, е количеството на течната фаза в зоната на евтектиката, а K_E е кристализационен коефициент на евтектиката. Точките t_L и t_E са очевидно точки на непрекъснатост, защото там температурите в две съседни фази съвпадат. Тези точки не са предварително известни и се намират в хода на изчислителната схема.

Ще извършим числени симулации за две различни сплави: алуминиева сплав **AlSi7Mg**, съдържаща 7 wt% силиций **Si** (**A356**), и сив чугун, съдържащ 3,3 wt% въглерод **C**. Стойностите на физичните параметри за споменатите сплави са нагледно представени в таблица 3.1 [23].

3.2 Диференчна схема и числени симулации

Да се заемем сега с решаването на едномерните задачи (3.7), (3.8); (3.9), (3.10) и (3.11), (3.12) в случая на сплавта **AlSi7Mg**. Както беше казано по-горе, за да различаваме получените решения във всяка от фазите, ще използваме индекси. Нека $u^{(i)}(t)$, $i = 1, 2, 3$, е търсената температура в i-тата фаза. При $i = 2$ поради равенството (3.4) имаме $u^{(2)}(t) = u_A - \beta_0 c_0 f_L^{k-1} - \Delta u$. Тогава, ако зависимостта $u = u(t)$ дефинира разпределението на температурата за целия интервал от време, в който протича кристализационният процес, то можем да запишем:

$$u(t) = \begin{cases} u^{(1)}(t), & t_0 \le t \le t_L, \\ u^{(2)}(t), & t_L \le t \le t_E, \\ u^{(3)}(t), & t_E \le t \le T. \end{cases} \tag{3.13}$$

Параметър	AlSi7Mg	Сив чугун
u_0, [°C]	660	1538
u_L, [°C]	613.70	1251.85
u_E, [°C]	583.85	1139.85
u_F, [°C]	19.85	19.85
u_A, [°C]	660	1538
α_1, [W/(m^2.° C)]	48.00	80.00
α_2, [W/(m^2.° C)]	75.00	80.00
α_3, [W/(m^2.° C)]	85.00	80.00
ρ_1, [kg/m^3]	2600.00	7000.00
ρ_2, [kg/m^3]	2450.00	7050.00
ρ_3, [kg/m^3]	2450.00	7100.00
c_1, [J/(kg.° C)]	1000.00	680.00
c_2, [J/(kg.° C)]	1050.00	570.00
c_3, [J/(kg.° C)]	1050.00	460.00
R, [m]	0.0071	0.0071
κ, [J/kg]	$4.02 \cdot 10^5$	$1.38 \cdot 10^5$
k, [ ]	0.14	0.78
K_E, [(°C. s)$^{-1}$]	0.005	0.005
K_V, [m/(°C. s)]	$1.515 \cdot 10^{-4}$	$3.0 \cdot 10^{-4}$
$\beta_0 c_0$, [°C]	43.5	302.61

Таблица 3.1: *Стойности на параметрите на модела за два вида сплави: алуминиева сплав **AlSi7Mg** [82] и сив чугун [23], [83]*

Понеже $u^{(1)}\Big|_{t=t_L} = u^{(2)}\Big|_{t=t_L} = u_L$ и $u^{(2)}\Big|_{t=t_E} = u^{(3)}\Big|_{t=t_E} = u_E$, то функцията $u(t)$, дефинирана с равенствата (3.13), е непрекъсната за всяко $t \in [t_0; T]$.

Лесно се забелязва, че в **течната фаза** диференциалното уравнение (3.7) е линейно с постоянни коефициенти. Решавме го при условието (3.12) и веднага получаваме явната формула

$$u^{(1)}(t) = u_F + (u_0 - u_F)e^{\dfrac{\alpha_1(t_0 - t)}{R\rho_1 c_1}}. \tag{3.14}$$

Тъй като функцията, определена от формулата (3.14), е непрекъсната, то $\lim\limits_{t \to t_L} u^{(1)}(t) = u^{(1)}(t_L) = u_L$. От това равенство след логаритмуване пресмятаме стойността t_L, в която по-късно ще поставим началното условие (3.10). Намираме $t_L = 67.9118$ s.

Двуфазната зона е в температурния интервал $u_L \geq u^{(2)}(t) \geq u_E$, или, което е същото, при $t \in [t_L;\, t_E]$. Тъй като моментът от време t_E не е предварително известен, можем да предполагаме, че решаваме началната задача (3.9), (3.10) в достатъчно широк интервал $[t_L;\, t^*]$, $t^* > t_E$. Без ограничение полагаме $t^* = 200$ s.

Диференциалното уравнение (3.9) е нелинейно относно функцията Δu. В интервала $[t_L;\, t^*]$ въвеждаме равномерна мрежа от точки със стъпка τ:

$$\overline{\omega}_\tau = \left\{ t_n :\, t_0 = t_L,\, t_J = t^*,\, \tau = \frac{t^* - t_L}{J};\, t_{n+1} = t_n + \tau,\, n = 0,\, 1,\, \ldots,\, J-1 \right\}.$$

Върху тази мрежа заменяме диференциалната задача (3.9), (3.10) с диференчна, като означаваме мрежовия аналог на Δu с y, и прилагаме явния метод на Ойлер:

$$\begin{aligned} y_0 &= u_A - u_L - \beta_0 c_0, \\ y_{n+1} &= y_n + \frac{\tau\alpha_2}{R\rho_2 c_2}\left[u_A - u_F - y_n - \beta_0 c_0 (e^{-\omega_n})^{k-1}\right] - \\ &\quad \tau\left[\frac{\kappa}{c_2} + \beta_0 c_0 (1-k)(e^{-\omega_n})^{k-2}\right] \cdot e^{-\omega_n} \cdot \frac{\omega_n - \omega_{n-1}}{\tau}, \\ &\qquad\qquad n = 1,\, 2,\, \ldots,\, J-1. \end{aligned} \quad (3.15)$$

В случая за определянето на y_2 са ни необходими стойностите y_0 и y_1. Очевидно, тъй като

$$\frac{\mathrm{d}\,\omega}{\mathrm{d}\,t} = 3\varphi N \left(\int_{t_L}^{t} K_V \Delta u \,\mathrm{d}\,s\right)^2 \cdot K_V \Delta u,$$

то

$$\left.\frac{\mathrm{d}\,\omega}{\mathrm{d}\,t}\right|_{t=t_L} = 0.$$

Оттук пресмятаме

$$y_1 = y_0 + \frac{\tau\alpha_2}{R\rho_2 c_2}\left[u_A - u_F - y_0 - \beta_0 c_0 (e^{-\omega_0})^{k-1}\right].$$

Получихме дискретната стойност y_1. Връщаме се в диференчния метод (3.15) и го прилагаме за $n = 1,\, 2,\, \ldots,\, J-1$. Сега вече мрежовата функция y е известна във всички възли на мрежата. От равенството (3.4) веднага намираме и решението $u^{(2)}(t)$ с грешка на приближението $O(\tau)$:

$$u^{(2)}(t) \approx u_A - \beta_0 c_0 f_L^{k-1} - y.$$

Остана да преценим за коя точка t_n от мрежата $\overline{\omega}_\tau$ е изпълнено неравенството $\left|u^{(2)}(t)-u_E\right| < \varepsilon$, където ε е достатъчно малко положително число (ние сме избрали $\varepsilon = 10^{-4}$). С итериране по дискретните координати на вектора $u^{(2)}(t)$ лесно установяваме, че $t_E = 177.0808$ s.

Накрая нека да намерим и приближение за аналитичното решение $u^{(3)}(t)$ в **зоната на евтектиката**. Това е зоната, в която кристализационният процес завършва. Времето за окончателно приключване на затвърдяването е $T = 300$ s, затова ще решаваме диференциалното уравнение (3.11) с начално условие (3.12) в интервала $t \in [t_E; T]$. Както беше и в случая на двуфазна зона, то е нелинейно по отношение на търсената температура u. За приближеното му решаване нека отново да използваме явния метод на Ойлер върху равномерната мрежа

$$\overline{\omega}'_\tau = \left\{ t_n : t_0 = t_E,\, t_{J'} = T,\, \tau = \frac{T - t_E}{J'};\, t_{n+1} = t_n + \tau,\, n = 0,\, 1,\, \ldots,\, J' - 1 \right\}.$$

Въвеждаме функцията y — мрежовия аналог на $u^{(3)}(t)$ върху мрежата $\overline{\omega}'_\tau$. Явната диференчна схема, съответна на задачата (3.11), (3.12), след необходимите преобразувания изглежда така:

$$\begin{aligned} &y_0 = u_E, \\ &\frac{y_{n+1} - y_n}{\tau} = \frac{\kappa K_E}{c_3}(u_E - y_n)e^{-K_E s_n} - \frac{\alpha_3}{R\rho_3 c_3}(y_n - u_F),\ n = 0,\, 1,\, \ldots,\, J' - 1. \end{aligned} \tag{3.16}$$

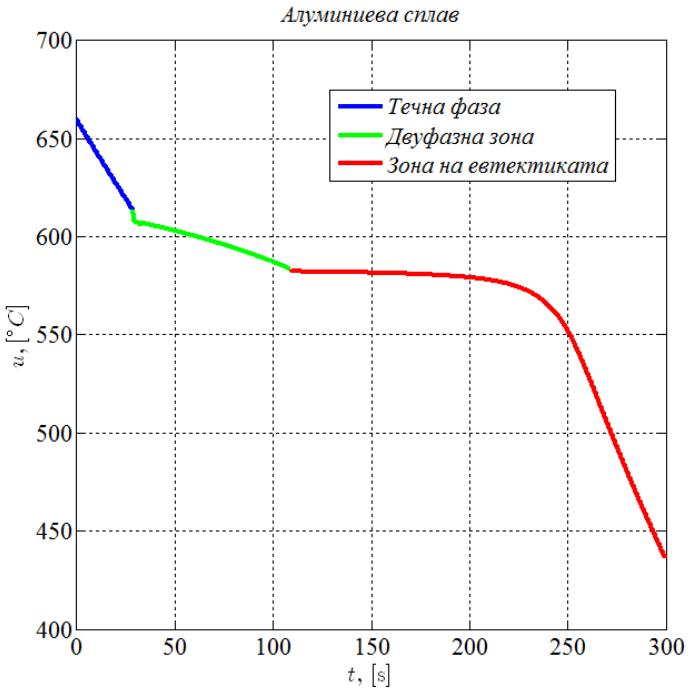
Тук сме означили за краткост $s_n = \int\limits_{t_E}^{t_n} (u_E - u)\,\mathrm{d}\,t < \infty$. Нека да представим s_n във вида

$$s_n = \sum_{k=0}^{n-1} \int\limits_{t_k}^{t_{k+1}} (u_E - u)\,\mathrm{d}\,t.$$

По този начин реалното число s_n, което е стойност на определен интеграл, се представя като крайна сума на n интеграла, чиито граници са последователни точки от избраната равномерна мрежа. За пресмятането на единичните интеграли прилагаме формулата на правоъгълниците.

И така, аналитичното решение $u^{(1)}(t)$ на задачата за топлообмен в течната фаза (3.7), (3.8) се дава със зависимостта (3.14), а за задачите в двуфазната зона (3.9), (3.10) и в зоната на евтектиката (3.11), (3.12) такива решения не са известни, но са намерени приближени аналози.

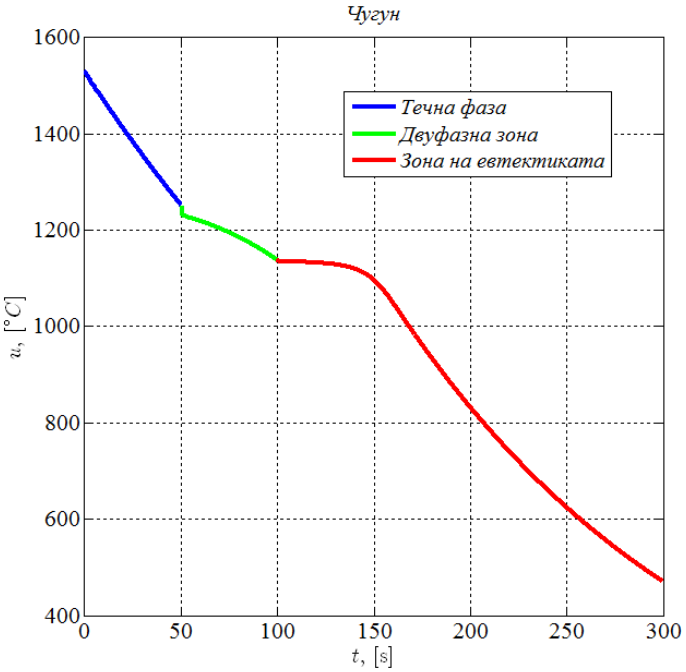
При това единственото решение на диференчната схема (3.15) е сходящо към истинското решение Δu — оттук веднага получаваме $u^{(2)}(t)$, — а на схемата (3.16) — към $u^{(3)}(t)$, с една и съща скорост на сходимост $O(\tau)$.



Фигура 3.1: *Разпределение на температурата като функция на времето, алуминиева сплав* ***AlSi7Mg***

На фиг. 3.1 графично е представено изменението на температурата в зависимост от времето за двукомпонентната сплав **AlSi7Mg**. С различни цветове са обозначени трите зони, в които протича процесът на кристализацията.

Същото изследване на температурното поле е извършено и върху сплавта на желязна основа, а именно сивия чугун. Числената симулация е осъществена при конкретните стойности на параметрите, представени в третия стълб на таблица 3.1. Моментите от време, за които се достигат температурите u_L и u_E, са съответно $t_L = 50.3650$ s и $t_E = 99.7860$ s. Общият вид на температурната зависимост за всички етапи на кристализационния процес е показан на фиг. 3.2.

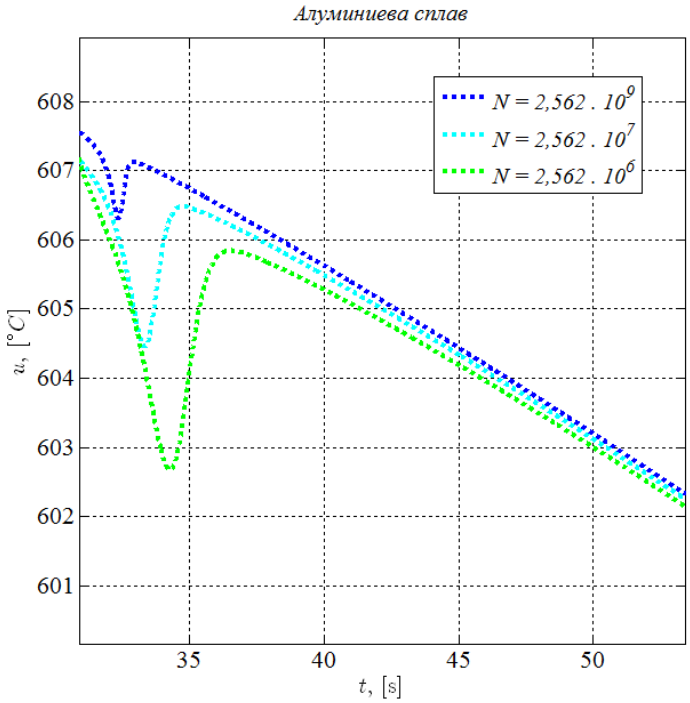


Фигура 3.2: *Разпределение на температурата като функция на времето, сив чугун*

3.2.1 Влияние на центровете на кристализация върху процеса на затвърдяване

От металните сплави, които се използват в практиката, се изисква да притежават добър комплекс от свойства — якост, пластичност, жилавост, износоустойчивост и др. За подобряването им в последно време се използва легиране с подходящи химични елементи или добавяне на т. нар. *частици с наноразмери* [11], [12], [61], [62], [77]. Частиците с наноразмери представляват пренебрежимо количество (части от процента) от дадено химично съединение, което се прибавя към течната сплав и подобрява нейните механични свойства. Обикновено тези частици се получават по плазмено-химичен метод [15], [18]. Металурзите и инженерите избират такива вещества, които са труднотопими, например карбиди, нитриди или карбонитриди [15], [62] (силициев карбид **SiC**, титаниев карбид **TiC**, титаниев нитрид **TiN**, танталов нитрид **TaN**, титаниев карбонитрид **TiCN** и др.).

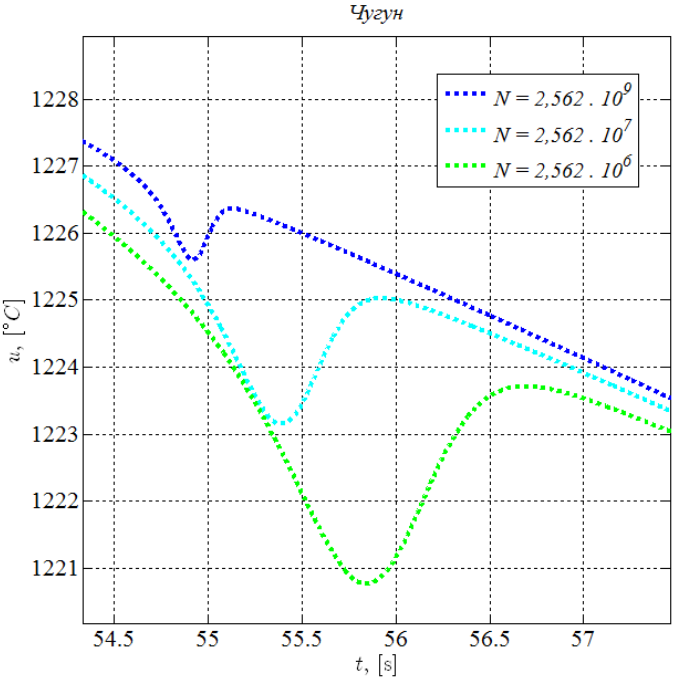
Размерът на микрозърната в кристализиралата сплав е качествен признак за свойствата на получената фасонна отливка. Експериментално е установено, че съдържанието на нанопрахове, покрити с желязо



Фигура 3.3: *Влияние на броя на кристализационните центрове върху преохлаждането, алуминиева сплав* ***AlSi7Mg***

Fe или хром **Cr** и добавени към стопилката, подобрява някои от споменатите свойства чрез издребняване на микроструктурата [15]. Ще отбележим, че увеличението на съдържанието на модифициращия прах не води до рязко и значимо изфиняване на структурата. Реалните опити показват, че все пак малки количества наномодификатори от порядъка на $0.015 - 0.025\,\mathrm{wt}\,\%$ са достатъчни за установяването на осезаема разлика в механичните свойства на отливките. Чрез технологични експерименти с алуминиева сплав, проведени в лабораториите на Института по металознание, съоръжения и технологии към БАН, е установено, че броят на кристализационните центрове е $2.562 \cdot 10^9\ \mathrm{m}^{-3}$ за модифицираната и $3.3 \cdot 10^8\ \mathrm{m}^{-3}$ за немодифицираната сплав.

Следвайки идеите, изложени в [15], да извършим няколко числени симулации за разгледаните по-горе сплави **AlSi7Mg** и сив чугун. Основната ни цел е да оценим в каква степен промяната в количеството N на наномодифициращите частици влияе върху микроструктурата на съответната отливка. За чугуна избраните модифициращи прахове са **TiCN** и $\mathbf{Y_2O_3}$, покрити с **Cr** и **Fe**, а за **AlSi7Mg** — силициев карбид **SiC**, покрит с мед **Cu**. Нека в качеството на симулационни стойности да



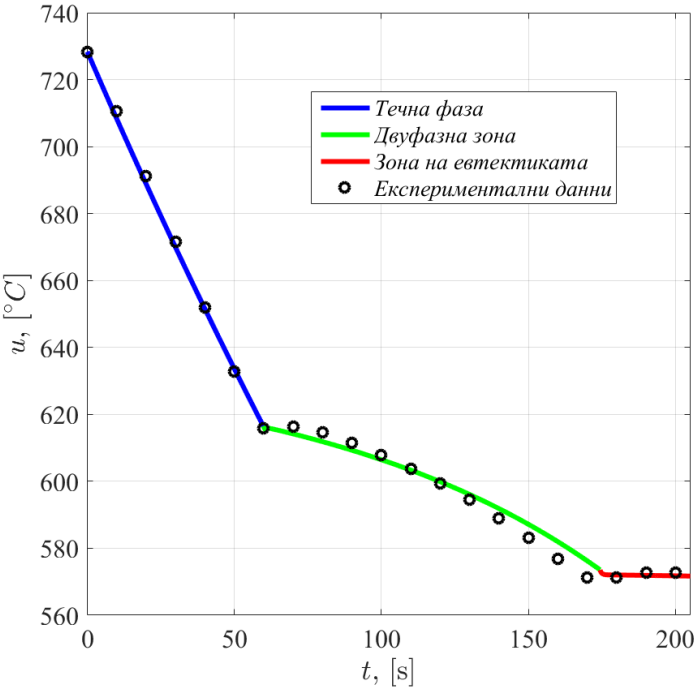
Фигура 3.4: *Влияние на броя на кристализационните центрове върху преохлаждането, сив чугун*

фиксираме последователно

$$N = \left\{2.562 \cdot 10^9;\ \ 2.562 \cdot 10^7;\ \ 2.562 \cdot 10^6\right\},\ \ [N] = \mathrm{m}^{-3},$$

първо за алуминиевата, а след това — за желязната сплав. Резултатите от симулациите са начертани с различни цветове на фиг. 3.3 и фиг. 3.4. Мащабът е подбран така, че на преден план да бъде преохлаждането Δu (вж. формула 3.4); графично то съответства на точка на локален минимум. Лесно се съобразява, че намаляването на броя наночастици (т. нар. *активни центрове на кристализация*) увеличава преохлаждането Δu, изместващи минимума надолу и надясно. Вследствие на това микроструктурата на образеца става по-груба, т. е. кристалите са с по-голям диаметър, което неминуемо е предпоставка за дефекти.

Както е посочено в [15], образците от чугунена стопилка, съдържащи 0.015 wt % и 0.025 wt % модификатор $\mathbf{TiN{+}Y_2O_3}$, имат по-фини и по-равномерно разпределени графитни пластини в сравнение с тези без съответния наномодификатор. Нещо повече — увеличаването на процентното съдържание на $\mathbf{TiN{+}Y_2O_3}$ до 0.040 wt % не води до издребняване на структурата и следователно не подобрява механичните показатели.



Фигура 3.5: *Валидиране на едномерния математически модел* (3.7) — (3.12) *въз основа на експериментални данни за алуминиева сплав* ***AlSi7Mg***

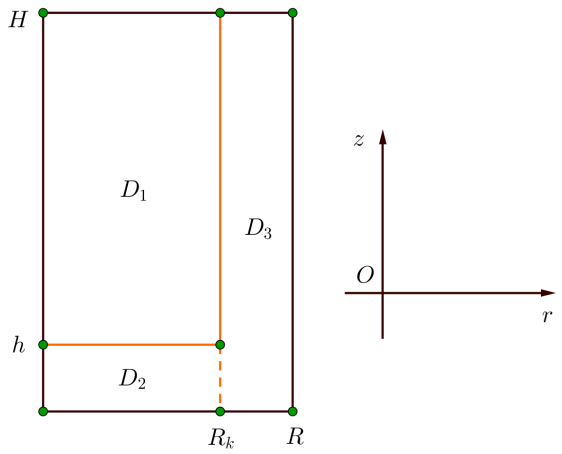
Вземайки предвид данните от фиг. 3.3 и 3.4, можем да предположим, че същото заключение се отнася и за алуминиевата сплав **AlSi7Mg**. Нещо повече, оказва се, че след модифициране на **AlSi7Mg** с нанопрах на **TiN** средният диаметър на кристалите намалява с 50% [66].

Валидирането на математическия модел (3.7) — (3.12) е извършено на базата на експериментални данни за проба на **AlSi7Mg** [82]. От фиг. 3.5, добиваме визуална представа за доброто съвпадение между резултатите от лабораторното измерване и тези от числената симулация. Към момента на написването на настоящия дисертационен труд не разполагаме с налични експериментални данни за чугун, но наблюдавайки фиг. 3.4, можем да предположим, че изменението в броя на активните кристализационни центрове ще окаже осезаемо влияние върху преохлаждането.

Глава 4

Двумерен математически модел за кристализация на метална сплав

4.1 Постановка на задачата



Фигура 4.1: *Осно сечение на цилиндрична форма*

Да разгледаме сега двумерната нестационарна задача за кристализация на бинарна метална сплав в цилиндрична форма. На фиг. 4.1 е показана половината от осното сечение на тази форма. С D_1 сме означили областта, заета от разтопената сплав, а с D_2 — дъното на експерименталната форма. То е композирано от плакиран пясък и служи като изолация срещу топлоотделянето към околната среда. Накрая с D_3 сме

означили областта на формата. В конкретния случай тази област представлява тънък слой стомана, през който се отдава топлина към околната повърхнина. Геометричните размери са означени на чертежа буквено, а конкретните им стойности са дадени по-нататък в изложението на тази глава.

В съответствие с гореизложеното нека да въведем следните означения:

$$\overline{D}_1 = \Big\{(r;\, z) :\, 0 \le r \le R_k,\, h \le z \le H\Big\},$$
$$\overline{D}_2 = \Big\{(r;\, z) :\, 0 \le r \le R_k,\, 0 \le z \le h\Big\},$$
$$\overline{D}_3 = \Big\{(r;\, z) :\, R_k \le r \le R,\, 0 \le z \le H\Big\}.$$

Металната сплав, плакираният пясък и стоманената форма се характеризират с определена температура u, коефициент на топлоемкост c, коефициент на топлопроводност λ и плътност ρ. Навсякъде по-надолу с индекс $k \in \{1;\, 2;\, 3\}$ ще означаваме топлофизичните характеристики на съответната област D_k. В общия случай те са прекъснати функции от температурата и този факт записваме така:

$$c,\, \rho,\, \lambda = \begin{cases} c_1(u),\, \rho_1,\, \lambda_1, & (r;\, z) \in \overline{D}_1, \\ c_2,\, \rho_2,\, \lambda_2, & (r;\, z) \in \overline{D}_2, \\ c_3,\, \rho_3,\, \lambda_3, & (r;\, z) \in \overline{D}_3. \end{cases}$$

Разпределението на температурата навсякъде в сечението $\overline{D} = \overline{D}_1 \cup \overline{D}_2 \cup \overline{D}_3$ ще считаме функция на пространствените координати и на времето: $u = u(r, z, t)$. Освен това поради прекъснатостта на топлофизичните характеристики и различната степен на охлаждане във всяка подобласт D_k, $k = 1,\, 2,\, 3$, функцията u също ще бъде прекъсната:

$$u(r, z, t) = \begin{cases} u^{(1)}(r, z, t), & (r;\, z) \in \overline{D}_1, \\ u^{(2)}(r, z, t), & (r;\, z) \in \overline{D}_2, \\ u^{(3)}(r, z, t), & (r;\, z) \in \overline{D}_3. \end{cases}$$

В момента на отливане във формата разтопената сплав притежава начална температура $u_{0,\text{м.}}(r, z)$, а формата — начална температура $u_{0,\text{ф.}}(r, z)$. С течение на времето температурата на метала вследствие на охлаждането се понижава до температурата на ликвидуса $u = u_L$. От тази точка нататък до температурата на солидуса $u = u_S$ процесът на охлаждане продължава в двуфазната зона $u_S \le u \le u_L$ [47], [48]. На

този етап във формата съществуват едновременно течна сплав и твърди кристали. Контурът на фазовия преход е върху линията $u = u_S$. Това всъщност е една подвижна граница между двете основни фазови състояния за всеки конкретен момент от време [46], [47]. Наличието на фазова граница се моделира чрез въвеждане на ефективната топлоемкост [4], [20] по следния начин:

$$c(u) = \begin{cases} c_L, & u > u_L, \\ c_S - \kappa \dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}, & u_S \le u \le u_L, \\ c_S, & u < u_S. \end{cases} \tag{4.1}$$

Тук с κ е означена скритата топлина на кристализация, която се отделя в двуфазната зона. Величината $\psi(u)$ е относителният дял на твърдата фаза в обема на отливката (вж. секция 4.2), а c_L и c_S са коефициентите на топлоемкост на сплавта съответно в течно и в твърдо агрегатно състояние. Ние ще предполагаме, че $c_L = c_S$.

Във всички вътрешни точки на формата е в сила уравнението [28], [29], [71]

$$c(u)\rho\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(u)\frac{\partial u}{\partial r}\right) + \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial z}\right). \tag{4.2}$$

При $t = 0$ задаваме прекъснатото начално условие

$$u(r, z, 0) = \begin{cases} u_{0,\text{м.}}, & (r;\, z) \in \overline{D}_1, \\ u_{0,\text{ф.}}, & (r;\, z) \in \overline{D}_2 \cup \overline{D}_3. \end{cases} \tag{4.3}$$

На дъното на формата задаваме условието

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=0} = \widetilde{\alpha}\left(u\Big|_{z=0} - u_{\text{ок. ср.}}\right), \tag{4.4}$$

което представлява топлообмен по закона на Нютон с коефициент на конвективен топлообмен $\widetilde{\alpha}$. Тук $u_{\text{ок. ср.}}$ е температурата на околната среда. По-нататък: на горната основа на цилиндъра отново задаваме топлообмен по закона на Нютон с коефициент $\overline{\alpha}$:

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=H} = -\overline{\alpha}\left(u\Big|_{z=H} - u_{\text{ок. ср.}}\right). \tag{4.5}$$

Върху границата $z = h$ задаваме контактно условие от типа

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^-} = \lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^+} = \alpha_k \left(u\Big|_{z=h^+} - u\Big|_{z=h^-} \right). \tag{4.6}$$

Върху оста на цилиндъра $r = 0$ поставяме обичайното условие за симетрия

$$\lim_{r\to 0} r\lambda(u)\frac{\partial u}{\partial r} = 0, \tag{4.7}$$

а върху околната повърхнина $r = R$ — за топлообмен по закона на Нютон с коефициент α:

$$\lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R} = -\alpha \left(u\Big|_{r=R} - u_{\text{ок. ср.}} \right). \tag{4.8}$$

По вертикалната граница $r = R_k$ поставяме условието

$$\lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R_k^-} = \lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R_k^+} = \alpha_k^{(1)} \left(u\Big|_{r=R_k^+} - u\Big|_{r=R_k^-} \right). \tag{4.9}$$

Коефициентът на контактен топлообмен $\alpha_k^{(1)}$ е постоянен в дъното (при $0 \le z \le h$) със стойност 10^6 W/(m^2.$^\circ$C), което на практика имитира „идеален“ контакт. С течение на времето фазовата крива $u = u_S$ ще се премества все по-навътре към оста на цилиндъра. Между отливката и формата се формира газова междина с ширина δ_M при дебелина на твърдата фаза δ_k. Така по границата $r = R_k$ при $h < z \le H$ коефициентът $\alpha_k^{(1)}$ ще има различна стойност. При отнапред зададена дебелина на твърдата фаза δ, която се определя индивидуално за всеки конкретен реален експеримент, коефициента $\alpha_k^{(1)}$ задаваме така:

$$\alpha_k^{(1)} = \begin{cases} 10^6, & 0 \le z \le h, \\ 10^4, & \delta_k < \delta,\ h < z \le H, \\ \dfrac{\lambda_M}{\delta_M}, & \delta_k \ge \delta,\ h < z \le H, \end{cases} \tag{4.10}$$

където λ_M е коефициентът на топлопроводност на въздуха.

Окончателно уравненията (4.2) — (4.9) дават пълния модел на разглеждания нестационарен кристализационен процес. По-нататък ще се

заемем с подробното дефиниране на коефициентите му и начина за приближеното му решаване.

Сама по себе си задачата (4.2) — (4.9) представлява задача от типа на Стефан за определяне положението на фазовата граница $u = u_S$. Това по естествен път налага поставянето на допълнително гранично условие, което затруднява изчислителния процес чрез добавяне на ново мрежово нелинейно уравнение. По-надолу ние ще покажем как можем да намерим разположението на линията $u = u_S$ във вътрешността на отливката за всички моменти от време, без да обособяваме наличието ѝ като отделно гранично условие. За целта ще използваме т. нар. „изгладена“ или приблизително делтаобразна функция $\delta(u - u_S)$.

В средата на 60-те години на XX век А. А. Самарский и Б. Д. Моисеенко установяват, че задачата на Стефан може да бъде адекватно решена, като се използват хомогенни диференчни схеми, без да се налага явно отделянето на свободната граница [4]. При това броят на пространствените променливи или броят на фазовите преходи е без значение. Самарский и Моисеенко въвеждат „изгладени“ коефициенти на топлоемкост $\widetilde{c}$ и на топлопроводност $\widetilde{\lambda}$, считайки, че и двата зависят от неизвестната температура. В модела (4.2) — (4.9) коефициентът на топлопроводност е прекъснат, но не е функция на температурата. Затова него няма да „изглаждаме“. Ще покажем как да приложим тази идея само към коефициента на топлоемкост.

Определянето на границата на раздела на фазите се основава на следното условие: по продължение на тази граница търсената температура $u(r, z, t)$ е равна на температурата на фазовия преход u_S, т. е. $u(r, z, t) = u_S$. Последната зависимост всъщност представлява едно нелинейно уравнение относно $R(t)$ — текущото разположение на границата в момента от време t. Без ограничение на общността ще го записваме във вида $\Phi(r, z, t) = 0$.

В конкретния случай е налице едно фазово превръщане (от течно в твърдо агрегатно състояние), което отговаря на две различни фази. Да означим с индекс 1 фазата, за която $u < u_S$, и с индекс 2 — тази, за която $u > u_S$. Тъй като $\operatorname{grad}\Phi$ е насочен по нормалата към повърхнината Σ на раздела на фазите, то нормалната компонента на топлинния поток $W = -\lambda \operatorname{grad} u$ върху Σ е равна на

$$W_{1,2} = -\left(\lambda \operatorname{grad} u, \frac{\operatorname{grad}\Phi}{|\operatorname{grad}\Phi|}\right).$$

Разликата от потоците $W_2 - W_1$ е равна на произведението от енталпията на фазовия преход η и нормалната компонента на скоростта $\dfrac{\mathrm{d}R}{\mathrm{d}t}$, с която

се премества в пространството границата на раздела на фазите:

$$W_2 - W_1 = \eta \left(\frac{\mathrm{d}\, R}{\mathrm{d}\, t}, \frac{\operatorname{grad} \Phi}{|\operatorname{grad} \Phi|} \right). \tag{4.11}$$

Като използваме още, че върху повърхнината Σ е изпълнено

$$\frac{\mathrm{d}\, \Phi(R(t), t)}{\mathrm{d}\, t} = \frac{\partial \Phi}{\partial t} + \left(\frac{\mathrm{d}\, R}{\mathrm{d}\, t}, \operatorname{grad} \Phi \right) = 0,$$

представяме (4.11) във вида

$$\begin{cases} u = u_S, \\ (W_1 - W_2, \operatorname{grad} \Phi) + \eta \dfrac{\partial \Phi}{\partial t} = 0 \ \text{ при } \ (r, z) = R(t). \end{cases} \tag{4.12}$$

Физичното изискване, от което следва граничното условие (4.12), е следното: при температура на фазовия преход $u = u_S$ енергията w като функция на температурата се характеризира със „скок“ η — *топлина (енталпия) на фазовия преход*. Имаме, че

$$w = \int_0^u c(u)\, \mathrm{d}\, u + \eta \mu(u - u_S),\ \mu(\xi) = \begin{cases} 1, & \xi \geq 0, \\ 0, & \xi < 0. \end{cases} \tag{4.13}$$

Тук енталпията η е пресметната по формулата $\eta = \kappa(1 - \psi_S)$, където κ е топлината на кристализация, а ψ_S е съдържанието на твърдата фаза в двуфазната зона [23]. Като заместим израза (4.13) в уравнението на енергията

$$\frac{\partial w}{\partial t} = \operatorname{div} (\lambda \operatorname{grad} u)$$

и отчетем, че $\dfrac{\mathrm{d}\, \mu(\xi)}{\mathrm{d}\, \xi} = \delta(\xi)$ представлява делта-функцията на Дирак, от уравнението на топлопроводността веднага получаваме

$$\left[c(u) + \eta \delta(u - u_S) \right] \frac{\partial u}{\partial t} = \operatorname{div}(\lambda \operatorname{grad} u). \tag{4.14}$$

В [4] е доказано, че, записано в тази форма, традиционното уравнение на топлопроводността с няколко пространствени променливи съдържа и условие (4.11).

И така, с цел преминаване към хомогенна консервативна диференчна схема остана да покажем как да конструираме приближение за делта-функцията $\delta(u - u_S)$.

Да заменим прекъснатата делта-функция с приблизително делтаобразната $\delta(u - u_S, \Delta) \geq 0$ [4], [20]. Нейният аргумент е означен с Δ и представлява половината от дължината на интервала $\big[u_S - \Delta;\, u_S + \Delta\big]$, в който $\delta(u - u_S,\, \Delta)$ е различна от нула. Подобно „размазване“ или „изглаждане“ е равносилно на това, да заменим в интервала $\big[u_S - \Delta;\, u_S + \Delta\big]$ прекъснатата функция $\mu(u - u_S)$ с непрекъснатата $\mu(u - u_S,\, \Delta)$, такава, че $\mu'(\xi,\, \Delta) = \delta(\xi,\, \Delta)$.

Въвеждаме „изгладения“ коефициент на топлоемкост

$$\widetilde{c} = c(u) + \eta\delta(u - u_S,\, \Delta)$$

така, че:

- $\widetilde{c} = c_1$ при $u < u_S - \Delta$, $\widetilde{c} = c_2$ при $u > u_S + \Delta$ (т. е. $\widetilde{c}(u) = c(u)$ за $u_S - \Delta < u < u_S + \Delta$);
- изменението на енталпията в интервала $\big(u_S - \Delta;\, u_S + \Delta\big)$ се запазва:

$$\int\limits_{u_S-\Delta}^{u_S+\Delta} \widetilde{c}(u)\,\mathrm{d}\,u = \eta + \int\limits_{u_S-\Delta}^{u_S} c_1(u)\,\mathrm{d}\,u + \int\limits_{u_S}^{u_S+\Delta} c_2(u)\,\mathrm{d}\,u. \tag{4.15}$$

В най-простия случай коефициентите c_1 и c_2 не зависят от u. Тогава в интервала $\big(u_S - \Delta;\, u_S + \Delta\big)$ можем да изберем

$$\widetilde{c}(u) = \frac{\eta}{2\Delta} + \frac{c_1 + c_2}{2}. \tag{4.16}$$

За разглежданата задача $c_1 = c_S$ и $c_2 = c_S - \kappa\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}$. От равенството за запазване на енталпията (4.15) имаме

$$\int\limits_{u_S-\Delta}^{u_S+\Delta} \widetilde{c}(u)\,\mathrm{d}\,u = \eta + \int\limits_{u_S-\Delta}^{u_S} c_S\,\mathrm{d}\,u + \int\limits_{u_S}^{u_S+\Delta} \left(c_S - \kappa\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right)\mathrm{d}\,u.$$

Ако предположим, че в интервала $\big(u_S - \Delta;\, u_S + \Delta\big)$ е изпълнено $\widetilde{c}(u) = \mathrm{const}$, и интегрираме явно в последното равенство, веднага получаваме

$$\widetilde{c} = c_S + \frac{\eta}{2\Delta} - \frac{\kappa}{2\Delta}\Big[\psi(u_S + \Delta) - \psi(u_S)\Big]. \tag{4.17}$$

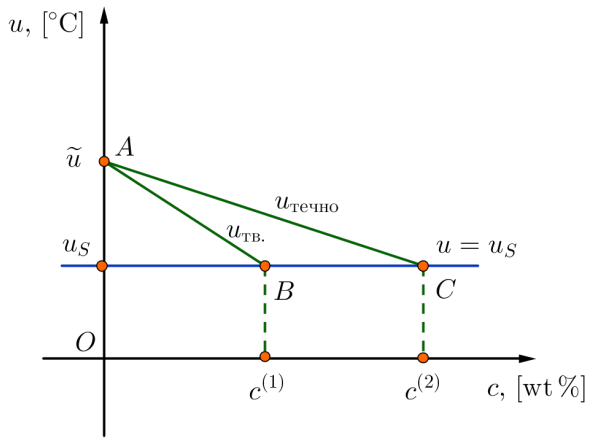
Формула (4.17) вече задава „изгладения“ коефициент на топлоемкост $\widetilde{c}(u)$ за $u_S - \Delta < u < u_S + \Delta$. И така, окончателно получаваме, че коефициентът на топлоемкост $c(u)$ за всяка допустима температура u е

прекъсната функция от тази температура и се пресмята така:

$$c(u) = \begin{cases} c_L, & u > u_L, \\ c_S - \kappa \dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}, & u_S + \Delta < u \leq u_L, \\ c_S + \dfrac{\eta}{2\Delta} - \dfrac{\kappa}{2\Delta}\Big[\psi(u_S + \Delta) - \psi(u_S)\Big], & u_S - \Delta < u \leq u_S + \Delta, \\ c_S, & u \leq u_S - \Delta. \end{cases} \tag{4.18}$$

За подробен алгоритъм относно пресмятането на параметъра Δ вж. секция 4.4.

4.2 Относителен дял на твърдата фаза $\psi(u)$



Фигура 4.2: *Схематично представяне на фазова диаграма за аналитично определяне на функцията $\psi(u)$*

Функцията $\psi(u)$ определя относителния дял на твърдата фаза в общия обем на отливката, следователно $\psi(u) \in [0; 1]$. Нейният явен вид представлява дробно-рационален израз на температурата u и се възстановява по равновесната диаграма на състоянията за конкретната сплав [14], [22], [24].

На фиг. 4.2 в ортогоналната координатна система cOu върху абсцисната ос са нанесени стойностите на масовата концентрация c, а върху

ординатната — на температурата u. Тук $\widetilde{u}$ е температурата на топене на чистия метал. Областта между линиите с уравнения $u = u_{\text{тв.}}(c)$, $u = u_{\text{течно}}(c)$ и $u = u_S$ дефинира двуфазната зона.

Ако решим уравненията $u = u_{\text{течно}}(c)$ и $u = u_{\text{тв.}}(c)$ относно концентрацията c, получаваме явните връзки $c = c_{\text{течно}}(u)$ и $c = c_{\text{тв.}}(u)$. Ще търсим $\psi(u)$ във вида

$$\psi(u) = \begin{cases} 0, & u \geq u_L, \\ \dfrac{c_{\text{течно}}(u) - C_0 + a}{c_{\text{течно}}(u) - c_{\text{тв.}}(u) + b}, & u_S \leq u \leq u_L, \\ 1, & u \leq u_S. \end{cases} \tag{4.19}$$

Тук величината C_0 представлява началната концентрация на легиращия елемент в сплавта. Константите a и b ще определим така, че функцията $\psi(u)$ да е непрекъсната в точките $u = u_L$ и $u = u_S$, т. е. да са изпълнени условията $\psi(u_L) = 0$ и $\psi(u_S) = 1$.

Намираме скаларните параметрични уравнения на правите $u_{\text{тв.}}(c)$ и $u_{\text{течно}}(c)$ и от тях изразяваме стойностите $c_{\text{тв.}}(u)$ и $c_{\text{течно}}(u)$. Понеже $A(0;\, \widetilde{u})$, $B(c^{(1)};\, u_S)$ и $C(c^{(2)};\, u_S)$, то направляващите вектори на тези прави са съответно $\overrightarrow{AB}(c^{(1)};\, u_S - \widetilde{u})$ и $\overrightarrow{AC}(c^{(2)};\, u_S - \widetilde{u})$ (вж. фиг. 4.2). Последователно получаваме

$$AB : \begin{cases} c = 0 + p_1 c^{(1)} \\ u = u_L + p_1(u_S - \widetilde{u}) \end{cases} \Rightarrow \frac{c - 0}{c^{(1)}} = \frac{u - u_L}{u_S - \widetilde{u}} \Rightarrow$$

$$c^{(1)}(u - u_L) = c(u_S - \widetilde{u}) \Rightarrow c_{\text{тв.}}(u) = \frac{c^{(1)}(u - u_L)}{u_S - \widetilde{u}};$$

$$AC : \begin{cases} c = 0 + p_2 c^{(2)} \\ u = u_L + p_2(u_S - \widetilde{u}) \end{cases} \Rightarrow \frac{c - 0}{c^{(2)}} = \frac{u - u_L}{u_S - \widetilde{u}} \Rightarrow$$

$$c^{(2)}(u - u_L) = c(u_S - \widetilde{u}) \Rightarrow c_{\text{течно}}(u) = \frac{c^{(2)}(u - u_L)}{u_S - \widetilde{u}},$$

където $p_1 \in \mathbb{R}$ и $p_2 \in \mathbb{R}$ са параметрите на правите AB и AC. Зависимос-

тите $c = c_{\text{течно}}(u)$ и $c = c_{\text{тв.}}(u)$ са линейни. Да означим за краткост

$$c_{\text{тв.}}(u) = a_1 u + b_1,\ a_1 = \frac{c^{(1)}}{u_S - \widetilde{u}},\ b_1 = -\frac{c^{(1)} u_L}{u_S - \widetilde{u}}, \tag{4.20}$$

$$c_{\text{течно}}(u) = a_2 u + b_2,\ a_2 = \frac{c^{(2)}}{u_S - \widetilde{u}},\ b_2 = -\frac{c^{(2)} u_L}{u_S - \widetilde{u}}. \tag{4.21}$$

От (4.19) достигаме до системата уравнения

$$\left|\begin{array}{l}\psi(u_L) = 0\\ \psi(u_S) = 1\end{array}\right. \Rightarrow \left|\begin{array}{l}\dfrac{a_2 u_L + b_2 - C_0 + a}{(a_2 - a_1)u_L + (b_2 - b_1) + b} = 0\\[2ex] \dfrac{a_2 u_S + b_2 - C_0 + a}{(a_2 - a_1)u_S + (b_2 - b_1) + b} = 1\end{array}\right. ,$$

чиито решения са

$$a \;=\; C_0 - b_2 - a_2 u_L, \tag{4.22}$$

$$b \;=\; a_1 u_S - a_2 u_L + b_1 - b_2. \tag{4.23}$$

Заместваме (4.22) и (4.23) в (4.19) при $u_S \le u \le u_L$, опростяваме и намираме окончателно

$$\psi(u) = \begin{cases} 0, & u \ge u_L,\\[2ex] \dfrac{a_2(u - u_L)}{(a_2 - a_1)u + a_1 u_S - a_2 u_L}, & u_S \le u \le u_L,\\[2ex] 1, & u \le u_S. \end{cases} \tag{4.24}$$

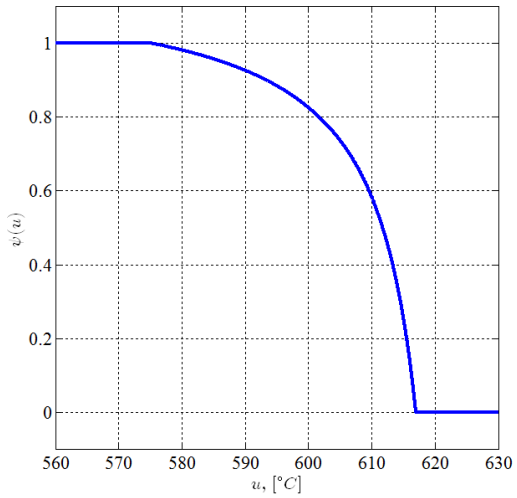
За сплавта **AlSi7Mg** със 7 wt%-но съдържание на легиращ елемент **Si** конкретните числени стойности на температурите $\widetilde{u}$, u_L и u_S, на масовите концентрации $c^{(1)}$ и $c^{(2)}$, както и на двата допълнителни параметъра a_1 и a_2, са дадени в таблица 4.1. При тези стойности функцията $\psi(u)$ има вида

$$\psi(u) = \begin{cases} 0, & u \ge u_L,\\[2ex] \dfrac{-0.00136471u + 0.84202353}{-0.001170059u + 0.73040588}, & u_S \le u \le u_L,\\[2ex] 1, & u \le u_S. \end{cases} \tag{4.25}$$

На фиг. 4.3 е начертана нейната графика за $u \in [560^\circ\,\text{C};\ 630^\circ\,\text{C}]$.

Параметър	Стойност
$\widetilde{u}$	660° C
u_L	617° C
u_S	575° C
$c^{(1)}$	1.65 wt%
$c^{(2)}$	11.6 wt%
a_1	-0.00019412
a_2	-0.00136471

Таблица 4.1: *Стойности на параметрите u_L, u_S, $c^{(1)}$, $c^{(2)}$, a_1 и a_2, които определят функцията $\psi(u)$ по формулата* (4.24)



Фигура 4.3: *Графика на функцията $\psi(u)$, алуминиева сплав* ***AlSi7Mg***

4.3 Диференчна схема

Да се заемем с приближеното решаване на така формулираната диференциална задача (4.2) —(4.9). За целта в областта $[0;\, R] \times [0;\, H] \times [0;\, T]$ да въведем следните мрежи:

- ☐ неравномерна отместена мрежа $\widehat{\omega}^*_{h^r}$ по променливата r;
- ☐ неравномерна неотместена мрежа $\widehat{\omega}_{h^z}$ по променливата z;
- ☐ равномерна неотместена мрежа $\overline{\omega}_\tau$ по променливата t.

Мрежата $\widehat{\omega}^*_{h^r}$ по променливата r конструираме така, че отместването да е $\dfrac{h^r_2}{2}$ от координатното начало и точката $r = R_k$ да бъде „двойна“:

$$\widehat{\omega}^*_{h^r} = \left\{ r_i = r_{i-1} + h^r_i,\ i = 2,\ 3,\ \ldots,\ N;\ r_1 = \frac{h^r_2}{2},\ r_Q = r_{Q+1} = R_k,\ r_N = R \right\}. \tag{4.26}$$

Освен това въвеждаме и означението

$$\hbar^r_i = \frac{1}{2}\left(h^r_i + h^r_{i+1}\right),\ i = 2,\ 3,\ \ldots,\ N-1,$$

като полагаме $\hbar^r_1 = h^r_2$ и $\hbar^r_N = \dfrac{h^r_N}{2}$.

Неравномерната мрежа $\widehat{\omega}_{h^z}$ избираме така, че точката $z = h$ също да бъде „двойна“:

$$\widehat{\omega}_{h^z} = \left\{ z_1 = 0,\ z_j = z_{j-1} + h^z_j,\ j = 2,\ 3,\ \ldots,\ M,\ z_P = z_{P+1} = h,\ z_M = H \right\}. \tag{4.27}$$

Нека

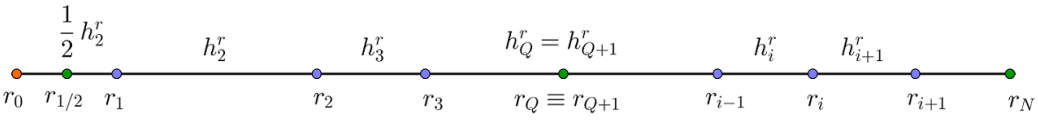
$$\hbar^z_j = \frac{1}{2}\left(h^z_j + h^z_{j+1}\right),\ j = 2,\ 3,\ \ldots,\ M-1,$$

като за установяване на единно изписване на означенията приемем, че $\hbar^z_1 = \dfrac{h^z_1}{2}$ и $\hbar^z_M = \dfrac{h^z_M}{2}$.

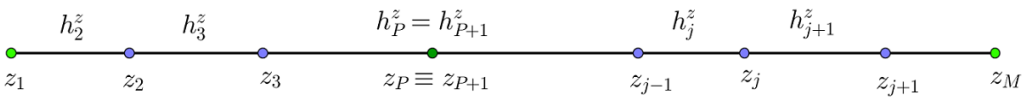
Мрежата $\overline{\omega}_\tau$ по времето t е

$$\overline{\omega}_\tau = \left\{ t_n = t_{n-1} + \tau,\ n = 1,\ 2,\ \ldots,\ S,\ t_0 = 0,\ t_S = T \right\}. \tag{4.28}$$

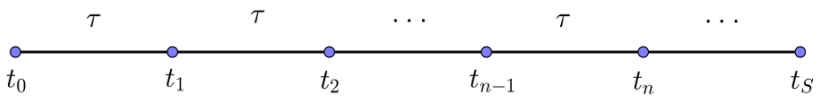
На фиг. 4.4 — 4.6 тези мрежи са изобразени графично.



Фигура 4.4: *Неравномерна отместена мрежа $\widehat{\omega}^*_{h^r}$ в направление r*



Фигура 4.5: *Неравномерна неотместена мрежа $\widehat{\omega}_{h^z}$ в направление z*



Фигура 4.6: *Равномерна неотместена мрежа $\overline{\omega}_\tau$ в направление t*

За решаване на изходната задача (4.2) —(4.9) ще използваме икономичен числен метод за нейното числено решаване — локално едномерна схема (ЛЕС) [1], [2], [3], [44]. Разглеждаме следните две едномерни задачи:

□ в направление r: z е фиксирано, $t \in (t_n;\, t_{n+1/2}]$, $n = 0,\, 1,\, \ldots,\, S-1$

$$\frac{1}{2}c(v)\rho\frac{\partial v}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(v)\frac{\partial v}{\partial r}\right),\ r \in \omega_{h^r}^*, \tag{4.29}$$

$$v(r,z,t_n) = u(r,z,t_n),\ v(r,z,0) = u(r,z,0),\ r \in \widehat{\omega}_{h^r}^*, \tag{4.30}$$

$$\lim_{r\to 0} r\lambda(v)\frac{\partial v}{\partial r} = 0, \tag{4.31}$$

$$\lambda(v)\frac{\partial v}{\partial r}\bigg|_{r=R_k^-} = \lambda(v)\frac{\partial v}{\partial r}\bigg|_{r=R_k^+} = \alpha_k^{(1)}\left(v\Big|_{r=R_k^+} - v\Big|_{r=R_k^-}\right), \tag{4.32}$$

$$\lambda(v)\frac{\partial v}{\partial r}\bigg|_{r=R} = -\alpha\left(v\Big|_{r=R} - u_{\text{ок. ср.}}\right); \tag{4.33}$$

□ в направление z: r е фиксирано, $t \in (t_{n+1/2};\, t_{n+1}]$, $n = 0,\, 1,\, \ldots,\, S-1$

$$\frac{1}{2}c(u)\rho\frac{\partial u}{\partial t} = \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial r}\right),\ z \in \omega_{h^z}, \tag{4.34}$$

$$u(r,z,t_{n+1/2}) = v(r,z,t_{n+1/2}),\ z \in \widehat{\omega}_{h^z}, \tag{4.35}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=0} = \widetilde{\alpha}\left(u\Big|_{z=0} - u_{\text{ок. ср.}}\right), \tag{4.36}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^-} = \lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^+} = \alpha_k\left(u\Big|_{z=h^+} - u\Big|_{z=h^-}\right), \tag{4.37}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=H} = -\overline{\alpha}\left(u\Big|_{z=H} - u_{\text{ок. ср.}}\right). \tag{4.38}$$

За всяка от задачите (4.29) — (4.33) и (4.34) — (4.38) ще изведем диференчни схеми по метода на баланса.

4.3.1 Диференчна схема в направление r

С цел постигане на безусловна устойчивост и сходимост ще построим чисто неявна линеаризирана схема. Навсякъде по-надолу ще считаме индекса j по променливата z фиксиран. Ще запазим означението v и за приближеното решение.

□ *Апроксимация на диференциалното уравнение (4.29)*

Умножаваме двете страни на уравнение (4.29) с r и интегрираме в правоъгълника $[r_{i-1/2};\, r_{i+1/2}] \times [t_n;\, t_{n+1/2}]$ при $i = 2,\, 3,\, \ldots,\, N-1$, $i \neq Q$, $i \neq Q+1$, $n = 0,\, 1,\, \ldots,\, S-1$. Последователно получаваме:

$$\int\limits_{r_{i-1/2}}^{r_{i+1/2}} \int\limits_{t_n}^{t_{n+1/2}} \frac{1}{2} rc(v)\rho \frac{\partial v}{\partial t}\,\mathrm{d}\,t\,\mathrm{d}\,r = \int\limits_{t_n}^{t_{n+1/2}} \int\limits_{r_{i-1/2}}^{r_{i+1/2}} \frac{\partial}{\partial r}\left(r\lambda(v)\frac{\partial v}{\partial r}\right)\mathrm{d}\,r\,\mathrm{d}\,t,$$

$$\int\limits_{r_{i-1/2}}^{r_{i+1/2}} \frac{1}{2} rc(v_j^n)\rho\left[v^{n+1/2}(r,z_j) - v^n(r,z_j)\right]\mathrm{d}\,r \approx \int\limits_{t_n}^{t_{n+1/2}} \left[W_{i-1/2,j}(t) - W_{i+1/2,j}(t)\right]\mathrm{d}\,t,$$

където $W = -r\lambda(v)\dfrac{\partial v}{\partial r}$ е топлинният поток. Ако приемем, че $r \approx r_i$ за

$r_{i-1/2} \le r \le r_{i+1/2}$, то имаме

$$\frac{1}{2} r_i \rho \left[v_{i,j}^{n+1/2} - v_{i,j}^{n} \right] \int\limits_{r_{i-1/2}}^{r_{i+1/2}} c(v_j^n)\, \mathrm{d}\, r \approx \frac{\tau}{2} \left(W_{i-1/2,j}^{n+1/2} - W_{i+1/2,j}^{n+1/2} \right),$$

$$r_i \rho \frac{v_{i,j}^{n+1/2} - v_{i,j}^{n}}{\tau} \frac{1}{\hbar_i^r} \int\limits_{r_{i-1/2}}^{r_{i+1/2}} c(v_j^n)\, \mathrm{d}\, r \approx \frac{1}{\hbar_i^r} \left(W_{i-1/2,j}^{n+1/2} - W_{i+1/2,j}^{n+1/2} \right).$$

Въвеждаме означението

$$c_{i,j}^{n} = \frac{1}{\hbar_i^r} \int\limits_{r_{i-1/2}}^{r_{i+1/2}} c(v_j^n)\, \mathrm{d}\, r \tag{4.39}$$

и последното равенство приема вида

$$r_i c_{i,j}^{n} \rho \frac{v_{i,j}^{n+1/2} - v_{i,j}^{n}}{2} \approx \frac{1}{\hbar_i^r} \left(W_{i-1/2,j}^{n+1/2} - W_{i+1/2,j}^{n+1/2} \right). \tag{4.40}$$

Да изведем сега диференчна апроксимация за потока W в точката $\left(r_{i-1/2};\, z_j;\, t_{n+1/2}\right)$. Имаме последователно:

$$\frac{\partial v}{\partial r} = -\frac{W}{r\lambda(v)}, \quad \int\limits_{t_n}^{t_{n+1/2}} \int\limits_{r_{i-1}}^{r_i} \frac{\partial v}{\partial r}\, \mathrm{d}\, r\, \mathrm{d}\, t = -\int\limits_{t_n}^{t_{n+1/2}} \int\limits_{r_{i-1}}^{r_i} \frac{W}{r\lambda(v)}\, \mathrm{d}\, r\, \mathrm{d}\, t,$$

$$\int\limits_{t_n}^{t_{n+1/2}} \left[v(r_i,\, z_j,\, t) - v(r_{i-1},\, z_j,\, t) \right] \mathrm{d}\, t \approx -\int\limits_{t_n}^{t_{n+1/2}} \frac{W(r_{i-1/2},\, z_j,\, t)}{r_{i-1/2}} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j)}\, \mathrm{d}\, r\, \mathrm{d}\, t,$$

$$\frac{\tau}{2} \left(v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2} \right) \approx -\frac{W_{i-1/2,j}^{n+1/2}}{r_{i-1/2}} \int\limits_{t_n}^{t_{n+1/2}} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j)}\, \mathrm{d}\, r\, \mathrm{d}\, t,$$

$$\frac{\tau}{2} \left(v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2} \right) \approx -\frac{W_{i-1/2,j}^{n+1/2}}{r_{i-1/2}} \cdot \frac{\tau}{2} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j^n)}\, \mathrm{d}\, r,$$

$$\frac{v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2}}{h_i^r} \approx -\frac{W_{i-1/2,j}^{n+1/2}}{r_{i-1/2}} \cdot \frac{1}{h_i^r} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j^n)}\, \mathrm{d}\, r,$$

$$W_{i-1/2,j}^{n+1/2} \approx -r_{i-1/2} \left[\frac{1}{h_i^r} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j^n)}\, \mathrm{d}\, r \right]^{-1} \frac{v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2}}{h_i^r}.$$

Въвеждаме означението

$$\overset{0}{a}_{i,j} = \left[\frac{1}{h_i^r} \int\limits_{r_{i-1}}^{r_i} \frac{1}{\lambda(v_j^n)} \,\mathrm{d}\, r \right]^{-1},$$

откъдето за потока окончателно получаваме

$$W_{i-1/2,j}^{n+1/2} \approx -r_{i-1/2} \overset{0}{a}_{i,j} \frac{v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2}}{h_i^r}. \tag{4.41}$$

Заменяйки във формула (4.41) индекса i с $i+1$, веднага намираме израз и за потока в точката $\left(r_{i+1/2};\, z_j;\, t_{n+1/2}\right)$:

$$W_{i+1/2,j}^{n+1/2} \approx -r_{i+1/2} \overset{0}{a}_{i+1,j} \frac{v_{i+1,j}^{n+1/2} - v_{i,j}^{n+1/2}}{h_{i+1}^r}. \tag{4.42}$$

Като запазим означението v и за приближеното решение и заместим апроксимациите (4.41), (4.42) в (4.40), получаваме окончателно апроксимацията на основното уравнение (4.29):

$$c_{i,j}^n \rho \frac{v_{i,j}^{n+1/2} - v_{i,j}^n}{2} = \frac{1}{\hbar_i^r} \cdot \frac{1}{r_i} \left[r_{i+1/2} \overset{0}{a}_{i+1,j} \frac{v_{i+1,j}^{n+1/2} - v_{i,j}^{n+1/2}}{h_{i+1}^r} - r_{i-1/2} \overset{0}{a}_{i,j} \frac{v_{i,j}^{n+1/2} - v_{i-1,j}^{n+1/2}}{h_i^r} \right].$$

Приведено във вид, удобен за прилагане на метода на прогонката, последното уравнение изглежда така:

$$-\frac{r_{i-1/2} \overset{0}{a}_i}{r_i \hbar_i^r h_i^r} v_{i-1,j}^{n+1/2} + \left[\frac{c_{i,j}^n \rho}{\tau} + \frac{r_{i-1/2} \overset{0}{a}_i}{r_i \hbar_i^r h_i^r} + \frac{r_{i+1/2} \overset{0}{a}_{i+1}}{r_i \hbar_i^r h_{i+1}^r} \right] v_{i,j}^{n+1/2} - \frac{r_{i+1/2} \overset{0}{a}_{i+1}}{r_i \hbar_i^r h_{i+1}^r} v_{i+1,j}^{n+1/2}$$
$$= \frac{c_{i,j}^n \rho}{\tau} v_{i,j}^n, i = 2,\, 3,\, \ldots,\, Q-1,\, Q+2,\, \ldots,\, N-1. \tag{4.43}$$

□ *Апроксимация на началното условие (4.30)*

Началното условие се апроксимира точно:

$$v_{i,j}^n = u_{i,j}^n,\ i = 1,\, 2,\, \ldots,\, N,\ n = 1,\, 2,\, \ldots,\, S. \tag{4.44}$$

□ *Апроксимация на условието за симетрия (4.31)*

Означаваме $r_0 = 0$. Умножаваме двете страни на уравнение (4.29) с $r \neq 0$, интегрираме го в правоъгълника $\left[r_0;\, r_{3/2}\right] \times \left[t_n; t_{n+1/2}\right]$ и вземаме

функционалните стойности в точката r_1:

$$\int_{r_0}^{r_{3/2}} \int_{t_n}^{t_{n+1/2}} \frac{1}{2} r c(v) \rho \frac{\partial v}{\partial t} \,\mathrm{d}\, t \,\mathrm{d}\, r = \int_{t_n}^{t_{n+1/2}} \int_{r_0}^{r_{3/2}} \frac{\partial}{\partial r} \left(r \lambda(v) \frac{\partial v}{\partial r} \right) \mathrm{d}\, r \,\mathrm{d}\, t,$$

$$\frac{1}{2} r_1 \rho \int_{r_0}^{r_{3/2}} c(v_j^n) \,\mathrm{d}\, r \left(v_{1,j}^{n+1/2} - v_{1,j}^n \right) \approx \frac{\tau}{2} \left(W_{1,j}^{n+1/2} - W_{3/2,j}^{n+1/2} \right).$$

Но $r_{3/2} - r_0 = \hbar_1^r$, тогава разделяме двете страни на последното равенство с $\hbar_1^r$, умножаваме по 2 и достигаме до

$$r_1 \rho \frac{1}{\hbar_1^r} \int_{r_0}^{r_{3/2}} c(v_j^n) \,\mathrm{d}\, r \left(v_{1,j}^{n+1/2} - v_{1,j}^n \right) \approx \frac{\tau}{\hbar_1^r} \left(W_{0,j}^{n+1/2} - W_{3/2,j}^{n+1/2} \right).$$

Нека

$$c_{1,j}^n = \frac{1}{\hbar_1^r} \int_{r_0}^{r_{3/2}} c(v_j^n) \,\mathrm{d}\, r.$$

Като отчетем, че от (4.31) имаме $W_{0,j}^{n+1/2} = 0$, разделяме с $r_1 \tau \neq 0$ и получаваме приближението

$$c_{1,j}^n \rho \frac{v_{1,j}^{n+1/2} - v_{1,j}^n}{\tau} \approx -\frac{1}{r_1 \hbar_1^r} W_{3/2,j}^{n+1/2}. \tag{4.45}$$

От формулата (4.41) при $i = 2$ лесно намираме, че

$$W_{3/2,j}^{n+1/2} \approx -r_{3/2} \overset{0}{a}_{2,j} \frac{v_{2,j}^{n+1/2} - v_{1,j}^{n+1/2}}{h_2^r},$$

и заместваики в уравнение (4.45), веднага достигаме до апроксимацията на условие (4.31):

$$c_{1,j}^n \rho \frac{v_{1,j}^{n+1/2} - v_{1,j}^n}{\tau} = \frac{1}{r_1 \hbar_1^r} r_{3/2} \overset{0}{a}_{2,j} \frac{v_{2,j}^{n+1/2} - v_{1,j}^{n+1/2}}{h_2^r},$$

където

$$\overset{0}{a}_{2,j} = \left[\frac{1}{h_2^r} \int_{r_1}^{r_2} \frac{1}{\lambda(v_j^n)} \,\mathrm{d}\, r \right]^{-1}.$$

Приведено във вид, удобен за прилагане на метода на прогонката, последното диференчно уравнение изглежда така:

$$\left[\frac{c_{1,j}^n\rho}{\tau}+\frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar_1^r h_2^r}\right]v_{1,j}^{n+1/2}-\frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar_1^r h_2^r}v_{2,j}^{n+1/2}=\frac{c_{1,j}^n\rho}{\tau}v_{1,j}^n. \tag{4.46}$$

□ *Апроксимация на контактното условие (4.32)*

Да представим условието (4.32) във вида

$$\lambda(v)\frac{\partial v}{\partial r}\bigg|_{r=R_k^-}=\alpha_k^{(1)}\left(u\Big|_{r=R_k^+}-u\Big|_{r=R_k^-}\right), \tag{4.47а}$$

$$\lambda(v)\frac{\partial v}{\partial r}\bigg|_{r=R_k^+}=\alpha_k^{(1)}\left(u\Big|_{r=R_k^+}-u\Big|_{r=R_k^-}\right). \tag{4.47б}$$

За всяко от уравненията (4.47а), (4.47б) ще изведем по метода на баланса диференчни апроксимации с ред на сходимост $O\left(\left\|h^r\right\|^2+\tau\right)$, където

$$\left\|h^r\right\|=\left[\sum_{k=1}^{N}\left(h_k^r\right)^2\right]^{\frac{1}{2}}$$

е евклидовата норма на вектора от стъпките $\left(h_1^r;\,h_2^r;\,\ldots,\,h_N^r\right)$ по радиалната координата.

Да получим сега диференчна апроксимация на условието (4.47а). След умножаване на двете страни на (4.29) с r и интегриране в правоъгълника $\left[r_{Q-1/2};\,r_Q\right]\times\left[t_n;\,t_{n+1/2}\right]$ имаме:

$$\frac{1}{2}\int\limits_{r_{Q-1/2}}^{r_Q}\int\limits_{t_n}^{t_{n+1/2}}rc(v)\rho\frac{\partial v}{\partial t}\,\mathrm{d}\,t\,\mathrm{d}\,r=\int\limits_{t_n}^{t_{n+1/2}}\int\limits_{r_{Q-1/2}}^{r_Q}\frac{\partial}{\partial r}\left(r\lambda(v)\frac{\partial v}{\partial r}\right)\mathrm{d}\,r\,\mathrm{d}\,t,$$

$$\frac{1}{2}\int\limits_{r_{Q-1/2}}^{r_Q}rc(v_j^n)\rho\left[v_j^{n+1/2}(r)-v_j^n(r)\right]\mathrm{d}\,r\approx\int\limits_{t_n}^{t_{n+1/2}}\left[W_{Q-1/2,j}(t)-W_{Q,j}(t)\right]\mathrm{d}\,t,$$

$$\frac{1}{2}r_Q\rho\int\limits_{r_{Q-1/2}}^{r_Q}c(v_j^n)\,\mathrm{d}\,r\left(v_{Q,j}^{n+1/2}-v_{Q,j}^n\right)\approx\frac{\tau}{2}\left(W_{Q-1/2,j}^{n+1/2}-W_{Q,j}^{n+1/2}\right).$$

Като използваме означението (4.39), достигаме до

$$c_{Q,j}^n\rho\frac{v_{Q,j}^{n+1/2}-v_{Q,j}^n}{\tau}\approx\frac{1}{\hbar_Q^r r_Q}\left(W_{Q-1/2,j}^{n+1/2}-W_{Q,j}^{n+1/2}\right). \tag{4.48}$$

Но от равенството (4.41) при $i = Q$ и условието (4.47а) след заместване на потоците в (4.48) получаваме мрежовото уравнение

$$c^n_{Q,j}\rho\frac{v^{n+1/2}_{Q,j} - v^n_{Q,j}}{\tau} = \frac{1}{\hbar^r_Q r_Q}\left[-r_{Q-1/2}\overset{0}{a}_{Q,j}\frac{v^{n+1/2}_{Q,j} - v^{n+1/2}_{Q-1,j}}{h^r_Q} + \alpha^{(1)}_k\left(v^{n+1/2}_{Q+1,j} - v^{n+1/2}_{Q,j}\right)\right]. \quad (4.49)$$

По аналогичен начин, като интегрираме основното уравнение (4.29) в правоъгълника $\left[r_{Q+1};\, r_{Q+3/2}\right] \times \left[t_n;\, t_{n+1/2}\right]$ и използваме, че

$$W^{n+1/2}_{Q+1,j} \approx -r_{Q+1}\alpha^{(1)}_k\left(v^{n+1/2}_{Q+1,j} - v^{n+1/2}_{Q,j}\right),$$

$$W^{n+1/2}_{Q+3/2,j} \approx -r_{Q+3/2}\overset{0}{a}_{Q+2,j}\frac{v^{n+1/2}_{Q+2,j} - v^{n+1/2}_{Q+1,j}}{h^r_{Q+2}},$$

получаваме диференчния аналог на (4.47б):

$$c^n_{Q+1,j}\rho\frac{v^{n+1/2}_{Q+1,j} - v^n_{Q+1,j}}{\tau} = \frac{1}{\hbar^r_{Q+1} r_{Q+1}}\left[-r_{Q+1}\alpha^{(1)}_k\left(v^{n+1/2}_{Q+1,j} - v^{n+1/2}_{Q,j}\right) + r_{Q+3/2}\overset{0}{a}_{Q+2}\frac{v^{n+1/2}_{Q+2,j} - v^{n+1/2}_{Q+1,j}}{h^r_{Q+2}}\right]. \quad (4.50)$$

Привеждаме уравненията (4.49), (4.50) във форма, подходяща за прилагане на метода на прогонката:

$$-\frac{r_{Q-1/2}\overset{0}{a}_Q}{r_Q\hbar^r_Q h^r_Q}v^{n+1/2}_{Q-1,j} + \left[\frac{c^n_{Q,j}\rho}{\tau} + \frac{r_{Q-1/2}\overset{0}{a}_Q}{r_Q\hbar^r_Q h^r_Q} + \frac{\alpha^{(1)}_k}{\hbar^r_Q}\right]v^{n+1/2}_{Q,j} - \frac{\alpha^{(1)}_k}{\hbar^r_Q}v^{n+1/2}_{Q+1,j} = \frac{c^n_{Q,j}\rho}{\tau}v^n_{Q,j}, \quad (4.51а)$$

$$-\frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}}v^{n+1/2}_{Q,j} + \left[\frac{c^n_{Q+1,j}\rho}{\tau} + \frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}} + \frac{r_{Q+3/2}\overset{0}{a}_{Q+2}}{r_{Q+1}\hbar^r_{Q+1}h^r_{Q+2}}\right]v^{n+1/2}_{Q+1,j} - \frac{r_{Q+3/2}\overset{0}{a}_{Q+2}}{r_{Q+1}\hbar^r_{Q+1}h^r_{Q+2}}v^{n+1/2}_{Q+2,j} = \frac{c^n_{Q+1,j}\rho}{\tau}v^n_{Q+1,j}. \quad (4.51б)$$

□ *Апроксимация на дясното гранично условие (4.33)*

Умножаваме уравнение (4.29) с r, интегрираме в правоъгълника $\left[r_{N-1/2};\, r_N\right] \times \left[t_n;\, t_{n+1/2}\right]$ и получаваме

$$c_{N,j}^n \rho \frac{v_{N,j}^{n+1/2} - v_{N,j}^n}{\tau} \approx \frac{1}{r_N \hbar_N^r} \left(W_{N-1/2,j}^{n+1/2} - W_{N,j}^{n+1/2} \right), \tag{4.52}$$

където

$$c_{N,j}^n = \frac{1}{\hbar_N^r} \int\limits_{r_{N-1/2}}^{r_N} c(v_j^n)\, \mathrm{d}\, r.$$

Но от формулата (4.41) при $i = N$ и от граничното условие (4.33) имаме, че

$$W_{N-1/2,j}^{n+1/2} \approx -r_{N-1/2} \overset{0}{a}_{N,j} \frac{v_{N,j}^{n+1/2} - v_{N-1,j}^{n+1/2}}{h_N^r},$$

$$W_{N,j}^{n+1/2} \approx r_N \alpha \left(v_{N,j}^{n+1/2} - u_{\text{ок. ср.}} \right),$$

откъдето след заместване в (4.52) достигаме до мрежовото уравнение

$$c_{N,j}^n \rho \frac{v_{N,j}^{n+1/2} - v_{N,j}^n}{\tau}$$

$$= \frac{1}{r_N \hbar_N^r} \left[-r_{N-1/2} \overset{0}{a}_{N,j} \frac{v_{N,j}^{n+1/2} - v_{N-1,j}^{n+1/2}}{h_N^r} - r_N \alpha \left(v_{N,j}^{n+1/2} - u_{\text{ок. ср.}} \right) \right]. \tag{4.53}$$

Във вид, подходящ за прилагане на метода на прогонката, уравнение (4.53) изглежда така:

$$-\frac{r_{N-1/2} \overset{0}{a}_{N,j}}{r_N \hbar_N^r h_N^r} v_{N-1,j}^{n+1/2} + \left[\frac{c_{N,j}^n \rho}{\tau} + \frac{\alpha}{\hbar_N^r} + \frac{r_{N-1/2} \overset{0}{a}_{N,j}}{r_N \hbar_N^r h_N^r} \right] v_{N,j}^{n+1/2}$$

$$= \frac{c_{N,j}^n \rho}{\tau} u_{N,j}^n + \frac{\alpha u_{\text{ок. ср.}}}{\hbar_N^r} \tag{4.54}$$

И така, мрежовите уравнения (4.43), (4.44), (4.46), (4.51а), (4.51б), (4.54) дефинират следната линейна система алгебрични уравнения, от която се пресмята решението v на полуслоя $t_{n+1/2}$:

$$
\begin{aligned}
&\left[\frac{c^n_{1,j}\rho}{\tau}+\frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar^r_1h^r_2}\right]v^{n+1/2}_{1,j}-\frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar^r_1h^r_2}v^{n+1/2}_{2,j}=\frac{c^n_{1,j}\rho}{\tau}v^n_{1,j},\ \ i=1,\ n=\overline{0,S-1}\\
&-\frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q\hbar^r_Qh^r_Q}v^{n+1/2}_{Q-1,j}+\left[\frac{c^n_{Q,j}\rho}{\tau}+\frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q\hbar^r_Qh^r_Q}+\frac{\alpha^{(1)}_k}{\hbar^r_Q}\right]v^{n+1/2}_{Q,j}-\frac{\alpha^{(1)}_k}{\hbar^r_Q}v^{n+1/2}_{Q+1,j}=\\
&\qquad\qquad\frac{c^n_{Q,j}\rho}{\tau}v^n_{Q,j},\ \ i=Q,\ n=\overline{0,S-1}\\
&-\frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}}v^{n+1/2}_{Q,j}+\left[\frac{c^n_{Q+1,j}\rho}{\tau}+\frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}}+\frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar^r_{Q+1}h^r_{Q+2}}\right]v^{n+1/2}_{Q+1,j}\\
&\quad-\frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar^r_{Q+1}h^r_{Q+2}}v^{n+1/2}_{Q+2,j}=\frac{c^n_{Q+1,j}\rho}{\tau}v^n_{Q+1,j},\ \ i=Q+1,\ n=\overline{0,S-1}\\
&-\frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i\hbar^r_ih^r_i}v^{n+1/2}_{i-1,j}+\left[\frac{c^n_{i,j}\rho}{\tau}+\frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i\hbar^r_ih^r_i}+\frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i\hbar^r_ih^r_{i+1}}\right]v^{n+1/2}_{i,j}\\
&\quad-\frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i\hbar^r_ih^r_{i+1}}v^{n+1/2}_{i+1,j}=\frac{c^n_{i,j}\rho}{\tau}v^n_{i,j},\ \ i\neq\{1;\,Q;\,Q+1;\,N\},\ n=\overline{0,S-1}\\
&-\frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N\hbar^r_Nh^r_N}v^{n+1/2}_{N-1,j}+\left[\frac{c^n_{N,j}\rho}{\tau}+\frac{\alpha}{\hbar^r_N}+\frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N\hbar^r_Nh^r_N}\right]v^{n+1/2}_{N,j}=\frac{c^n_{N,j}\rho}{\tau}u^n_{N,j}\\
&\qquad\qquad+\frac{\alpha u_{\text{ок. ср.}}}{\hbar^r_N},\ \ i=N,\ n=\overline{0,S-1}.
\end{aligned}
\tag{4.55}
$$

Тази система може да бъде решена стандартно чрез някой директен метод (например метода на Гаус). Но поради нейната специална тридиагонална структура е по-икономично от гледна точка на бързодействие (брой аритметични операции) да се използва методът на дясната прогонка [3], [44]. Ще докажем, че за построената диференчна схема (4.55) методът на дясната прогонка е реализуем и устойчив. Нека разгледаме

системата

$$\begin{cases} -C_1 y_1 + B_1 y_2 & = -F_1,\ i = 1 \\ A_i y_{i-1} - C_i y_i + B_i y_{i+1} & = -F_i,\ i = 2,\ 3,\ \ldots,\ N_0 - 1 \\ A_{N_0} y_{N_0-1} - C_{N_0} y_{N_0} & = -F_{N_0},\ i = N_0. \end{cases} \tag{4.56}$$

Методът на дясната прогонка, приложен за намиране на решението на линейната система (4.56), ще бъде реализуем и устойчив, ако са изпълнени условията на следната [3]

Теорема 1 *Нека коефициентите на* (4.56) *са реални числа, такива, че*

$$C_1 \neq 0, \quad C_{N_0} \neq 0, \tag{4.57}$$

$$A_i \neq 0,\ B_i \neq 0, \quad i = 2,\ 3,\ \ldots,\ N_0 - 1, \tag{4.58}$$

$$\left|C_i\right| \geq \left|A_i\right| + \left|B_i\right|, \quad i = 2,\ 3,\ \ldots,\ N_0 - 1, \tag{4.59}$$

$$\left|C_1\right| \geq \left|B_1\right|, \quad \left|C_{N_0}\right| \geq \left|A_{N_0}\right|, \tag{4.60}$$

като поне едно от неравенствата (4.59) *и* (4.60) *е строго, т. е. матрицата от коефициентите има диагонално преобладаване. Тогава алгоритъмът на дясната прогонка е коректен и устойчив.*

За кеофециентите на системата (4.55) имаме:

$$C_1 = -\left[\frac{c^n_{1,j}\rho}{\tau} + \frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1 \hbar^r_1 h^r_2}\right],\ B_1 = -\frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1 \hbar^r_1 h^r_2};$$

$$A_Q = -\frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q \hbar^r_Q h^r_Q},\ C_Q = -\left[\frac{c^n_{Q,j}\rho}{\tau} + \frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q \hbar^r_Q h^r_Q} + \frac{\alpha^{(1)}_k}{\hbar^r_Q}\right],\ B_Q = -\frac{\alpha^{(1)}_k}{\hbar^r_Q};$$

$$A_{Q+1} = -\frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}},\ C_{Q+1} = -\left[\frac{c^n_{Q+1,j}\rho}{\tau} + \frac{\alpha^{(1)}_k}{\hbar^r_{Q+1}} + \frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar^r_{Q+1} h^r_{Q+2}}\right],$$

$$B_{Q+1} = -\frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar^r_{Q+1} h^r_{Q+2}};$$

$$A_N = -\frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N \hbar^r_N h^r_N},\ C_N = -\left[\frac{c^n_{N,j}\rho}{\tau} + \frac{\alpha}{\hbar^r_N} + \frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N \hbar^r_N h^r_N}\right];$$

$$A_i = -\frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i\hbar_i^r h_i^r},\ C_i = -\left[\frac{c_{i,j}^n\rho}{\tau} + \frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i\hbar_i^r h_i^r} + \frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i\hbar_i^r h_{i+1}^r}\right],$$

$$B_i = -\frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i\hbar_i^r h_{i+1}^r}.$$

Понеже $c(u) > 0$, то мрежовите му аналози ще притежават същото свойство. Тогава неравенствата (4.57) и (4.58) са очевидно изпълнени. Да проверим верността на (4.59) и (4.60):

■ $i = 1$:

$$\left|B_1\right| = \frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar_1^r h_2^r},\ \left|C_1\right| = \frac{c_{1,j}^n\rho}{\tau} + \frac{r_{3/2}\overset{0}{a}_{2,j}}{r_1\hbar_1^r h_2^r} = \frac{c_{1,j}^n\rho}{\tau} + \left|B_1\right| > \left|B_1\right|;$$

■ $i = N$:

$$\left|A_N\right| = \frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N\hbar_N^r h_N^r},\ \left|C_N\right| = \frac{c_{N,j}^n\rho}{\tau} + \frac{\alpha}{\hbar_N^r} + \frac{r_{N-1/2}\overset{0}{a}_{N,j}}{r_N\hbar_N^r h_N^r}$$

$$= \frac{c_{N,j}^n\rho}{\tau} + \frac{\alpha}{\hbar_N^r} + \left|A_N\right| > \left|A_N\right|;$$

■ $i = Q$:

$$\left|A_Q\right| = \frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q\hbar_Q^r h_Q^r},\ \left|B_Q\right| = \frac{\alpha_k^{(1)}}{\hbar_Q^r},\ \left|C_Q\right| = \frac{c_{Q,j}^n\rho}{\tau} + \frac{\alpha_k^{(1)}}{\hbar_Q^r} + \frac{r_{Q-1/2}\overset{0}{a}_{Q,j}}{r_Q\hbar_Q^r h_Q^r}$$

$$= \frac{c_{Q,j}^n\rho}{\tau} + \left|A_Q\right| + \left|B_Q\right| > \left|A_Q\right| + \left|B_Q\right|;$$

■ $i = Q + 1$:

$$\left|A_{Q+1}\right| = \frac{\alpha_k^{(1)}}{\hbar_{Q+1}^r},\ \left|B_{Q+1}\right| = \frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar_{Q+1}^r h_{Q+2}^r},\ \left|C_{Q+1}\right| = \frac{c_{Q+1,j}^n\rho}{\tau} + \frac{\alpha_k^{(1)}}{\hbar_{Q+1}^r};$$

$$+ \frac{r_{Q+3/2}\overset{0}{a}_{Q+2,j}}{r_{Q+1}\hbar_{Q+1}^r h_{Q+2}^r} = \frac{c_{Q+1,j}^n\rho}{\tau} + \left|A_{Q+1}\right| + \left|B_{Q+1}\right| > \left|A_{Q+1}\right| + \left|B_{Q+1}\right|;$$

■ $i \neq \{1;\, Q;\, Q+1;\, N\}$:

$$\left|A_i\right| = \frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i \hbar_i^r h_i^r},\ \left|B_i\right| = \frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i \hbar_i^r h_{i+1}^r},\ \left|C_i\right| = \frac{c_{i,j}^n \rho}{\tau} + \frac{r_{i-1/2}\overset{0}{a}_{i,j}}{r_i \hbar_i^r h_i^r} + \frac{r_{i+1/2}\overset{0}{a}_{i+1,j}}{r_i \hbar_i^r h_{i+1}^r}$$

$$= \frac{c_{i,j}^n \rho}{\tau} + \left|A_i\right| + \left|B_i\right| > \left|A_i\right| + \left|B_i\right|.$$

И така, всички условия за реализуемост и устойчивост на метода на прогонката са изпълнени. Следователно системата (4.55) има, при това единствено, решение $v = \left(v_{1,j}^{n+1/2};\, v_{2,j}^{n+1/2};\, \ldots\, v_{N,j}^{n+1/2}\right)$.

4.3.2 Диференчна схема в направление z

Да изведем сега диференчна схема за приближено решаване на едномерната задача (4.34) — (4.38). Отново ще използваме метода на баланса.

□ *Апроксимация на диференциалното уравнение (4.34)*

Интегрираме уравнение (4.34) в правоъгълника $[z_{j-1/2};\, z_{j+1/2}]\times[t_{n+1/2};\, t_{n+1}]$ за $j = 2,\, 3,\, \ldots,\, M-1$ и $n = 1,\, 2,\, \ldots,\, S-1$ и получаваме последователно:

$$\frac{1}{2}\int\limits_{z_{j-1/2}}^{z_{j+1/2}}\int\limits_{t_{n+1/2}}^{t_{n+1}} c(u)\rho\frac{\partial u}{\partial t}\,\mathrm{d}\,t\,\mathrm{d}\,z = \int\limits_{t_{n+1/2}}^{t_{n+1}}\int\limits_{z_{j-1/2}}^{z_{j+1/2}} \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial z}\right)\mathrm{d}\,z\,\mathrm{d}\,t,$$

$$\frac{1}{2}\int\limits_{z_{j-1/2}}^{z_{j+1/2}} c(u_i^{n+1/2})\rho\left[u_i^{n+1}(z) - u_i^{n+1/2}(z)\right]\mathrm{d}\,z \approx \int\limits_{t_{n+1/2}}^{t_{n+1}}\left[W_{i,j-1/2}(t) - W_{i,j+1/2}(t)\right]\mathrm{d}\,t,$$

$$\frac{1}{2}\rho\int\limits_{z_{j-1/2}}^{z_{j+1/2}} c(u_i^{n+1/2})\,\mathrm{d}\,z\left(u_{i,j}^{n+1} - u_{i,j}^{n+1/2}\right) \approx \frac{\tau}{2}\left(W_{i,j-1/2}^{n+1} - W_{i,j+1/2}^{n+1}\right).$$

Разделяме двете страни на последното равенство с $\dfrac{2}{\hbar_j^z \tau}$, въвеждаме означението

$$c_{i,j}^{n+1/2} = \frac{1}{\hbar_j^z}\int\limits_{z_{j-1/2}}^{z_{j+1/2}} c(u_i^{n+1/2})\,\mathrm{d}\,z,$$

и достигаме до диференчното уравнение

$$c_{i,j}^{n+1/2}\rho\frac{u_{i,j}^{n+1} - u_{i,j}^{n+1/2}}{\tau} \approx \frac{1}{\hbar_j^z}\left(W_{i,j-1/2}^{n+1} - W_{i,j+1/2}^{n+1}\right). \qquad (4.61)$$

Както обикновено, от равенството $\dfrac{\partial u}{\partial z} = -\dfrac{W}{\lambda(u)}$ след почленно интегриране в правоъгълника $[z_{j-1};\, z_j] \times [t_{n+1/2};\, t_{n+1}]$ имаме

$$\int\limits_{t_{n+1/2}}^{t_{n+1}} \int\limits_{z_{j-1}}^{z_j} \frac{\partial u}{\partial z}\,\mathrm{d}\,z\,\mathrm{d}\,t = -\int\limits_{t_{n+1/2}}^{t_{n+1}} \int\limits_{z_{j-1}}^{z_j} \frac{W}{\lambda(u)}\,\mathrm{d}\,z\,\mathrm{d}\,t,$$

$$\frac{\tau}{2}\left(u_{i,j}^{n+1} - u_{i,j-1}^{n+1}\right) \approx -W_{i,j-1/2}^{n+1} \cdot \frac{\tau}{2} \int\limits_{z_{j-1}}^{z_j} \frac{1}{\lambda(u_i^{n+1/2})}\,\mathrm{d}\,z,$$

$$W_{i,j-1/2}^{n+1} \approx -\left[\frac{1}{h_j^z} \int\limits_{z_{j-1}}^{z_j} \frac{1}{\lambda(u_i^{n+1/2})}\right]^{-1} \frac{u_{i,j}^{n+1} - u_{i,j-1}^{n+1/2}}{h_j^z}.$$

Полагаме

$$\overset{0}{b}_{i,j} = \left[\frac{1}{h_j^z} \int\limits_{z_{j-1}}^{z_j} \frac{1}{\lambda(u_i^{n+1/2})}\right]^{-1},$$

заместваме в последното равенство и определяме израза за потока в точката $(r_i;\, z_{j-1/2};\, t_{n+1})$:

$$W_{i,j-1/2}^{n+1} \approx -\overset{0}{b}_{i,j} \frac{u_{i,j}^{n+1} - u_{i,j-1}^{n+1}}{h_j^z}. \tag{4.62}$$

От (4.62) след замяна на индекса j с $j+1$ веднага намираме и потока в точката $(r_i;\, z_{j+1/2};\, t_{n+1})$:

$$W_{i,j+1/2}^{n+1} \approx -\overset{0}{b}_{i,j+1} \frac{u_{i,j+1}^{n+1} - u_{i,j}^{n+1}}{h_{j+1}^z}. \tag{4.63}$$

Заместваме (4.62) и (4.63) в (4.61) и получаваме диференчното уравнение

$$c_{i,j}^{n+1/2} \rho \frac{u_{i,j}^{n+1} - u_{i,j}^{n+1/2}}{\tau} = \frac{1}{\hbar_j^z}\left[\overset{0}{b}_{i,j+1} \frac{u_{i,j+1}^{n+1} - u_{i,j}^{n+1}}{h_{j+1}^z} - \overset{0}{b}_{i,j} \frac{u_{i,j}^{n+1} - u_{i,j-1}^{n+1}}{h_j^z}\right],$$

като сме запазили означението u и за приближеното решение. Във вид, подходящ за прилагане на метода на прогонката, горното уравнение изглежда така:

$$-\frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z} u_{i,j-1}^{n+1} + \left[\frac{c_{i,j}^{n+1}\rho}{\tau} + \frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z} + \frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}\right] u_{i,j}^{n+1} - \frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z} u_{i,j+1}^{n+1} = \frac{c_{i,j}^{n+1}\rho}{\tau} u_{i,j}^{n+1/2}. \tag{4.64}$$

□ *Апроксимация на началното условие (4.35)*

За начално условие на задачата (4.34) — (4.38) в интервала $[t_{n+1/2};\, t_{n+1}]$ приемаме функцията v, която формално можем да разглеждаме като стойности на функцията u на полуслоя $t_{n+1/2}$:

$$u_{i,j}^{n+1/2} = v_{i,j}^{n+1/2}, \;\; j = 1,\, 2,\, \ldots,\, M. \tag{4.65}$$

□ *Апроксимация на граничното условие (4.36)*

От уравнение (4.34) след интегриране в интервала $[z_1;\, z_{3/2}]$ при всяко $t \in [t_{n+1/2};\, t_{n+1}]$, $n = 1,\, 2,\, \ldots,\, S-1$, имаме:

$$\frac{1}{2}\int\limits_{z_1}^{z_{3/2}}\int\limits_{t_{n+1/2}}^{t_{n+1}} c(u)\rho\frac{\partial u}{\partial t}\,\mathrm{d}\,t\,\mathrm{d}\,z = \int\limits_{t_{n+1/2}}^{t_{n+1}}\int\limits_{z_1}^{z_{3/2}} \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial z}\right)\mathrm{d}\,z\,\mathrm{d}\,t,$$

$$\frac{1}{2\hbar_1^z}\int\limits_{z_1}^{z_{3/2}} c(u_i^{n+1/2})\rho\left[u_i^{n+1}(z) - u_i^{n+1/2}(z)\right]\mathrm{d}\,z \approx \frac{1}{\hbar_1^z}\int\limits_{t_{n+1/2}}^{t_{n+1}}\left[W_{i,1}(t) - W_{i,3/2}(t)\right]\mathrm{d}\,t,$$

$$\rho\frac{1}{\hbar_1^z}\int\limits_{z_1}^{z_{3/2}} c(u_i^{n+1/2})\,\mathrm{d}\,z\frac{u_{i,1}^{n+1} - u_{i,1}^{n+1/2}}{\tau} \approx \frac{1}{\hbar_1^z}\left(W_{i,1}^{n+1} - W_{i,3/2}^{n+1}\right).$$

Ако въведем означението

$$c_{i,1}^{n+1/2} = \frac{1}{\hbar_1^z}\int\limits_{z_1}^{z_{3/2}} c(u_i^{n+1/2})\,\mathrm{d}\,z$$

и използваме, че

$$W_{i,1}^{n+1} = -\widetilde{\alpha}\left(u_{i,1}^{n+1} - u_{\text{ок. ср.}}\right),\; W_{i,3/2}^{n+1} \approx -\overset{0}{b}_{i,2}\frac{u_{i,2}^{n+1} - u_{i,1}^{n+1}}{h_2^z},$$

достигаме до следното диференчно уравнение при $j = 1$:

$$c_{i,1}^{n+1/2}\rho\frac{u_{i,1}^{n+1}-u_{i,1}^{n+1/2}}{\tau}=\frac{1}{\hbar_1^z}\left[-\widetilde{\alpha}\left(u_{i,1}^{n+1}-u_{\text{ок. ср.}}\right)+\overset{0}{b}_{i,2}\frac{u_{i,2}^{n+1}-u_{i,1}^{n+1}}{h_2^z}\right].$$

След преобразуване в тридиагонален вид окончателно получаваме

$$\left[\frac{c_{i,1}^{n+1/2}\rho}{\tau}+\frac{\widetilde{\alpha}}{\hbar_1^z}+\frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}\right]u_{i,1}^{n+1}-\frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}u_{i,2}^{n+1}=\frac{c_{i,1}^{n+1/2}\rho}{\tau}u_{i,1}^{n+1/2}+\frac{\widetilde{\alpha}u_{\text{ок. ср.}}}{\hbar_1^z}. \tag{4.66}$$

□ *Апроксимация на контактното условие (4.37)*

От основното уравнение (4.34) чрез интегриране в интервала $[z_{P-1/2};\, z_P]$ за $t \in [t_{n+1/2};\, t_{n+1}]$ получаваме (аналогично на направеното в направление r) диференчното уравнение

$$c_{i,P}^{n+1}\rho\frac{u_{i,P}^{n+1}-u_{i,P}^{n+1/2}}{\tau}=\frac{1}{\hbar_P^z}\left[-\overset{0}{b}_P\frac{u_{i,P}^{n+1}-u_{i,P-1}^{n+1}}{h_P^z}-\alpha_k\left(u_{i,P+1}^{n+1}-u_{i,P}^{n+1}\right)\right], \tag{4.67}$$

където

$$c_{i,P}^{n+1/2}=\frac{1}{\hbar_P^z}\int\limits_{z_{P-1/2}}^{z_P}c(u_i^{n+1/2})\,\mathrm{d}\,z.$$

По същия начин след интегриране в правоъгълника $[z_{P+1};\, z_{P+3/2}] \times [t_{n+1/2};\, t_{n+1}]$ на диференциалното уравнение (4.34) получаваме мрежовото уравнение

$$c_{i,P+2}^{n+1}\rho\frac{u_{i,P+1}^{n+1}-u_{i,P+1}^{n+1/2}}{\tau}=\frac{1}{\hbar_{P+2}^z}\left[-\alpha_k\left(u_{i,P+1}^{n+1}-u_{i,P}^{n+1}\right)+\overset{0}{b}_{i,P+2}\frac{u_{i,P+2}^{n+1}-u_{i,P+1}^{n+1}}{h_{P+2}^z}\right]. \tag{4.68}$$

Тук

$$c_{i,P+2}^{n+1/2}=\frac{1}{\hbar_{P+2}^z}\int\limits_{z_{P+1}}^{z_{P+3/2}}c(u_i^{n+1/2})\,\mathrm{d}\,z.$$

Диференчните уравнения (4.67), (4.68) съвместно дават апроксимацията на контактното условие (4.37). Записваме ги във вида:

$$-\frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}u_{i,P-1}^{n+1}+\left[\frac{c_{i,P}^{n+1/2}\rho}{\tau}+\frac{\alpha_k}{\hbar_P^z}+\frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}\right]u_{i,P}^{n+1}-\frac{\alpha_k}{\hbar_P^z}u_{i,P+1}^{n+1}=\frac{c_{i,P}^{n+1/2}\rho}{\tau}u_{i,P}^{n+1/2}, \tag{4.69а}$$

$$-\frac{\alpha_k}{\hbar_{P+1}^z}u_{i,P}^{n+1}+\left[\frac{c_{i,P+2}^{n+1/2}\rho}{\tau}+\frac{\alpha_k}{\hbar_{P+1}^z}+\frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}\right]u_{i,P+1}^{n+1}-\frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}u_{i,P+2}^{n+1}$$
$$=\frac{c_{i,P+2}^{n+1/2}\rho}{\tau}u_{i,P+1}^{n+1/2}, \tag{4.69б}$$

който вече е подходящ за метода на прогонката.

□ *Апроксимация на горното гранично условие (4.38)*

Накрая след интегриране на уравнението (4.34) в правоъгълника $[z_{M-1/2};\, z_M]\times[t_{n+1/2};\, t_{n+1}]$ по метода на баланса намираме

$$c_{i,M}^{n+1/2}\rho\frac{u_{i,M}^{n+1}-u_{i,M}^{n+1/2}}{\tau}=\frac{1}{\hbar_M^z}\left[-\overset{0}{b}_{i,M}\frac{u_{i,M}^{n+1}-u_{i,M-1}^{n+1}}{h_M^z}-\overline{\alpha}\left(u_{i,M}^{n+1}-u_{\text{ок. ср.}}\right)\right], \tag{4.70}$$

където сме въвели означението

$$c_{i,M}^{n+1/2}=\frac{1}{\hbar_M^z}\int\limits_{z_{M-1/2}}^{z_M}c(u_i^{n+1/2})\,\mathrm{d}\,z.$$

След кратка преработка диференчното уравнение (4.70) добива вида

$$-\frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}u_{i,M-1}^{n+1}+\left[\frac{c_{i,M}^{n+1/2}\rho}{\tau}+\frac{\overline{\alpha}}{\hbar_M^z}+\frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}\right]u_{i,M}^{n+1}=\frac{c_{i,M}^{n+1/2}\rho}{\tau}u_{i,M}^{n+1/2}+\frac{\overline{\alpha}u_{\text{ок. ср.}}}{\hbar_M^z}. \tag{4.71}$$

И така, диференчните уравнения (4.64), (4.65), (4.66), (4.69а), (4.69б), (4.71) водят до следната система от линейни алгебрични уравнения за пресмятане на решението на слоя t_{n+1}:

$$
\left[\frac{c_{i,1}^{n+1/2}\rho}{\tau}+\frac{\widetilde{\alpha}}{\hbar_1^z}+\frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}\right]u_{i,1}^{n+1}-\frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}u_{i,2}^{n+1}=\frac{c_{i,1}^{n+1/2}\rho}{\tau}u_{i,1}^{n+1/2}+\frac{\widetilde{\alpha}u_{\text{ок. ср.}}}{\hbar_1^z},
$$

$$
j=1,\ n=\overline{1,S-1}
$$

$$
-\frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}u_{i,P-1}^{n+1}+\left[\frac{c_{i,P}^{n+1/2}\rho}{\tau}+\frac{\alpha_k}{\hbar_P^z}+\frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}\right]u_{i,P}^{n+1}-\frac{\alpha_k}{\hbar_P^z}u_{i,P+1}^{n+1}
$$

$$
=\frac{c_{i,P}^{n+1/2}\rho}{\tau}u_{i,P}^{n+1/2},\quad j=P,\ n=\overline{1,S-1}
$$

$$
-\frac{\alpha_k}{\hbar_{P+1}^z}u_{i,P}^{n+1}+\left[\frac{c_{i,P+2}^{n+1/2}\rho}{\tau}+\frac{\alpha_k}{\hbar_{P+1}^z}+\frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}\right]u_{i,P+1}^{n+1}-\frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}u_{i,P+2}^{n+1}
$$

$$
=\frac{c_{i,P+2}^{n+1/2}\rho}{\tau}u_{i,P+1}^{n+1/2},\quad j=P+1,\ n=\overline{1,S-1}
$$

$$
-\frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z}u_{i,j-1}^{n+1}+\left[\frac{c_{i,j}^{n+1/2}\rho}{\tau}+\frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z}+\frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}\right]u_{i,j}^{n+1}-\frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}u_{i,j+1}^{n+1}
$$

$$
=\frac{c_{i,j}^{n+1/2}\rho}{\tau}u_{i,j}^{n+1/2},\quad j\neq\{0;\ P;\ P+1;\ M\},\ n=\overline{1,S-1}
$$

$$
-\frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}u_{i,M-1}^{n+1}+\left[\frac{c_{i,M}^{n+1/2}\rho}{\tau}+\frac{\overline{\alpha}}{\hbar_M^z}+\frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}\right]u_{i,M}^{n+1}
$$

$$
=\frac{c_{i,M}^{n+1/2}\rho}{\tau}u_{i,M}^{n+1/2}+\frac{\overline{\alpha}u_{\text{ок. ср.}}}{\hbar_M^z},\quad j=M,\ n=\overline{1,S-1}.
\tag{4.72}
$$

Нейните коефициенти се дават с формулите:

$$C_1 = \left[\frac{c_{i,1}^{n+1/2}\rho}{\tau} + \frac{\widetilde{\alpha}}{\hbar_1^z} + \frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}\right],\ B_1 = -\frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z};$$

$$A_P = -\frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z},\ C_P = \left[\frac{c_{i,P}^{n+1/2}\rho}{\tau} + \frac{\alpha_k}{\hbar_P^z} + \frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}\right],\ B_P = -\frac{\alpha_k}{\hbar_P^z};$$

$$A_{P+1} = -\frac{\alpha_k}{\hbar_{P+1}^z},\ C_{P+1} = \left[\frac{c_{i,P+2}^{n+1/2}\rho}{\tau} + \frac{\alpha_k}{\hbar_{P+1}^z} + \frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}\right],$$

$$B_{P+1} = -\frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z};$$

$$A_M = -\frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z},\ C_M = \left[\frac{c_{i,M}^{n+1/2}\rho}{\tau} + \frac{\overline{\alpha}}{\hbar_M^z} + \frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}\right];$$

$$A_j = -\frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z},\ C_j = \left[\frac{c_{i,j}^{n+1/2}\rho}{\tau} + \frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z} + \frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}\right],$$

$$B_j = -\frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}.$$

Отново проверяваме условията за реализуемост и устойчивост на метода на прогонката. Неравенствата (4.57) и (4.58) са очевидно удовлетворени. За неравенствата (4.59) и (4.60) имаме:

■ $j = 1$:

$$\left|B_1\right| = \frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z},\ \left|C_1\right| = \left[\frac{c_{i,1}^{n+1/2}\rho}{\tau} + \frac{\widetilde{\alpha}}{\hbar_1^z} + \frac{\overset{0}{b}_{i,2}}{\hbar_1^z h_2^z}\right] = \frac{c_{i,1}^{n+1/2}\rho}{\tau} + \frac{\widetilde{\alpha}}{\hbar_1^z} + \left|B_1\right| > \left|B_1\right|;$$

■ $j = M$:

$$|A_M| = \frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z},\ |C_M| = \left[\frac{c_{i,M}^{n+1/2}\rho}{\tau} + \frac{\overline{\alpha}}{\hbar_M^z} + \frac{\overset{0}{b}_{i,M}}{\hbar_M^z h_M^z}\right]$$

$$= \frac{c_{i,M}^{n+1/2}\rho}{\tau} + \frac{\overline{\alpha}}{\hbar_M^z} + |A_M| > |A_M|;$$

■ $j = P$:

$$|A_P| = \frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z},\ |B_P| = \frac{\alpha_k}{\hbar_P^z},\ |C_P| = \left[\frac{c_{i,P}^{n+1/2}\rho}{\tau} + \frac{\alpha_k}{\hbar_P^z} + \frac{\overset{0}{b}_{i,P}}{\hbar_P^z h_P^z}\right]$$

$$= \frac{c_{i,P}^{n+1/2}\rho}{\tau} + |A_P| + |B_P| > |A_P| + |B_P|;$$

■ $j = P + 1$:

$$|A_{P+1}| = \frac{\alpha_k}{\hbar_{P+1}^z},\ |B_{P+1}| = \frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z},\ |C_{P+1}| = \left[\frac{c_{i,P+2}^{n+1/2}\rho}{\tau} + \frac{\alpha_k}{\hbar_{P+1}^z} + \frac{\overset{0}{b}_{i,P+2}}{\hbar_{P+1}^z h_{P+2}^z}\right]$$

$$= \frac{c_{i,P+2}^{n+1/2}\rho}{\tau} + |A_{P+1}| + |B_{P+1}| > |A_{P+1}| + |B_{P+1}|;$$

■ $j \neq \{1;\ P;\ P+1;\ M\}$:

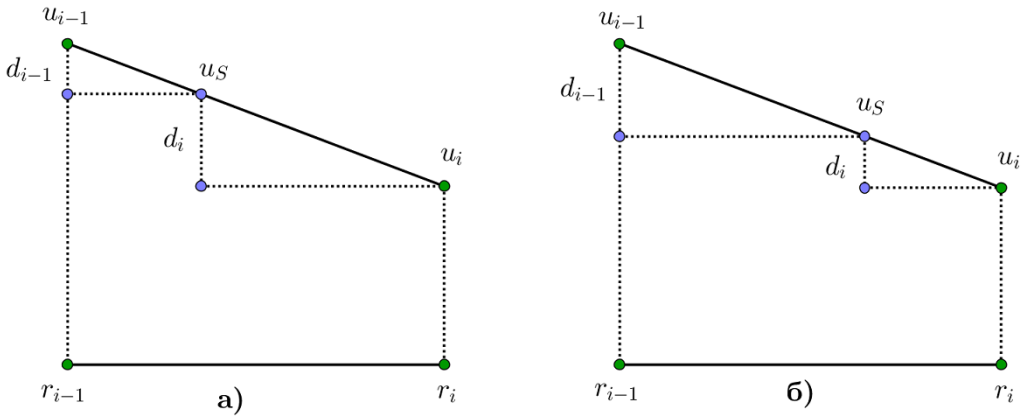
$$|A_j| = \frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z},\ |B_j| = \frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z},\ |C_j| = \left[\frac{c_{i,j}^{n+1/2}\rho}{\tau} + \frac{\overset{0}{b}_{i,j}}{\hbar_j^z h_j^z} + \frac{\overset{0}{b}_{i,j+1}}{\hbar_j^z h_{j+1}^z}\right]$$

$$= \frac{c_{i,j}^{n+1/2}\rho}{\tau} + |A_j| + |B_j| > |A_j| + |B_j|.$$

Получихме, че всички условия на теорема 1 са изпълнени. Следователно диференчната задача в направление z (4.72) притежава единствено решение u^{n+1}. Редът на сходимост на построената линеаризирана чисто неявна локално едномерна схема (4.55), (4.72) е $O\left(\left\|h^r\right\|^2 + \left\|h^z\right\|^2 + \tau\right)$, където $\|\cdot\|$ има смисъл на евклидова норма, а h^r и h^z са векторите от дискретните стъпки по всяко от пространствените направления.

4.4 Изглаждане на делта-функцията

Да се спрем сега на „изгладената“ или т. нар. приблизително делта-образна функция, за която вече стана дума. Тя е функция на единствен аргумент Δ и както е показано в статията [4], този аргумент трябва всеки път да се избира по такъв начин, че температурният интервал $\left[u_S - \Delta;\, u_S + \Delta\right]$ да обхваща 1, 2 или 3 точки от мрежата. Ще покажем прост алгоритъм за избора на Δ, при който споменатият интервал съдържа точно една точка от мрежата. Единият от краищата му винаги ще съвпада с цял възел. Наличието на изгладена делта-функция означава, че за конкретния момент от време съществува поне една точка от областта $\overline{D}$, в която се е достигнала температурата на солидуса u_S. Тогава за пресмятането на коефициента $c(u)$ директно прилагаме (4.18). Обратно — ако u_S не се достига за този момент от време, при пресмятането на коефициента $c(u)$ трябва да се позовем на формулата (4.1).



Фигура 4.7: *Пресмятане на параметъра Δ в случая на намаляваща мрежова функция, когато:* ***а)*** $d_{i-1} < d_i$; ***б)*** $d_{i-1} > d_i$

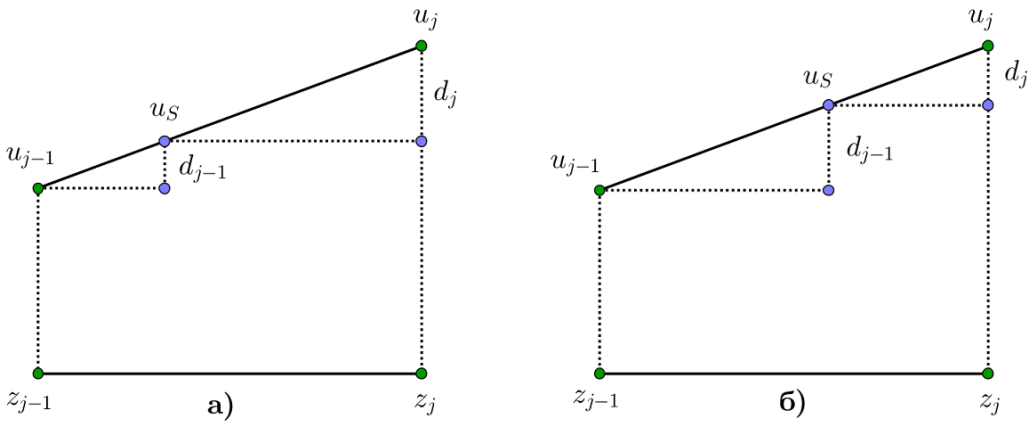
Без ограничение на общността нека сме в направление r. Изпълнено е, че мрежовата функция е намаляваща. Инедксите, отговарящи на променливите z и t, ще пропускаме и ще считаме известни. Предполагаме, че сме установили местоположението на температурата солидус между две съседни точки u_{i-1}, u_i, $i = 2, 3, \ldots, N$, от неравномерната мрежа $\widehat{\omega}^*_{h^r}$. Алгоритъмът е следният.

1. Пресмятаме температурните разлики $d_{i-1} = u_{i-1} - u_S$ и $d_i = u_S - u_i$.
2. Сравняваме d_{i-1} и d_i (вж. фиг. 4.7 **а)** и **б)**):
   - ☐ ако $d_{i-1} < d_i$, полагаме $\Delta = u_{i-1} - u_S$;

☐ ако $d_i < d_{i-1}$, полагаме $\Delta = u_S - u_i$.

Нека сега, обратно, пресмятаме решението u в направление z. Ще считаме, че индексите i и n са фиксирани. В околност на равнината $z = H$ мрежовата функция u е растяща. За определяне на параметъра Δ ще използваме същата идея, както по-горе. Измежду точките u_j, $j = 2, 3, \ldots, M$, търсим две съседни u_{j-1}, u_j, такива, че температурата u_S да е между тях.

1. Както в случая на намаляваща функция, пресмятаме разликите $d_{j-1} = u_S - u_{j-1}$ и $d_j = u_j - u_S$.
2. Сравняваме d_{j-1} и d_j (вж. фиг. 4.8 **а)** и **б)**):
   - ☐ ако $d_{j-1} < d_j$, полагаме $\Delta = u_S - u_{j-1}$;
   - ☐ ако $d_{j-1} > d_j$, полагаме $\Delta = u_j - u_S$.



Фигура 4.8: *Пресмятане на параметъра Δ в случая на растяща мрежова функция, когато:* ***а)*** *$d_{j-1} < d_j$;* ***б)*** *$d_{j-1} > d_j$*

И така, без значение дали мрежовата функция u е монотонно растяща, или монотонно намаляваща, параметъра Δ пресмятаме така: $\Delta = \min\{d_{i-1};\, d_i\}$ (или $\Delta = \min\{d_{j-1};\, d_j\}$).

Оттук нататък със $\widetilde{c}$ ще означаваме „изгладения“ коефициент на топлоемкост в интервала $(u_S - \Delta;\, u_S + \Delta)$ [4]:

$$\widetilde{c} = \frac{\eta}{2\Delta} + c_S - \frac{\kappa}{2\Delta}\Big[\psi(u_S + \Delta) - \psi(u_S)\Big].$$

Тъй като стойностите на параметрите c_S, κ и η са фиксирани, а параметърът Δ се определя в хода на алгоритъма, то $\widetilde{c}$ е функция само на променливата Δ: $\widetilde{c} = \widetilde{c}(\Delta)$.

4.5 Пресмятане на интегралните коефициенти $c_{i,j}$

Сега ще покажем как ще бъдат пресмятани интегралните коефициенти $\frac{1}{\hbar_i^r}\int\limits_{r_{i-1/2}}^{r_{i+1/2}} c(u)\,\mathrm{d}\,r$ и $\frac{1}{\hbar_j^z}\int\limits_{z_{j-1/2}}^{z_{j+1/2}} c(u)\,\mathrm{d}\,z$, като се вземе предвид наличието на подвижната фазова граница $u = u_S$. Преди всичко да отбележим следните особености:

1. При движение в направление r подинтегралната функция се взема на долния слой t_n, $n = 0, 1, \ldots, S-1$, а при движение в направление z — на полуслоя $t_{n+1/2}$, $n = 0, 1, \ldots, S-1$.

2. Интегралните граници $r_{i\pm1/2}$, респ. $z_{j\pm1/2}$, са в непрекъсната зависимост от разположението на точките $r_{S\pm\Delta}$ и r_L, където последните са ортогоналните проекции на точките $u_S \pm \Delta$ и u_L върху хоризонталната ос Or.

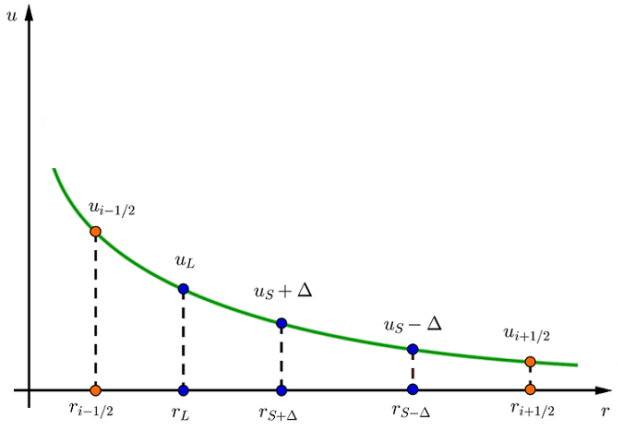
3. Радиалните координати $r_{S\pm\Delta}$ и r_L не са предварително известни, а трябва да се пресмятат на всяка стъпка от алгоритъма. Това може да стане с помощта на най-простата интерполация — линейната.

4. Мрежовата функция в направление z е намаляваща в цялата си дефиниционна област, с изключение на околността на равнината $z = H$ и в областта на формата.

Има 10 възможни разположения на полуцелите възли относно споменатите хоризонтални проекции. Ние ще се спрем на онова разположение, което представлява най-голяма изчислителна и алгоритмична трудност. Изследването на останалите случаи се извършва по аналогичен начин и затова ще се ограничим само с представянето на съответните формули. И така, без ограничение на общността нека сме в радиално направление. Тогава мрежовата функция е намаляваща. Разположението на интегралните граници е показано на фиг. 4.9. Конкретните стойности на $r_{S\pm\Delta}$ и r_L определяме така:

1. С линейна интерполация по точките $(r_{i-1/2};\, u_{i-1/2})$ и $(r_{i+1/2};\, u_{i+1/2})$ изчисляваме r_L. Тук $u_{i-1/2} = \dfrac{u_{i-1} + u_i}{2}$ и $u_{i+1/2} = \dfrac{u_i + u_{i+1}}{2}$.

2. След това с линейна интерполация по точките $(r_L;\, u_L)$ и $(r_{i+1/2};\, u_{i+1/2})$ получаваме $r_{S+\Delta}$.



Фигура 4.9: *Разположение на полуцелите възли* $r_{i\pm1/2}$ *спрямо точките* $r_{S\pm\Delta}$ *и* r_L

3. Накрая отново с линейна интерполация по точките $(r_{S+\Delta};\, u_S+\Delta)$ и $(r_{i+1/2};\, u_{i+1/2})$ намираме стойността на $r_{S-\Delta}$.

По-нататък: като вземем предвид формулата (4.18), получаваме последователно:

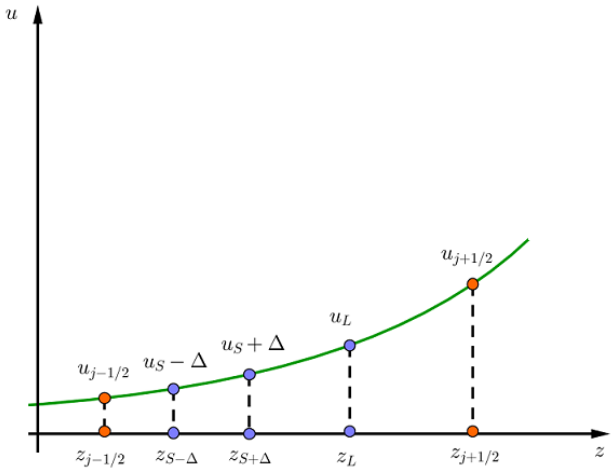
$$\int\limits_{r_{i-1/2}}^{r_{i+1/2}} c(u)\,\mathrm{d}\,r$$

$$= \int\limits_{r_{i-1/2}}^{r_L} c(u)\,\mathrm{d}\,r + \int\limits_{r_L}^{r_{S+\Delta}} c(u)\,\mathrm{d}\,r + \int\limits_{r_{S+\Delta}}^{r_{S-\Delta}} c(u)\,\mathrm{d}\,r + \int\limits_{r_{S-\Delta}}^{r_{i+1/2}} c(u)\,\mathrm{d}\,r$$

$$= \int\limits_{r_{i-1/2}}^{r_L} c_L\,\mathrm{d}\,r + \int\limits_{r_L}^{r_{S+\Delta}} \left[c_S - \kappa\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right]\mathrm{d}\,r + \int\limits_{r_{S+\Delta}}^{r_{S-\Delta}} \widetilde{c}\,\mathrm{d}\,r + \int\limits_{r_{S-\Delta}}^{r_{i+1/2}} c_S\,\mathrm{d}\,r$$

$$= c_L(r_L - r_{i-1/2}) \;+\; c_S(r_{S+\Delta} - r_L) \;-\; \kappa\frac{r_{S+\Delta}-r_L}{2}\left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right|_{u_S+\Delta} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right|_{u_L}\right)$$
$$+\; \widetilde{c}(r_{S-\Delta} - r_{S+\Delta}) \;+\; c_S(r_{i+1/2} - r_{S-\Delta}).$$

Да извършим сега същите разглеждания, но когато се придвижваме в направление z. Конкретното разположение на интегралните граници



Фигура 4.10: *Разположение на полуцелите възли $z_{j\pm1/2}$ спрямо точките $z_{S\pm\Delta}$ и z_L*

относно точките $z_{S\pm\Delta}$ и z_L е изобразено на фиг. 4.10. В този случай имаме:

$$\int_{z_{j-1/2}}^{z_{j+1/2}} c(u)\,\mathrm{d}\,z$$

$$= \int_{z_{j-1/2}}^{z_{S-\Delta}} c(u)\,\mathrm{d}\,z + \int_{z_{S-\Delta}}^{z_{S+\Delta}} c(u)\,\mathrm{d}\,z + \int_{z_{S+\Delta}}^{z_L} c(u)\,\mathrm{d}\,z + \int_{z_L}^{z_{j+1/2}} c(u)\,\mathrm{d}\,z$$

$$= \int_{z_{j-1/2}}^{z_{S-\Delta}} c_S\,\mathrm{d}\,z + \int_{z_{S-\Delta}}^{z_{S+\Delta}} \widetilde{c}\,\mathrm{d}\,z + \int_{z_{S+\Delta}}^{z_L} \left[c_S - \kappa\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right]\mathrm{d}\,z + \int_{z_L}^{z_{j+1/2}} c_L\,\mathrm{d}\,z$$

$$= c_S(z_{S-\Delta} - z_{j-1/2}) \;+\; \widetilde{c}(z_{S+\Delta} \;-\; z_{S-\Delta}) \;+\; c_S(z_L - z_{S+\Delta})$$

$$- \kappa\frac{z_L - z_{S+\Delta}}{2}\left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right|_{u_L} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right|_{u_S+\Delta}\right) \;+\; c_L(z_{j+1/2} - z_L).$$

Таблици 4.2 и 4.3 обобщават цялостно информацията за пресмятането на интегралните коефициенти $c_{i,j}$. В първия стълб на всяка от тях са представени възможните разположения на стойностите на температурите в полуцелите възли относно специалните температури $u_S \pm \Delta$

и u_L, а във втория — съответните алгебрични формули за приближено пресмятане. Таблиците са конструирани така, че да вземат предвид свойството функционална монотонност. Синтезираните резултати са получени въз основа на това, дали в текущия момент от време по даденото направление се достига температурата u_S. Ако u_S не се достига никъде на текущата стъпка от алгоритъма, точките $u_S \pm \Delta$ не са определени и използването на изгладена делта-функция не е възможно. Обратно, ако съществува поне една точка от пространството, в която температурата е равна на солидуса, въвеждаме изгладена делта-функция за пресмятането на интегралните коефициенти. Споменатата особеност се реализира чрез въвеждането на допълнителен параметър n_S — брой точки от мрежата, в които температурата не превишава u_S.

Таблица 4.2: *Пресмятане на коефициента* $\int_{r_{i-1/2}}^{r_{i+1/2}} c(u)\,\mathrm{d}\,r$ *при намаляваща функция*

Намаляваща функция	Коефициент
$n_S = 0$	
1, а): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \geq u_L \end{cases}$	$c_L(r_{i+1/2} - r_{i-1/2})$
1, б): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \in [u_S;\, u_L] \end{cases}$	$c_L(r_L - r_{i-1/2}) + c_S(r_{i+1/2} - r_L) - \kappa\frac{r_{i+1/2} - r_L}{2}\left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i+1/2}}\right)$
1, в): $\begin{cases} u_{i-1/2} \in [u_S;\, u_L] \\ u_{i+1/2} \in [u_S;\, u_L] \end{cases}$	$c_S(r_{i+1/2} - r_{i-1/2}) - \kappa\frac{r_{i+1/2} - r_{i-1/2}}{2}\left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i-1/2}} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i+1/2}}\right)$
$n_S \neq 0$	
2, а): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \geq u_L \end{cases}$	$c_L(r_{i+1/2} - r_{i-1/2})$
2, б): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \in [u_S + \Delta;\, u_L] \end{cases}$	$c_L(r_L - r_{i-1/2}) + c_S(r_{i+1/2} - r_L) - \kappa\frac{r_{i+1/2} - r_L}{2}\left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i+1/2}}\right)$

Продължава на следващата страница

Продължава от предишната страница

Намаляваща функция	**Коефициент**
2, в): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \in [u_S - \Delta;\, u_S + \Delta] \end{cases}$	$c_L(r_L - r_{i-1/2}) + c_S(r_{S+\Delta} - r_L) - \kappa \frac{r_{S+\Delta} - r_L}{2} \left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} \right) + \widetilde{c}(r_{i+1/2} - r_{S+\Delta})$
2, г): $\begin{cases} u_{i-1/2} \geq u_L \\ u_{i+1/2} \leq u_S - \Delta \end{cases}$	$c_L(r_L - r_{i-1/2}) + c_S(r_{S+\Delta} - r_L) - \kappa \frac{r_{S+\Delta} - r_L}{2} \left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} \right) + \widetilde{c}(r_{S-\Delta} - r_{S+\Delta}) + c_S(r_{i+1/2} - r_{S-\Delta})$
2, д): $\begin{cases} u_{i-1/2} \in [u_S + \Delta;\, u_L] \\ u_{i+1/2} \in [u_S + \Delta;\, u_L] \end{cases}$	$c_S(r_{i+1/2} - r_{i-1/2}) - \kappa \frac{r_{i+1/2} - r_{i-1/2}}{2} \left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i-1/2}} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i+1/2}} \right)$
2, е): $\begin{cases} u_{i-1/2} \in [u_S + \Delta;\, u_L] \\ u_{i+1/2} \in [u_S - \Delta;\, u_S + \Delta] \end{cases}$	$c_S(r_{S+\Delta} - r_{i-1/2}) - \kappa \frac{r_{S+\Delta} - r_{i-1/2}}{2} \left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i-1/2}} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} \right) + \widetilde{c}(r_{i+1/2} - r_{S+\Delta})$
2, ж): $\begin{cases} u_{i-1/2} \in [u_S + \Delta;\, u_L] \\ u_{i+1/2} \leq u_S - \Delta \end{cases}$	$c_S(r_{S+\Delta} - r_{i-1/2}) - \kappa \frac{r_{S+\Delta} - r_{i-1/2}}{2} \left(\left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{i-1/2}} + \left.\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} \right) + \widetilde{c}(r_{S-\Delta} - r_{S+\Delta}) + c_S(r_{i+1/2} - r_{S-\Delta})$
2, з): $\begin{cases} u_{i-1/2} \in [u_S - \Delta;\, u_S + \Delta] \\ u_{i+1/2} \in [u_S - \Delta;\, u_S + \Delta] \end{cases}$	$\widetilde{c}(r_{i+1/2} - r_{i-1/2})$
2, и): $\begin{cases} u_{i-1/2} \in [u_S - \Delta;\, u_S + \Delta] \\ u_{i+1/2} \leq u_S - \Delta \end{cases}$	$\widetilde{c}(r_{S-\Delta} - r_{i-1/2}) + c_S(r_{i+1/2} - r_{S-\Delta})$
2, й): $\begin{cases} u_{i-1/2} \leq u_S - \Delta \\ u_{i+1/2} \leq u_S - \Delta \end{cases}$	$c_S(r_{i+1/2} - r_{i-1/2})$

Таблица 4.3: *Пресмятане на коефициента* $\int_{z_{j-1/2}}^{z_{j+1/2}} c(u)\,\mathrm{d}\,z$ *при растяща функция*

Растяща функция	Коефициент
$n_S = 0$	
1, а): $\begin{cases} u_{j-1/2} \in [u_S;\, u_L] \\ u_{j+1/2} \in [u_S;\, u_L] \end{cases}$	$c_S(z_{j+1/2} - z_{j-1/2}) -$ $\kappa \frac{z_{j+1/2} - z_{j-1/2}}{2} \left(\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_{j+1/2}} + \frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_{j-1/2}} \right)$
1, б): $\begin{cases} u_{j-1/2} \in [u_S;\, u_L] \\ u_{j+1/2} \geq u_L \end{cases}$	$c_S(z_L - z_{j-1/2}) -$ $\kappa \frac{z_L - z_{j-1/2}}{2} \left(\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_{j-1/2}} + \frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_L} \right)$ $+ c_L(z_{j+1/2} - z_L)$
1, в): $\begin{cases} u_{j-1/2} \geq u_L \\ u_{j+1/2} \geq u_L \end{cases}$	$c_L(z_{j+1/2} - z_{j-1/2})$
$n_S \neq 0$	
2, а): $\begin{cases} u_{j-1/2} \leq u_S - \Delta \\ u_{j+1/2} \leq u_S - \Delta \end{cases}$	$c_S(z_{j+1/2} - z_{j-1/2})$
2, б): $\begin{cases} u_{j-1/2} \leq u_S - \Delta \\ u_{j+1/2} \in [u_S - \Delta;\, u_S + \Delta] \end{cases}$	$c_S(z_{S-\Delta} - z_{j-1/2}) + \widetilde{c}(z_{j+1/2} - z_{S-\Delta})$
2, в): $\begin{cases} u_{j-1/2} \leq u_S - \Delta \\ u_{j+1/2} \in [u_S + \Delta;\, u_L] \end{cases}$	$c_S(z_{S-\Delta} - z_{j-1/2}) +$ $\widetilde{c}(z_{S+\Delta} - z_{S-\Delta}) + c_S(z_{j+1/2} - z_{S+\Delta}) -$ $\kappa \frac{z_{j+1/2} - z_{S+\Delta}}{2} \left(\frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_{S+\Delta}} + \frac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\Big\vert_{u_{j+1/2}} \right)$

Продължава на следващата страница

Продължава от предишната страница

Растяща функция	**Коефициент**
2, г): $\begin{cases} u_{j-1/2} \le u_S - \Delta \\ u_{j+1/2} \ge u_L \end{cases}$	$c_S(z_{S-\Delta} - z_{j-1/2}) +$ $\widetilde{c}(z_{S+\Delta} - z_{S-\Delta}) + c_S(z_L - z_{S+\Delta})$ $- \kappa \dfrac{z_L - z_{S+\Delta}}{2} \left(\left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} + \left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} \right)$ $+ c_L(z_{j+1/2} - z_L)$
2, д): $\begin{cases} u_{j-1/2} \in [u_S - \Delta;\, u_S + \Delta] \\ u_{j+1/2} \in [u_S - \Delta;\, u_S + \Delta] \end{cases}$	$\widetilde{c}(z_{j+1/2} - z_{j-1/2})$
2, е): $\begin{cases} u_{j-1/2} \in [u_S - \Delta;\, u_S + \Delta] \\ u_{j+1/2} \in [u_S + \Delta;\, u_L] \end{cases}$	$\widetilde{c}(z_{S+\Delta} - z_{j-1/2}) + c_S(z_{j+1/2} - z_{S+\Delta}) -$ $\kappa \dfrac{z_{j+1/2} - z_{S+\Delta}}{2} \left(\left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} + \left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{j+1/2}} \right)$
2, ж): $\begin{cases} u_{j-1/2} \in [u_S - \Delta;\, u_S + \Delta] \\ u_{j+1/2} \ge u_L \end{cases}$	$\widetilde{c}(z_{S+\Delta} - z_{j-1/2}) + c_S(z_L - z_{S+\Delta}) -$ $\kappa \dfrac{z_L - z_{S+\Delta}}{2} \left(\left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_S+\Delta} + \left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} \right)$ $+ c_L(z_{j+1/2} - z_L)$
2, з): $\begin{cases} u_{j-1/2} \in [u_S + \Delta;\, u_L] \\ u_{j+1/2} \in [u_S + \Delta;\, u_L] \end{cases}$	$c_S(z_{j+1/2} - z_{j-1/2}) -$ $\kappa \dfrac{z_{j+1/2} - z_{j-1/2}}{2} \left(\left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{j-1/2}} + \left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{j+1/2}} \right)$
2, и): $\begin{cases} u_{j-1/2} \in [u_S + \Delta;\, u_L] \\ u_{j+1/2} \ge u_L \end{cases}$	$c_S(z_L - z_{j-1/2}) -$ $\kappa \dfrac{z_L - z_{j-1/2}}{2} \left(\left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_{j-1/2}} + \left.\dfrac{\mathrm{d}\,\psi}{\mathrm{d}\,u}\right\|_{u_L} \right)$ $+ c_L(z_{j+1/2} - z_L)$
2, й): $\begin{cases} u_{j-1/2} \ge u_L \\ u_{j+1/2} \ge u_L \end{cases}$	$c_L(z_{j+1/2} - z_{j-1/2})$

4.6 Методически изследвания

Ще конструираме тестови примери, които са едномерни по всяко едно от пространствените направления. С тяхна помощ ще проверим верността на изведените схеми (4.55) и (4.72), както и програмната им реализация.

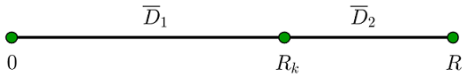
4.6.1 Тестов пример в направление r

В областта $\overline{D} = \overline{D}_1 \cup \overline{D}_2$ (вж. фиг. 4.11), където

$$\overline{D}_1 = \left\{ r : \, 0 \leq r \leq R_k \right\} \text{ (метал)},$$

$$\overline{D}_2 = \left\{ r : \, R_k \leq r \leq R \right\} \text{ (форма)},$$

разглеждаме едномерното по пространството уравнение



Фигура 4.11: *Схематично представяне на областта* $\overline{D} = \overline{D}_1 \cup \overline{D}_2$

$$c(u)\rho(u)\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(u)\frac{\partial u}{\partial r} \right) + f(r,t), \;\; r \in D, \;\; t \in (0;\, T], \tag{4.73}$$

при начално условие

$$u(r,0) = u_0(r), \;\; r \in \overline{D}, \tag{4.74}$$

условие за симетрия

$$\lim_{r \to 0} r\lambda(u)\frac{\partial u}{\partial r} = 0, \tag{4.75}$$

условие за неидеален контакт върху границата $r = R_k$

$$\lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R_k^-} = \lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R_k^+} = \alpha_k^{(1)}\left(u\Big|_{r=R_k^+} - u\Big|_{r=R_k^-} \right), \;\; \alpha_k^{(1)} > 0, \tag{4.76}$$

и гранично условие

$$\lambda(u)\frac{\partial u}{\partial r}\bigg|_{r=R} = -\alpha\left(u\Big|_{r=R} - u_{\text{ок. ср.}}\right), \quad \alpha \geq 0. \tag{4.77}$$

Геометричните размери на областта $\overline{D}$, стойностите на температурите u_L и u_S, стойностите на топлофизичните характеристики c, ρ, λ, както и крайното време T са дадени в таблица 4.4.

Параметър	Числена стойност
R_k; R	2; 3
u_L; u_S	62; 59
c_1; c_2	10; 11
ρ_1; ρ_2	25; 80
λ_1; λ_2	2; 1
T	1

Таблица 4.4: *Геометрични параметри и топлофизични характеристики на едномерната в направление r задача* (4.73) – (4.77)

Нека $\psi(u)$ е линейна функция на температурата:

$$\psi(u) = \begin{cases} 1, & u \leq u_S, \\ \dfrac{u_L - u}{u_L - u_S}, & u_S \leq u \leq u_L, \\ 0, & u \geq u_L. \end{cases} \tag{4.78}$$

Търсим решение от вида

$$u(r,t) = \begin{cases} u_1(r,t), & r \in \overline{D}_1, \\ u_2(r,t), & r \in \overline{D}_2 \end{cases} = \begin{cases} a_1 t^2 + b_1 r^3 + d_1 R_k^3, & r \in \overline{D}_1, \\ a_2 t^2 + b_2 r^3 + d_2 R_k^3, & r \in \overline{D}_2, \end{cases} \tag{4.79}$$

където коефициентите a_m, b_m, d_m, $m = 1, 2$, и дясната част $f(r,t)$ засега са неизвестни. Освен това искаме още да са изпълнени условията

$$a_m < 0, \;\; b_m < 0, \;\; d_m R_k^3 > -a_m T^2 - b_m R^3. \tag{4.80}$$

При фиксирана стойност на z те осигуряват монотонното намаляване на решението $u(r,t)$ и неговата положителност.

За дясната част на уравнение (4.73) получаваме

$$f(r,t) = c(u)\rho(u)\frac{\partial u}{\partial t} - \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda(u)\frac{\partial u}{\partial r}\right) = \begin{cases} 2c_1\rho_1 a_1 t - 9\lambda_1 b_1 r, & r \in \overline{D}_1, \\ 2c_2\rho_2 a_2 t - 9\lambda_2 b_2 r, & r \in \overline{D}_2. \end{cases} \tag{4.81}$$

Началното условие (4.74) има вида

$$u(r,0) = \begin{cases} b_1 r^3 + d_1 R_k^3, & r \in \overline{D}_1, \\ b_2 r^3 + d_2 R_k^3, & r \in \overline{D}_2. \end{cases} \tag{4.82}$$

Условието (4.75) е изпълнено:

$$\lim_{r\to 0} r\lambda(u)\frac{\partial u}{\partial r} = \lim_{r\to 0} \begin{cases} 3\lambda_1 b_1 r^3, & r \in \overline{D}_1, \\ 3\lambda_2 b_2 r^3, & r \in \overline{D}_2 \end{cases} = 0. \tag{4.83}$$

От първото равенство в (4.76) имаме

$$3\lambda_1 b_1 \left(R_k^-\right)^2 = 3\lambda_2 b_2 \left(R_k^+\right)^2.$$

Следователно, ако b_1 е дадено число, то b_2 се определя еднозначно по формулата

$$b_2 = \frac{\lambda_1 b_1}{\lambda_2}.$$

Изпълнено е още

$$\alpha_k^{(1)}\left(u\Big|_{r=R_k^+} - u\Big|_{r=R_k^-}\right) = \alpha_k^{(1)}\Big[(a_2-a_1)t^2 + (b_2-b_1)R_k^3 + (d_2-d_1)R_k^3\Big],$$

т. е.

$$3\lambda_1 b_1 \left(R_k^-\right)^2 = \alpha_k^{(1)}\Big[(a_2-a_1)t^2 + (b_2-b_1)R_k^3 + (d_2-d_1)R_k^3\Big].$$

Лявата страна не зависи от t, следователно и дясната не трябва да зависи от t и затова избираме $a_1 = a_2 = a \in \mathbb{R}^-$. Тогава имаме

$$3\lambda_1 b_1 \left(R_k^-\right)^2 = \alpha_k^{(1)}\Big[(b_2-b_1)R_k^3 + (d_2-d_1)R_k^3\Big]$$

и за $\alpha_k^{(1)}$ получаваме

$$\alpha_k^{(1)} = \frac{3\lambda_1 b_1}{R_k\Big[(b_2-b_1) + (d_2-d_1)\Big]}.$$

Искаме $\alpha_k^{(1)} > 0$, което е равносилно на неравенството

$$b_2 - b_1 + d_2 - d_1 < 0.$$

По-нататък: за да е изпълнено (4.77), трябва

$$3\lambda_2 b_2 R^2 = -\alpha \left(a_2 t^2 + b_2 R^3 + d_2 R_k^3 - u_{\text{ок. ср.}} \right).$$

Както преди, понеже лявата страна не е функция на t, то и дясната не трябва да е функция на t. Освен това трябва

$$a_2 t^2 + b_2 R^3 + d_2 R_k^3 - u_{\text{ок. ср.}} > 0$$

и затова избираме

$$u_{\text{ок. ср.}} = a_2 t^2 + b_2 R^3.$$

Тогава за коефициента α получаваме формулата

$$\alpha = -\frac{3\lambda_2 b_2 R^2}{d_2 R_k^3} \geq 0.$$

Параметър	Числена стойност
$a_1 = a_2$	-1
b_1; b_2	-1; -2
d_1; d_2	8; 7
$\alpha_k^{(1)}$; α	2; 27/28

Таблица 4.5: *Стойности на допълнителните параметри a_m, b_m, d_m $(m = 1, 2)$ и на коефициентите $\alpha_k^{(1)}$, α за едномерната в направление r задача* (4.73) – (4.77)

Стойностите на параметрите a_m, b_m и d_m, $m = 1, 2$, както и на коефициентите на топлообмен $\alpha_k^{(1)}$ и α, са обобщени в таблица 4.5. За така определените a_m, b_m и d_m формулата (4.79) добива вида

$$u(r,t) = \begin{cases} -t^2 - r^3 + 64, & r \in \overline{D}_1, \\ -t^2 - 2r^3 + 56, & r \in \overline{D}_2. \end{cases} \tag{4.84}$$

4.6.2 Тестов пример в направление z

В областта $\overline{D} = \overline{D}_1 \cup \overline{D}_2$ (вж. фиг. 4.12), където

$$\overline{D}_1 = \Big\{ z : \, h \leq z \leq H \Big\} \ \text{(метал)},$$

$$\overline{D}_2 = \Big\{ z : \, 0 \leq z \leq h \Big\} \ \text{(форма)},$$

търсим решение на задачата

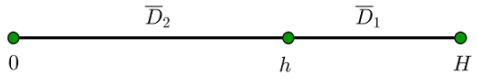
$$c(u)\rho(u)\frac{\partial u}{\partial t} = \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial z}\right) + g(z,t), \ \ z \in D, \ \ t \in (0;\, T], \tag{4.85}$$

$$u(z,0) = u_0(z), \ \ z \in \overline{D}, \tag{4.86}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=0} = \widetilde{\alpha}\Big(u\Big|_{z=0} - u^0_{\text{ок. ср.}}\Big), \tag{4.87}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^-} = \lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^+} = \alpha_k\left(u\Big|_{z=h^+} - u\Big|_{z=h^-} + x\right), \tag{4.88}$$

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=H} = -\overline{\alpha}\left(u\Big|_{z=H} - u^H_{\text{ок. ср.}}\right), \tag{4.89}$$



Фигура 4.12: *Схематично представяне на областта* $\overline{D} = \overline{D}_1 \cup \overline{D}_2$

където функцията $\psi(u)$ отново е линейна функция на температурата, а добавената в (4.88) променлива x улеснява конструирането на аналитично решение. В условия (4.87) и (4.89) приемаме, че температурата на околната среда е различна в граничните точки $z = 0$ и $z = H$.

Накрая, като вземем предвид познатия начин на задаване на коефициента на топлоемкост $c(u)$ и формулата (4.78) при $u_S \le u \le u_L$, получаваме

$$c(u) = \begin{cases} c_1, & u \ge u_L, \ \ z \in \overline{D}_1, \\ c_1 + \dfrac{\kappa}{u_L - u_S}, & u_S + \Delta < u \le u_L, \ \ z \in \overline{D}_1, \\ c_1 + \dfrac{\eta}{2\Delta} + \dfrac{\kappa}{2(u_L - u_S)}, & u_S - \Delta < u < u_S + \Delta, \ \ z \in \overline{D}_1, \\ c_1, & u < u_S - \Delta, \ \ z \in \overline{D}_1. \end{cases}$$

В таблица 4.6 сме синтезирали основните характеристики на областта $\overline{D}$ и на сплавта, както и стойността на крайното време T.

Параметър	**Числена стойност**
h; H	1; 3
u_L; u_S	50; 30
c_1; c_2	10; 11
ρ_1; ρ_2	25; 80
λ_1; λ_2	2; 1
T	1

Таблица 4.6: *Геометрични параметри и топлофизични характеристики на едномерната в направление z задача* (4.85) — (4.89)

И така, нека аналитичното решение на задача (4.85) — (4.89) да бъде

$$u(z,t) = \begin{cases} u_1(z,t), & z \in \overline{D}_1, \\ u_2(z,t), & z \in \overline{D}_2 \end{cases} = \begin{cases} -t^2 + 2z^3 + 12, & z \in \overline{D}_1, \\ t^2 + z^3 + 2, & z \in \overline{D}_2. \end{cases} \tag{4.90}$$

За дясната част получаваме

$$g(z,t) = c(u)\rho(u)\frac{\partial u}{\partial t} - \frac{\partial}{\partial z}\left(\lambda(u)\frac{\partial u}{\partial z}\right) = \begin{cases} -2c(u)\rho_1 t - 12\lambda_1 z, & z \in \overline{D}_1, \\ 2c_2\rho_2 t - 6\lambda_2 z, & z \in \overline{D}_2. \end{cases}$$

Началното условие (4.86) изглежда така:

$$u(z,0) = \begin{cases} 2z^3 + 12, & z \in \overline{D}_1, \\ z^3 + 2, & z \in \overline{D}_2. \end{cases}$$

За условие (4.87) имаме

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=0} = 0.$$

Тогава, ако положим $\widetilde{\alpha} = 0$, то $u^0_{\text{ок. ср.}}$ можем да изберем произволно.

По-нататък: за (4.88) имаме

$$\lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^-} = 3\lambda_2 h^2, \;\; \lambda(u)\frac{\partial u}{\partial z}\bigg|_{z=h^+} = 6\lambda_1 h^2$$

и следователно $3\lambda_2 h^2 = 6\lambda_1 h^2$. Оттук получаваме връзката

$$\lambda_2 = 2\lambda_1.$$

Имаме още, че

$$\alpha_k\left(-t^2 + 2h^3 + 12 - t^2 - h^3 - 2 + x\right) = \alpha_k\left(-2t^2 + h^3 + 10 + x\right),$$

и ако изберем $x = 2t^2 - h^3 - 9$, ще получим $3\lambda_2 h^2 = 6\lambda_1 h^2 = \alpha_k$. Следователно

$$\alpha_k = 6\lambda_1 h^2.$$

За да бъде удовлетворено условие (4.89), е необходимо да е изпълнено

$$6\lambda_1 H^2 = -\overline{\alpha}\left(-t^2 + 2H^3 + 12 - u^H_{\text{ок. ср.}}\right).$$

Тогава, ако $\overline{\alpha}$ е дадено, за $u^H_{\text{ок. ср.}}$ получаваме

$$u^H_{\text{ок. ср.}} = -t^2 + 2H^3 + 12 + \frac{6\lambda_1 H^2}{\overline{\alpha}}.$$

Таблица 4.7 представя обобщена информация за стойностите на коефициентите на топлообмен α_k, $\widetilde{\alpha}$ и $\overline{\alpha}$ на задача (4.85) — (4.89).

Коефициент	**Числена стойност**
α_k; $\widetilde{\alpha}$; $\overline{\alpha}$	6; 0; 1

Таблица 4.7: *Стойности на коефициентите α_k, $\widetilde{\alpha}$ и $\overline{\alpha}$ за едномерната в направление z задача* (4.85) — (4.89)

4.6.3 Резултати от методическите изследвания

За всяка от едномерните задачи (4.73) — (4.77) и (4.85) — (4.89) са извършени числени симулации върху издребняващи редици от мрежи по пространството. Резултатите от тези симулации са систематизирани в таблица 4.8 и таблица 4.9. По метода на Рунге е установено, че независимо от направлението, сходимостта по пространството е квадратична.

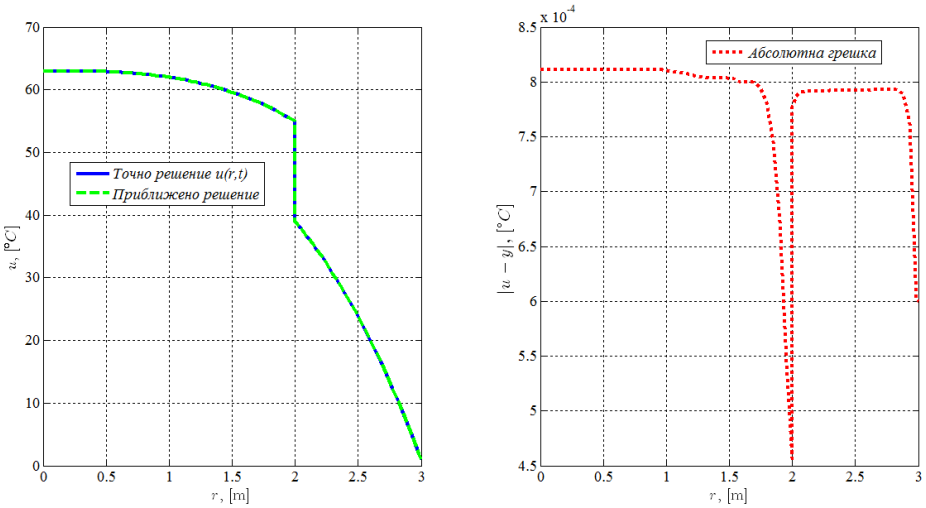
Брой точки по r	$Q = 4$, $N = 7$	$Q = 8$, $N = 13$	$Q = 16$, $N = 25$	$Q = 32$, $N = 49$	$Q = 64$, $N = 97$	$Q = 128$, $N = 193$
C-норма на грешката	9.2e-03	4.9e-03	1.8e-03	5.0e-04	1.4e-04	3.8e-05
Отн. на две посл. гр.	–	1.89	2.72	3.50	3.59	3.76
Ред на сходимост γ_r	–	0.91	1.44	1.81	1.84	1.91

Таблица 4.8: *Числени резултати за сходимост на едномерната задача* (4.73) — (4.77) *в направление* r

Брой точки по z	$P = 3$, $M = 8$	$P = 6$, $M = 17$	$P = 12$, $M = 35$	$P = 24$, $M = 71$	$P = 48$, $M = 143$	$P = 96$, $M = 287$
C-норма на грешката	1.95e-03	7.1e-04	2.39e-04	6.3e-05	1.56e-05	4.09e-06
Отн. на две посл. гр.	–	2.74	2.99	3.80	4.02	3.80
Ред на сходимост γ_z	–	1.45	1.58	1.93	2.01	1.93

Таблица 4.9: *Числени резултати за сходимост на едномерната задача* (4.85) — (4.89) *в направление* z

На фиг. 4.13 графично сме представили сравнението между аналитичното решение (4.84) на задача (4.73) — (4.77) и съответното приближено решение, пресметнато от чисто неявната диференчна схема (4.55), за фиксиран момент от време $t = 0.7805$ s. Същите резултати са дадени на фиг. 4.14 за аналитичното решение (4.90) на задача (4.85) — (4.89) и решението на диференчната схема (4.72). Както се вижда, и в двата случая абсолютната грешка не надминава 10^{-4}.



Фигура 4.13: *Сравнение между точното решение $u(r,t)$, даващо се с формулата (4.84), и приближеното решение, получено с метода на крайните разлики*

4.7 Валидиране на математическия модел на база експериментални данни

Приближеното решение на математическия модел (4.2) — (4.9) е пресметнато върху следната двумерна неравномерна пространствена мрежа

$$\widehat{\omega}^*_{h^r} \times \widehat{\omega}_{h^z},$$

чиято структура е представена детайлно в таблици 4.10 и 4.11. Времевата мрежа $\overline{\omega}_\tau$ въвеждаме в интервала $0 \leq t \leq 743.6$, като избираме равномерна стъпка $\tau = 0.2$.

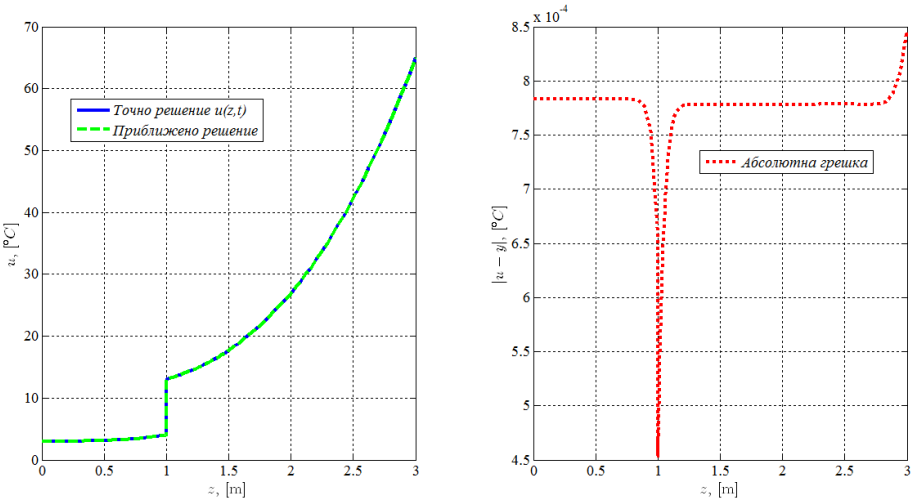
h^r_1	h^r_2	$h^r_{3:Q}$	h^r_{Q+1}	$h^r_{Q+2:N-1}$	h^r_N
0	0.0007	0.0001	0	0.0007	0.0015

Таблица 4.10: *Неравномерна мрежа $\widehat{\omega}^*_{h^r_i}$*

h^z_1	h^z_2	$h^z_{3:P}$	h^z_{P+1}	$h^z_{P+2:M-1}$	h^z_M
0	0.002	0.0028	0	0.003	0.003

Таблица 4.11: *Неравномерна мрежа $\widehat{\omega}_{h^z_j}$*

Стойностите на входните данни, които използваме за числената си-



Фигура 4.14: *Сравнение между точното решение $u(z,t)$, даващо се с формулата (4.90), и приближеното решение, получено с метода на крайните разлики*

мулация, са дадени подробно по-долу.

■ *Топлофизични характеристики*

$$c_1 = 1080 \text{ J/(kg.}^\circ\text{C)}, \quad \rho_1 = 2500 \text{ kg/m}^3, \quad \lambda_1 = 104 \text{ W/(m.}^\circ\text{C)[23]},$$

$$c_2 = 1080 \text{ J/(kg.}^\circ\text{C)}, \quad \rho_2 = 1650 \text{ kg/m}^3, \quad \lambda_2 = 1.28 \text{ W/(m.}^\circ\text{C)[23]},$$

$$c_3 = 753 \text{ J/(kg.}^\circ\text{C)}, \quad \rho_3 = 7500 \text{ kg/m}^3, \quad \lambda_3 = 54.5 \text{ W/(m.}^\circ\text{C)[23]}$$

■ *Начални температури на сплавта и на формата*

$$u_{0,\text{м.}} = 680^\circ\text{C},$$

$$u_{0,\text{ф.}} = 20^\circ\text{C}$$

■ *Температури на ликвидуса и на солидуса*

$$u_L = 617^\circ\text{C},$$

$$u_S = 575^\circ\text{C}$$

■ *Скрита топлина на кристализация, енталпия на фазов преход*

$$\kappa = 3.69 \cdot 10^5 \text{ J/kg}, \quad \eta = 2.7306 \cdot 10^5 \text{ J/kg}$$

■ *Коефициенти на контактен топлообмен*

$$\alpha_k = 10^4 \text{ W/(m}^2.^\circ\text{C)}, \quad \widetilde{\alpha} = 10 \text{ W/(m}^2.^\circ\text{C)},$$

$$\overline{\alpha} = 40 \text{ W/(m}^2.^\circ\text{C)}, \quad \alpha = 41 \text{ W/(m}^2.^\circ\text{C)}$$

■ *Геометрични размери на формата*

$$h = 0.015 \text{ m}, \quad H = 0.099 \text{ m},$$

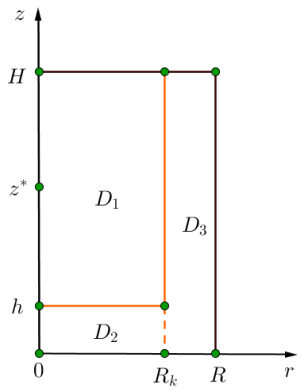
$$R_k = 0.02865 \text{ m}, \quad R = 0.0315 \text{ m}$$

■ *Дебелина на твърдата фаза, ширина на междината, коефициент на топлопроводност на въздуха*

$$\delta = 0.006 \text{ m}, \quad \delta_M = 0.00015 \text{ m}, \quad \lambda_M = 0.0257 \text{ W/(m.}^\circ\text{C)}$$

Ще обърнем внимание на следния факт. Представените стойности на топлофизичните характеристики на сплавта **AlSi7Mg** съвпадат с използваните вече такива при едномерния математически модел (вж. глава 3, таблица 3.1).

Симулационната стойност на началната температура $u_{0,\text{м.}}$ на сплавта е взаимствана от технологичен експеримент, представен в [82]. Стойността на u_L е получена приближено чрез използване на диаграмата на състоянията за сплав **AlSi7** [82] (вж. отново секция 4.2). Енталпията η на фазовия преход, както вече беше казано, е пресметната по формулата $\eta = \kappa(1 - \psi_S)$, където κ е топлината на кристализация, а ψ_S е съдържанието на твърдата фаза в двуфазната зона. За разглежданата сплав със 7 wt%-но съдържание на силиций **Si** е установено, че в интервала $(u_S; u_L)$ се отделят $\psi_S = 26\%$ твърда фаза [23].



Фигура 4.15: *Схематично представяне на формата за отливане на отливката. Термодвойката е разположена в точката с координати* $(r^*; z^*) = (0; 0.0565)$ m

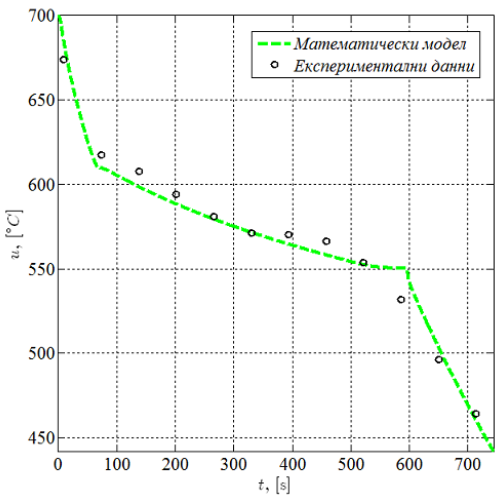
Валидирането на модела (4.2) — (4.9) е осъществено на базата на експериментални данни за сплавта **AlSi7Mg** [82]. Тези данни са получени по време на реален физически експеримент чрез използване на термодвойка, монтирана върху оста на формата на височина $z^* = 0.0565$ m (вж. фиг. 4.15).

Резултатите от сравнението между експеримента и математическия модел са изобразени графично на фиг. 4.16. Симулацията показва добро съвпадение. Качеството му може да се установи не само визуално, но и чисто теоретично чрез стойностите на L_2-нормата и на относителната грешка ε. В конкретния случай $\|\cdot\|_{L_2} = 22.1765$, а $\varepsilon < 4\%$. На практика се оказва, че за технолозите и изследователите относителна грешка от порядъка на 10% е напълно удовлетворителна.

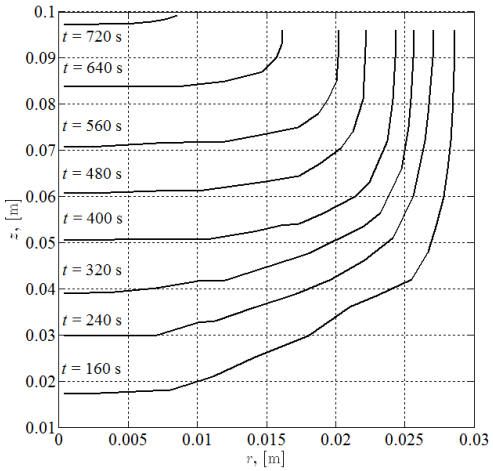
В таблици 4.12 и 4.13 сме систематизирали данни за разпределението на температурата в обема на отливката при $t = 100$ s и $t = 400$ s.

Табличните стойности, които са изписани с получер курсив, отговарят на температурата на сплавта в конкретния момент от време. Останалите стойности, които са дадени с нормален шрифт, се отнасят за температурата на дъното на отливката и за формата.

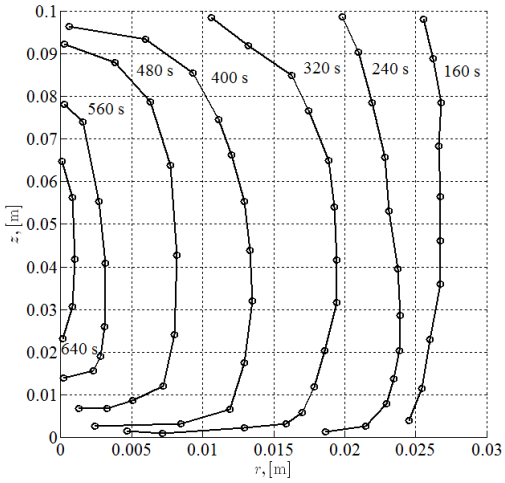
На фиг. 4.17 сме показали движението на изотермите $u = u_S$ в случая, когато нямаме топлинни загуби през горната основа на отливката ($\overline{\alpha} = 0$ W/(m^2.° C)), за някои конкретни моменти от време. Можем да забележим, че последният участък, в който кристализацията е завършила напълно, е разположен близо до тази основа. Това означава, че дефектите в микроструктурата на получения слитък биха били минимални.



Фигура 4.16: *Валидиране на теоретичния модел* (4.2) — (4.9) *чрез сравнение с експериментални резултати, получени в лабораторни условия*



Фигура 4.17: *Изотермични линии на солидуса в обема на отливката при отсъствие на топлинни загуби* ($\overline{\alpha} = 0$)



Фигура 4.18: *Изотермични линии на солидуса в обема на отливката при наличие на топлинни загуби* ($\overline{\alpha} \neq 0$)

Движението на изотермите в случая, когато имаме топлинни загуби през горната основа на отливката ($\overline{\alpha} = 40$ W/(m^2.$^\circ$C)), е показано графично на фиг. 4.18. От чертежа се вижда, че в областта от $z = 0.02$ m до $z = 0.07$ m се формира последната твърда фаза, която е разположена в близост до оста на формата.

0.0990	***608.22***	***608.12***	***607.36***	***606.69***	***605.83***	508.49
0.0870	***610.34***	***610.24***	***609.48***	***608.81***	***607.94***	512.36
0.0750	***611.35***	***611.25***	***610.48***	***609.81***	***608.95***	514.95
0.0510	***610.24***	***610.14***	***609.37***	***608.70***	***607.83***	516.40
0.0270	***604.39***	***604.29***	***603.44***	***602.70***	***601.77***	512.98
0.0150	***599.75***	***599.63***	***598.66***	***597.77***	***596.53***	509.50
0.0103	440.02	439.83	437.13	432.20	419.51	506.13
0.0042	297.34	297.12	293.81	288.01	275.57	504.51
0.0000	259.55	259.33	256.11	250.64	239.49	503.87
$t = 100$ s	**0.0004**	**0.0048**	**0.0143**	**0.0190**	**0.0237**	**0.0299**

Таблица 4.12: *Температурен профил на цилиндрична отливка от сплав* ***AlSi7Mg***, $t = 100$ s

4.8 Параметрично изследване

Както всеки друг теоретичен модел, така и (4.2) — (4.9) съдържа допускания, които идеализират възникването, протичането и изследването

0.0990	***561.47***	***561.38***	***560.68***	***560.06***	***559.27***	482.77
0.0870	***563.54***	***563.45***	***562.74***	***562.13***	***561.33***	486.54
0.0750	***564.79***	***564.70***	***563.99***	***563.38***	***562.58***	489.32
0.0510	***565.11***	***565.02***	***564.30***	***563.68***	***562.88***	492.01
0.0270	***563.05***	***562.94***	***562.09***	***561.36***	***560.47***	490.95
0.0150	***561.77***	***561.62***	***560.49***	***559.47***	***558.14***	489.04
0.0103	527.28	526.11	515.28	502.91	480.77	487.08
0.0042	490.12	488.12	470.04	450.57	420.16	486.06
0.0000	471.42	469.29	450.16	429.90	399.24	485.57
$t = 400$ s	**0.0004**	**0.0048**	**0.0143**	**0.0190**	**0.0237**	**0.0299**

Таблица 4.13: *Температурен профил на цилиндрична отливка от сплав* ***AlSi7Mg****,* $t = 400$ s

на кристализацията. Необходимо е извършването на съответно параметрично изследване върху някои от параметрите на този модел.

Нека $\mathscr{M}_B$ е един базов модел, спрямо който ще извършваме параметричното изследване. Естествено е този модел да съвпада с вече валидирания (4.2) — (4.9) при посочените по-горе стойности на входните данни. Да означим с p_v онзи от параметрите, по отношение на който извършваме параметричното изследване, а с $\mathscr{M}_v$ — съответния му математически модел. Ако $p_v^{(i)}$ е една конкретна стойност на избрания параметър p_v, то съответния и́ математически модел да означим с $\mathscr{M}_v^{(i)}$. Благоразумни критерии за това, колко силно влияе изменението на параметъра p_v върху цялостния реален процес, могат да бъдат например:

1. C-нормата на разликата в температурите за текущия модел $\mathscr{M}_v^{(i)}$ и базовия модел $\mathscr{M}_B$:
$$\left\| \mathscr{M}_B - \mathscr{M}_v^{(i)} \right\|_C;$$

2. относителната грешка на разликата в температурите за текущия модел $\mathscr{M}_v^{(i)}$ и базовия модел $\mathscr{M}_B$:
$$\varepsilon = \frac{\left\| \mathscr{M}_B - \mathscr{M}_v^{(i)} \right\|_C}{\left\| \mathscr{M}_B \right\|_C} \cdot 100;$$

3. L_2-нормата на разликата в температурите за текущия модел $\mathscr{M}_v^{(i)}$ и базовия модел $\mathscr{M}_B$:
$$\left\| \mathscr{M}_B - \mathscr{M}_v^{(i)} \right\|_{L_2}.$$

Нека да означим с χ_v степента на влияние на параметъра p_v и да разгледаме случаите, когато конкретната ѝ стойност е породена от C-нормата, от относителната грешка ε или от L_2-нормата: $\chi_v \sim \|\cdot\|_C$, $\chi_v \sim \varepsilon$, $\chi_v \sim \|\cdot\|_{L_2}$. Очевидно във всеки от трите случая $\chi_v \in [0; +\infty)$. При това, ако за оценка на χ_v използваме относителна грешка, от физични съображения можем да считаме, че $0\% \leq \chi_v \leq 10\%$, или, което е същото, $0 \leq \chi_v \leq 0.1$.

Да въведем следните дефиниции, които ще са ни необходими при анализиране степента на влияние χ_v на параметъра p_v.

Дефиниция 1 *Моделът $\mathscr{M}_v$ се нарича* ***устойчив*** *по отношение на параметъра p_v, ако на малки изменения в стойностите на p_v съответстват малки изменения в стойностите на решението на модела.*

Дефиниция 2 *Моделът $\mathscr{M}_v$ се нарича* ***неустойчив*** *по отношение на параметъра p_v, ако на малки изменения в стойностите на p_v съответстват големи изменения в стойностите на решението на модела.*

Таблица 4.14: *Степен на влияние на параметъра α*

α	$\|\cdot\|_C$	ε	$\|\cdot\|_{L_2}$
36	48.9653	6.9950	1184.6920
38	30.7667	4.3952	738.5282
40	13.1572	1.8796	254.9282
41	**0.0000**	**0.0000**	**0.0000**
42	13.3034	1.9005	258.2196
50	76.9087	10.9869	2271.0110
Средно:	30.5169	4.3595	784.5632

В качеството на обекти на параметричното изследване да изберем коефициентите на контактен топлообмен α, $\widetilde{\alpha}$, $\overline{\alpha}$, α_k и скритата топлина на кристализация κ. В таблици 4.14 — 4.18 нагледно са представени резултатите от експерименталните оценки за степента на влияние χ_v на споменатите параметри. В първия стълб на всяка таблица са дадени стойностите на χ_v, когато тя е породена от C-нормата, във втория стълб — когато е породена от относителната грешка ε, и в третия стълб — когато е породена от L_2-нормата. Сравнението с базовия модел $\mathscr{M}_B$ е реализирано за всяко време в точката на термодвойката. Нулевите редове в разчетните таблици 4.14 — 4.18 са референция към базовия модел

Таблица 4.15: *Степен на влияние на параметъра* $\widetilde{\alpha}$

$\widetilde{\alpha}$	$\|\cdot\|_C$	ε	$\|\cdot\|_{L_2}$
4	6.1202	0.8744	123.3571
6	4.7310	0.6759	81.6417
8	2.9917	0.4274	41.0726
10	**0.0000**	**0.0000**	**0.0000**
12	3.4932	0.4990	40.0707
15	6.7364	0.9623	98.5222
Средно:	4.0121	0.5732	64.1107

Таблица 4.16: *Степен на влияние на параметъра* $\overline{\alpha}$

$\overline{\alpha}$	$\|\cdot\|_C$	ε	$\|\cdot\|_{L_2}$
36	12.8066	1.8295	194.8519
38	7.8788	1.1255	98.7587
40	**0.0000**	**0.0000**	**0.0000**
42	6.9877	0.9982	98.4789
50	21.3468	3.0495	480.5495
56	30.7362	4.3909	760.9253
Средно:	13.2927	1.8989	272.2607

и са включени само като ориентир. Последните редове представляват усреднени оценки за степента на влияние χ_v по стълбове.

Като съобразим смисъла на приведените таблични данни и конкретните им стойности, достигаме до следните изводи.

1. Без значение по кой от трите начина оценяваме степента на влияние χ_v на параметрите на модела (4.2) — (4.9), нейните усреднени стойности удовлетворяват неравенствата

$$\chi_{\alpha_k} < \chi_{\widetilde{\alpha}} < \chi_{\kappa} < \chi_{\overline{\alpha}} < \chi_{\alpha}.$$

2. Моделът (4.2) — (4.9) е устойчив по отношение на параметрите α_k, $\widetilde{\alpha}$, κ и е неустойчив по отношение на $\overline{\alpha}$ и α.

3. Параметърът, който оказва най-малко влияние върху поведението на температурната крива, е коефициентът на топлообмен α_k между легираното пясъчно дъно и сплавта. Обратно — най-голямо влияние оказва коефициентът на топлообмен α с околната повърхнина на формата. Така например увеличението на α с 2 $\mathrm{W/(m^2.^\circ C)}$

Таблица 4.17: *Степен на влияние на параметъра* α_k

α_k	$\|\cdot\|_C$	ε	$\|\cdot\|_{L_2}$
$10^4 - 100$	0.0059	0.0008	0.1122
$10^4 - 50$	0.0029	0.0004	0.0539
$\mathbf{10^4}$	**0.0000**	**0.0000**	**0.0000**
$10^4 + 200$	0.0180	0.0026	0.4892
$10^4 + 600$	0.0333	0.0048	0.7034
$10^4 + 1000$	0.0535	0.0076	1.0768
Средно:	0.0189	0.0033	0.4059

Таблица 4.18: *Степен на влияние на параметъра* κ

κ	$\|\cdot\|_C$	ε	$\|\cdot\|_{L_2}$
$3.62 \cdot 10^5$	11.2481	1.6069	180.0466
$\mathbf{3.69 \cdot 10^5}$	**0.0000**	**0.0000**	**0.0000**
$3.71 \cdot 10^5$	4.9106	0.7015	52.4233
$3.72 \cdot 10^5$	6.4831	0.9262	76.9577
$3.73 \cdot 10^5$	7.7915	1.1131	103.4508
Средно:	6.0867	0.8695	82.5757

води до разминаване спрямо модела $\mathscr{M}_B$ с 2%, а увеличението с 10 W/(m^2.$^\circ$ C) — до разминаване от около 11% (вж. таблица 4.14).

На база генерираните графични и числени резултати можем с голяма достоверност да предположим, че моделът (4.2) — (4.9) адекватно описва процеса на затвърдяване при бинарните сплави. Предложеният диференчен метод и компютърната програма позволяват с достатъчна точност да бъде получена предварителна информация за основните характеристики на кристализационния процес. По-нататъшните изследвания в тази област вероятно ще вземат предвид особеностите на онези сплави, легирани с повече от един легиращ елемент, както и разнообразието в тримерната структура на отливката и формата. Без особени трудности предложената двуслойна, неявна и абсолютно устойчива локално едномерна схема може да се трансформира в трислойната явна и абсолютно устойчива схема на Дюфорт и Франкел.

Глава 5

Създаване на графичен потребителски интерфейс чрез средствата на софтуера MATLAB

5.1 Графичен потребителски интерфейс

В последните години съвременното информационно общество непрекъснато се адаптира към условията на околната среда. То се стреми непрекъснато да създава нови технологични и програмни продукти, които да прилага в ежедневието си. Тези продукти са резултат от достиженията в природните науки и компютърната техника. Най-общо можем да ги разделим на две големи групи:

1. *софтуер с общо предназначение*;

2. *софтуер от специализиран тип.*

Софтуерът с общо предназначение е насочен към масовия потребител, инструкциите за употреба са непосредствени и не изискват предварителна подготовка.

Софтуерът от специализиран тип се прилага в по-тесни предметни области като химия, медицина, физика, математика и др. Той е създаден за нуждите на тези науки и изисква поначало добро познаване на фундаменталните им принципи. Работата с програмни пакети от този вид налага предварително запознаване с тяхната документация и с начина им на употреба.

Системите за компютърна алгебра са типичен пример за специализиран софтуер. Възникват около 70-те години на XX век. В началото основното им предназначение са символните преобразувания (включително символно диференциране и символно интегриране). След появата и развитието на изчислителната математика през 70-те години към основните им концепции се прибавят приближаването на функции с алгебрични полиноми, численото решаване на нелинейни уравнения, приближеното решаване на обикновени и частни диференциални уравнения, както и съответната визуализация на резултатите. От 1980 г. досега те са претърпели значителни промени и подобрения в настройките си. Документацията е снабдена с продуктов **Help**, съдържащ синтаксиса на всички вградени функции и алгоритми и освен това голям набор от готови примери.

Към днешна дата най-известните системи за компютърна алгебра са **MATLAB** [32], [54], [56], [67] (създадена от **MathWorks Inc.** през 1984 г. в гр. Нейтик, щат Масачузетс, САЩ), **Maple** [63] (разработена от компанията **Maplesoft Inc.** през 1982 г. в гр. Ватерло, окръг Онтарио, Канада) и **Wolfram Mathematica** [31] (продукт на **Wolfram Research**, реализирана за първи път на 23 юни 1988 г. в гр. Шампейн, щат Илинойс, САЩ).

Системата за компютърна математика **MATLAB** е предпочитана софтуерна платформа в повечето научни среди. С ясен и лаконичен синтаксис, с богати възможности за графично изобразяване и с нестрога типизация на входните данни, тя съчетава в едно цяло много от добрите програмистки практики. Нейният език за програмиране е близък по структура до познатите **Fortran**, **C/C++**, **Java**, **Python**, **JavaScript** и е *интерпретативен*. Това означава, че програмният код се изпълнява директно, без предварително да се превежда като списък с инструкции на машинен език и да се компилира. Високите стойностни качества на **MATLAB** се допълват и от вградения инструментариум на програмиста. Нека да направим кратък обзор на някои от елементите на този инструментариум.

□ *Пакет cftool (**C**urve **F**itting **Tool**box)*

Той е незаменим в случаите, когато е необходима бърза апроксимация по експериментални данни. Стартира се от командния ред на **MATLAB** чрез въвеждане на командата

```
>> cftool
```

След изпълнението на един вариант автоматично се подава информация към потребителя за качеството на приближението, в това число мярка за средноквадратична грешка и доверителни интервали, съдържащи коефициентите на приближаващата функция. Ще отбележим, че за разлика от вградените процедури за числени апроксимации, *cftool* предлага много по-богати възможности и избор на тази приближаваща функция във всяка конкретна ситуация. Като наръчник за практическото използване на *cftool* препоръчваме потребителското ръководство [73] или продуктовия **Help**.

□ *Пакет pdetool (**P**artial **D**ifferential **E**quations **Tool**box)*

Диференциалните уравнения, срещащи се в науката и практиката, в болшинството от случаите не притежават аналитични решения. Затова е важно да можем да намираме приближени аналози в дефиниционната област на тези уравнения. Основните начини за построяване на апроксимация са метод на крайните разлики и метод на крайните елементи.

Методът на крайните разлики (МКР) изисква дискретизиране на областта, в която ще търсим приближено решение, чрез изброим брой правоъгълници, такива, че страните им са успоредни на координатните оси. След дискретизацията, като се използват основните апроксимации на производните чрез крайни разлики, се съставя диференчна задача и ако тя е сходяща, нейното решение е търсеното приближено решение. Този метод е подходящ при пространствени области с несложен контур и многостенна форма.

Методът на крайните елементи (МКЕ) е по-универсален, защото е приложм за области с произволна форма. При него дискретизацията се извършва чрез т. нар. *триангулация*: мрежата, върху която ще се строи апроксимацията, е съставена от триъгълници, при това всеки два от тях или нямат обща точка, или имат общ връх, или имат обща страна. Пакетът *pdetool* решава елиптични, хиперболични и параболични частни диференциални уравнения именно чрез МКЕ. За подробно запознаване с *pdetool* и разбиране на принципите му неоценима помощ ще окажат книгите [75] и [76], както и съответната софтуерна документация в [85].

□ *Пакет Simulink*

Основната концепция на този пакет лежи върху изграждането и проектирането на човеко-машинен интерфейс, както и моделиране на реални процеси в реално време. Тук се включват понятия като блокова диаграма на модела, автоматично генериране на код, верификация на вградени

системи, солвъри за моделиране и симулация на динамични системи и др. Всички тези първични понятия очертават рамките, в които е заключена мултифункционалността на *Simulink*. Освен това е възможен експорт на резултатите, генерирани от *Simulink*, в работното пространство на **MATLAB** с цел по-нататъшно изследване. Най-пълна и подробна информация за структурата, механизма на действие и употребата на *Simulink* може да бъде намерена директно в продуктовата документация, представена в [84].

□ *Пакет guide (**G**raphical **U**ser **I**nterface)*

Пакетът *guide* е кулминацията във функционалността, която платформата **MATLAB** предоставя. По своята същност той е интерактивна връзка между софтуера, натоварен да изпълнява определени команди, и потребителя, който осигурява входната информация [72], [74]. Самото графично приложение, създадено от програмиста като мост между софтуера и потребителя, е прието да се нарича **графичен потребителски интерфейс (GUI)**. Един **GUI** може ефективно да бъде конструиран в средата *guide* [32], [72]. За целта се изисква добро познаване на основните графични елементи, както и какви взаимодействия могат да осъществяват те помежду си. Фундаменталните принципи на програмирането въобще остават в сила.

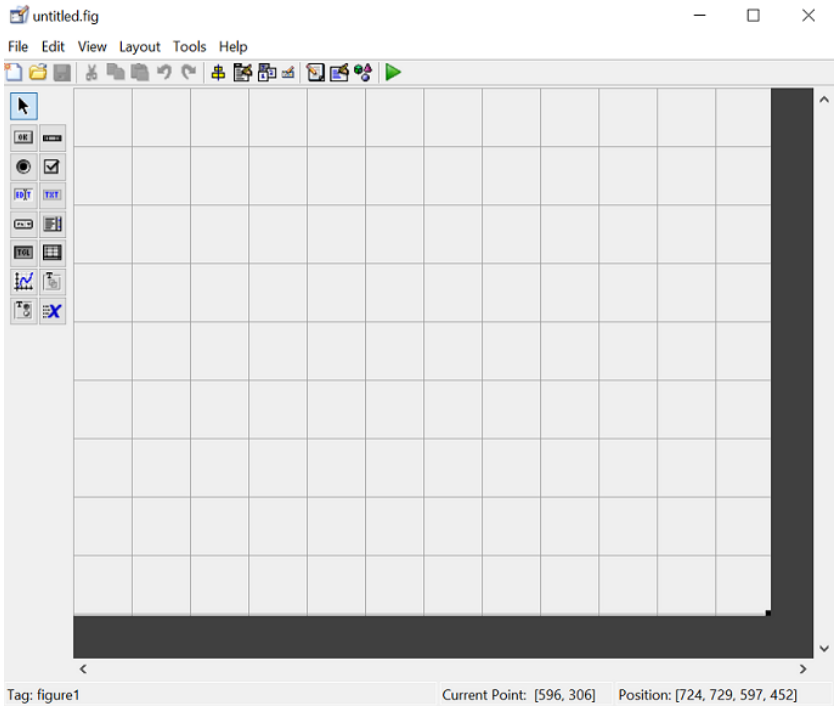
5.2 Класът *handles*

Успешното проектиране и реализиране на **GUI** е свързано неминуемо и с обектно ориентираното програмиране (ООП). Както е добре известно от концепциите на ООП, градивната единица, скелетът на програмата е *класът*. Класът, погледнат абстрактно, представлява своеобразна черна кутия, задаваща една и съща група от свойства на обект, който реално съществува. От всеки клас могат да се създават конкретни негови инстанции — *обекти*. Самият процес се нарича инстанциране. Обектите, от своя страна, приемат очертанията, които класът им е задал.

Свойствата на обектите са винаги достъпни в рамките на класа и евентуално достъпни извън него (в други класове или именувани пространства). Управлението и изменението им се осъществява от функциите на класа: конструктори, деструктори, операторни функции за присвояване, предефинирани операторни функции, функции за извличане на стойността на свойството (*get-методи*), функции за модификация на стойността на свойството (*set-методи*).

По същия начин дефинираме и чертите на един графичен интерфейс. Всеки **GUI** се характеризира със свой базов, главен клас, наречен **handles**. В него се съхраняват указателите на всички графични обекти, които са създадени и използвани до момента [32]. Ако работим например с обект **Line**, който представлява линия в равнината, той е от класа **handles** и може да бъде достъпен с познатата нотация **handles.Line**.

5.3 Основни графични обекти в GUI



Фигура 5.1: *Стандартен изглед на графичния прозорец в средата* ***guide***

Нека да погледнем по-отблизо графичния прозорец и управляващите графични обекти на средата *guide*. След стартирането ѝ с командата

```
>> guide
```

от командния ред на **MATLAB** се появява прозорецът, изобразен на фиг. 5.1. Панелът с графичните обекти е подреден отляво. Съответните обекти са разделени на два типа:

- *статични* — не променят стойностите на свойствата си до края на потребителската сесия;

- *динамични* — променят стойностите на някои от свойствата си или на всичките си свойства до края на съответната сесия.

За еднозначната идентификация на всеки елемент от **GUI** служи свойството **Tag**, което е видимо в секцията **Property Inspector**. Така например, ако сме създали графичен обект **Axes** и сме именували неговото свойство **Tag** с **mainAx**, то достъпът до този обект в програмния код се извършва така: **handles.mainAx**.

Графичните обекти, разположени върху повърхността на интерфейса (динамични текстови полета, падащи менюта, плъзгачи, полета за отметки, радиобутони и т. н.), са свързани с определено събитие. Натискането на един стандартен бутон от тип **Push Button** може както да започне изпълнението на даден алгоритъм, така и да го прекъсне. Зад организацията и реализирането на събитията стоят т. нар. *callback-функции*, които не се отличават съществено от обичайните функции. Разликата се изразява в това, че с всеки графичен обект е асоциирана единствена *callback*-функция и изпълнението на съответното събитие всъщност представлява изпълнението на тази функция.

Когато за първи път съхраним или стартираме приложението, **guide** запазва два файла [72]:

- **FIG-file** с разширение *.fig*, който съдържа пълното описание на нашия **GUI** и на неговите компоненти; това е бинарен файл, който не допуска модифициране освен в случаите, когато променяме облика на приложението в средата *guide*;

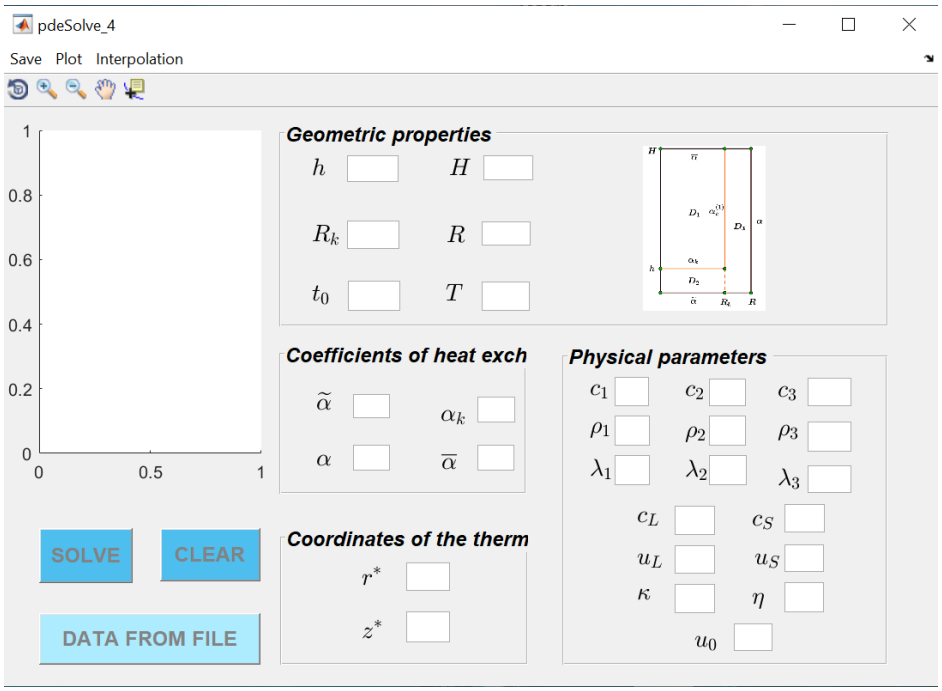
- файл с разширение *.m*, който по подразбиране съдържа началния инициализиращ код и шаблоните на някои от *callback*-функциите; в процеса на тестване и усъвършенстване на **GUI** тези функции постепенно се запълват с програмен код.

5.4 Създаване на графичен потребителски интерфейс за решаване на двумерна кристализационна задача с фазов преход в цилиндрични координати

Реалните експерименти в научните лаборатории безспорно са носители на ценна информация. От една страна, те служат за потвърждаване или отхвърляне на дадена хипотеза, но от друга, са скъпоструващи. Необходимо е научните звена и техните подразделения да разполагат с

алтернативен вариант, който да прилагат при конструирането и доказването на тези теории.

Методът на изчислителния експеримент или още методът на числените симулации използва достиженията в научно-техническите среди. Той е свързващото звено между теоретичния математически модел и компютърната му реализация. От 70-те години на XX в. досега е бил използван в множество реални ситуации. Предпочитан инструмент е в дейността на инженери, физици, биолози. Институтът по металознание и Институтът по математика и информатика към БАН дълги години са работили съвместно и ползотворно, замествайки някои от експерименти в лабораторни условия с изчислителни експерименти. Доказано е на базата на експериментални данни, че компютърната симулация е сигурен и надежден метод за получаване на теоретично и практически издържани резултати. Например в [28], [29] акцентът е върху числено моделиране на реален кристализационен процес за реална сплав в реални атмосферни условия.



Фигура 5.2: *Примерна схема на **GUI** за решаване на двумерна кристализационна задача с фазов преход в цилиндрична координатна система*

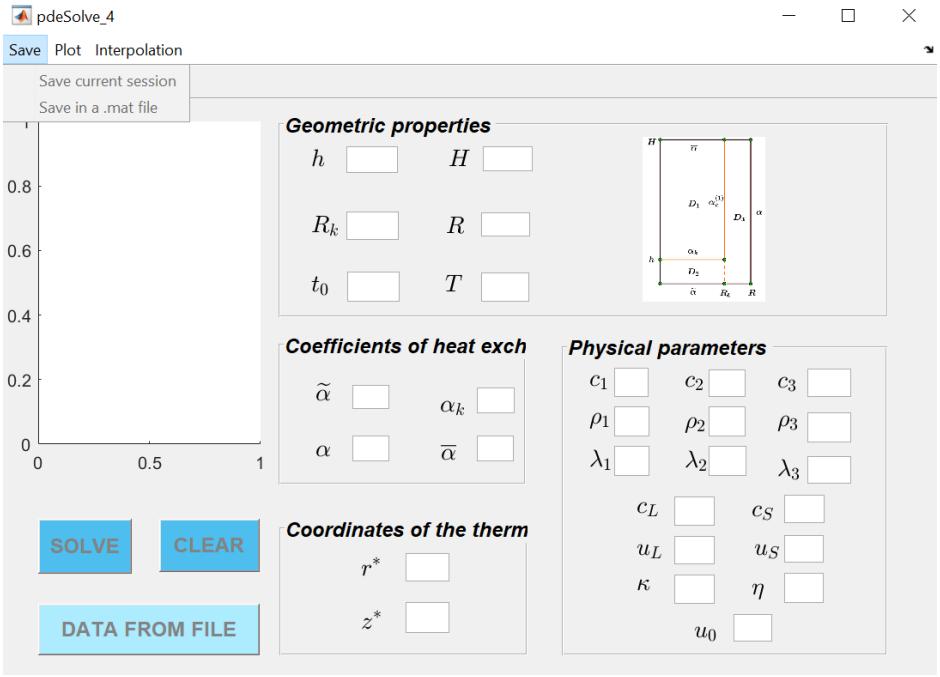
Ще покажем как, използвайки богатите възможности на системата **MATLAB** и по-конкретно на нейния пакет *guide*, можем да проектира-

ме графичен потребителски интерфейс за решаване на двумерни кристализационни задачи с фазов преход в цилиндрична координатна система. Текущата версия на **MATLAB**, с която ще работим, е **MATLAB R2014a**. Ще следваме основните принципи при конструирането на **GUI**, които са подробно изложени например в [32], [72].

Ясно е, че наборът от функции и скриптове, с помощта на който ние решихме двумерната кристализационна задача за алуминиевата сплав **AlSi7Mg**, трябва да остава в сила. Водещите идеи сега са следните:

- ☐ графичното приложение да е съотносимо към изискванията на съвременните технологични процеси на леене, т. е. да има възможност за динамична промяна в параметрите на модела;
- ☐ да е налице възможност за извличане на данни във вид на текстови или други файлове за температурните полета въз основа на числените симулации;
- ☐ по интерактивен път да се представя във вид на двумерна графика температурното разпределение за кое да е правоъгълно сечение, успоредно на оста на цилиндъра, в конкретен момент от време;
- ☐ да се покаже нагледно движението на изотермите $u = u_S$ в обема на отливката с течение на времето;
- ☐ автоматично да бъде изтривано съдържанието на всички динамични текстови и графични полета след приключване на текущата симулация;
- ☐ да може да се започне нова потребителска сесия с нови данни за нова сплав без рестартиране на приложението;
- ☐ интерфейсът да поддържа лента с менюта, някои от които могат да дублират функциите на основните бутони в приложението;
- ☐ информацията за конкретния програмен код, генерирал графичното приложение и резултатите от числените симулации, да остава капсулирана за външния потребител (основен принцип на обектно ориентираното програмиране).

На фиг. 5.2 е представена примерна скица на **GUI**, който удовлетворява изброените по-горе критерии. При реализацията му сме се стремили да го адаптираме към нуждите на практиката. Ще отбележим изрично, че този интерфейс не може да бъде конкуренция на големите софтуерни платформи, които пазарът предлага, нито е проектиран с подобна цел.



Фигура 5.3: *Меню* ***Save***

В рамките на графичния прозорец на приложението от фиг. 5.2 са разположени следните графични елементи:

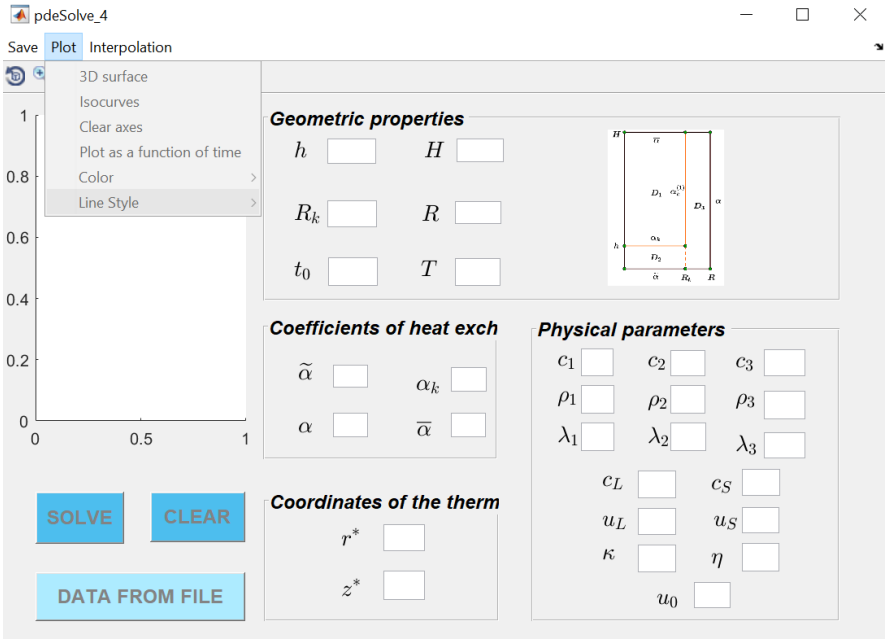
- ☐ 28 статични текстови полета, които указват съответния геометричен или физичен параметър;
- ☐ 28 динамични текстови полета за въвеждане на конкретни стойности на тези параметри;
- ☐ 4 обикновени панела, върху които по определен признак са групирани споменатите полета;
- ☐ 3 обекта от тип **Push Button** за начало на изчислителния процес, за край и за добавяне на експериментални данни към текущата сесия;
- ☐ 1 обект от тип **Axes** за графично изобразяване на получените резултати;
- ☐ лента с управляващи елементи;
- ☐ лента с менюта.

Да се концентрираме върху лентата с менюта. Тя съдържа три главни менюта: **Save**, **Plot** и **Interpolation**.

Менюто **Save**, както е видно от фиг. 5.3, предлага две опции (направени недостъпни до фактическото стартиране на изчисленията):

1. *Save current session* — запазване като **.fig**-файл на текущата потребителска сесия и на резултатите от нея;

2. *Save in a .mat file* — съхраняване в **.mat**-файл на данни за приближеното решение, в това число дискретната пространствена и времевата мрежа, индексите P и Q на двойните точки и приближеното решение u във възлите на мрежите.



Фигура 5.4: *Меню* ***Plot***

По-нататък: менюто **Plot** (вж. фиг. 5.4) е съставено от шест подменюта:

1. *3D surface* — изобразяване на повърхнината $u(r, z, t')$ в точките на дискретната пространствена мрежа за фиксиран момент от време t';

2. *Isocurves* — изчертаване на изолиниите $u = u_S$ в обема на отливката;

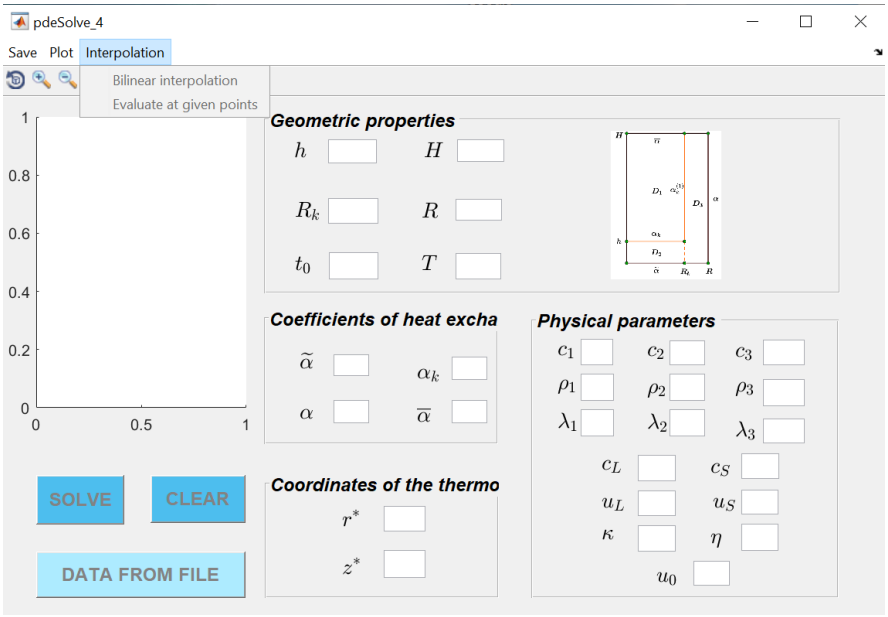
3. *Clear axes* — изчистване на координатната система;

4. *Plot as a function of time* — изчертаване на зависимостта $(t;\, u(t))$ като функция от времето във фиксирана точка $(r';\, z')$ от отливката;

5. *Color* — избор на цвят на линията $u = u(t)$ от предходната точка (син, червен, зелен или цианов);

6. *Line Style* — избор на тип на линията $u = u(t)$ (непрекъсната, прекъсната, точкова).

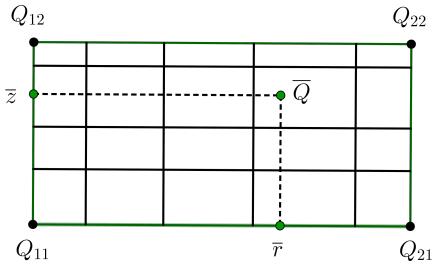
Последните две подменюта стават активни едва след като е начертана функционалната зависимост $u = u(t)$.



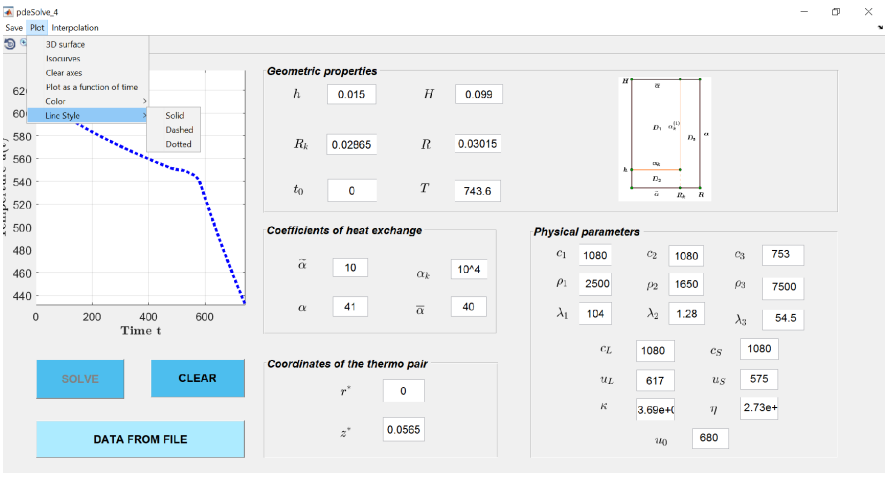
Фигура 5.5: *Меню* ***Interpolation***

Менюто **Interpolation** (вж. фиг. 5.5) служи за извличане на информация въз основа на приближеното решение u. Поддържа две функционалности:

1. *Bilinear interpolation* — чрез билинейна интерполация [40], [81] се намира стойността на температурата в допустима указана точка $\overline{Q} = (\overline{r};\, \overline{z};\, \overline{t})$, за нито една от координатите на която не е задължително да е възел от мрежата (вж. фиг. 5.6);



Фигура 5.6: *Билинейна интерполация по точките* Q_{11}, Q_{12}, Q_{21} *и* Q_{22}



Фигура 5.7: *Функционална зависимост на температурата от времето в целия обем на отливката. Подменюто* ***Plot as a function of time*** *е свързано с диалогово въвеждане на координатите на точката* $(r; z)$. *По подразбиране е заложено* $(r; z) = (0.010; 0.010)$. *Представената симулация е извършена именно за тази точка.*

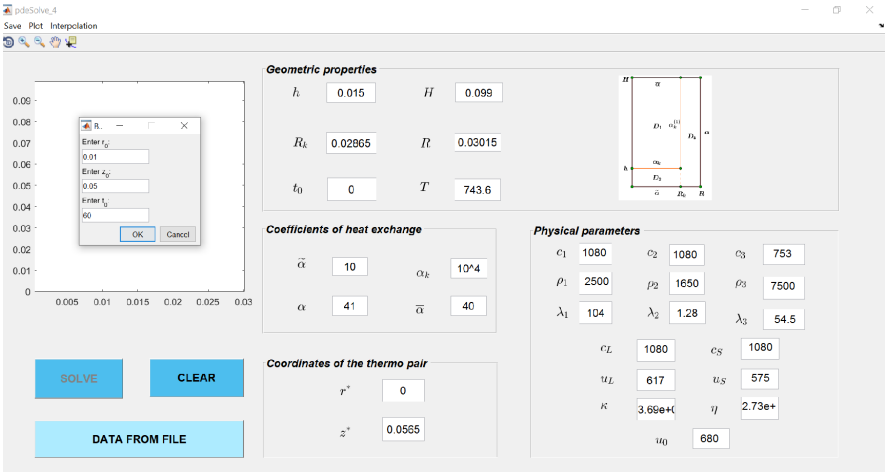
2. *Evaluate at given points* — подобно на функцията **deval** (вж. [79] или том 2 на [32]), на базата на полученото приближено решение се намират стойностите му (чрез кубична сплайнова интерполация) в набор от точки, които не са измежду възлите на мрежата; при това процедурата е едномерна, т. е. действа само в едно от двете възможни пространствени направления r или z; например ако желаем да изчислим координатите на решението u върху отсечката

$$\mathbf{r} = \left[r_{i_1};\, r_{i_2};\, \ldots;\, r_{i_{k-1}};\, r_{i_k}\right],\;\; r_1 \leq r_{i_1} < r_{i_2} < \cdots < r_{i_k} \leq r_N,$$

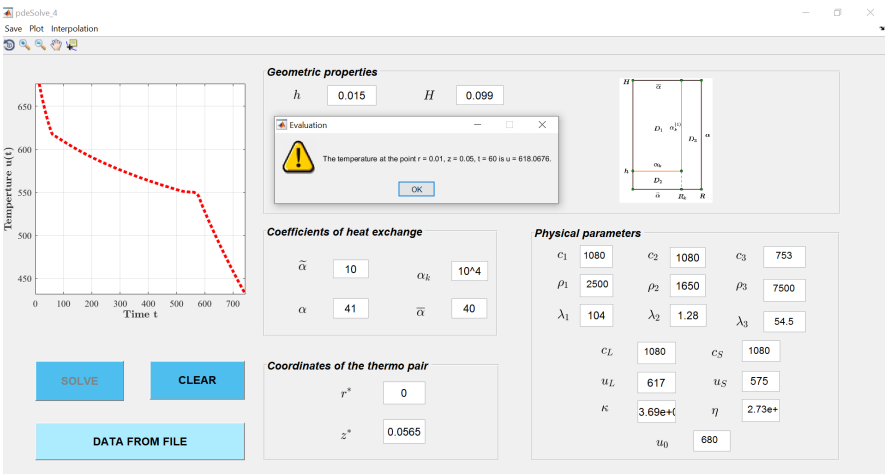
логично е да фиксираме момента от време t и височината z и след това да приложим едномерна интерполация по r.

По-долу е представен един примерен вариант на използване на при-

ложението с данните за разглежданата вече сплав **AlSi7Mg**. След пресмятането на решението в координатната система на приложението е представена зависимостта на температурата от времето (вж. фиг. 5.7). Демонстрирани са възможностите за динамична промяна в цвета и типа на линията $u = u(t)$.



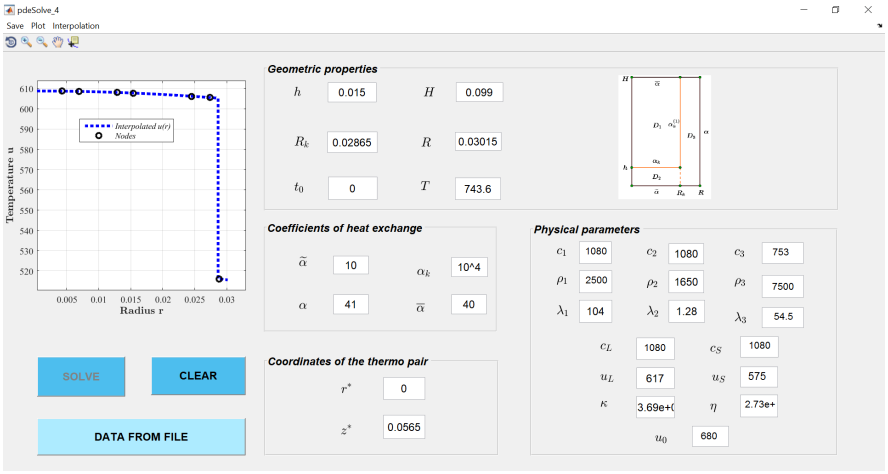
Фигура 5.8: *Диалогов прозорец, изискващ въвеждането на данни за билинейна интерполация*



Фигура 5.9: *Числен резултат от билинейна интерполация*

Командите **Interpolation** → **Bilinear interpolation** генерират диалоговия прозорец, изобразен на фиг. 5.8. По подразбиране в него е заложена точката $(\overline{r}; \overline{z}; \overline{t}) = (0.01; 0.05; 60)$, която попада в дискретната област $\overline{\Omega} = [r_1; r_N] \times [z_1; z_M] \times [t_0; t_S]$. При въвеждане на нева-

лидни стойности (т. е. поне едната координата на точката $\overline{Q}$ е извън паралелепипеда $\overline{\Omega}$ или не е от числов формат) се изпраща до повърхността на графичния интерфейс съобщение за грешка. На фиг. 5.9 е дадена желаната интерполирана стойност на температурата в точката $(\overline{r};\, \overline{z};\, \overline{t}) = (0.018;\, 0.060;\, 200)$. Резултатът може да бъде проверен лесно и чрез графиката на функцията $u = u(t)$ в координатната система отляво. Тази графика съответства на фиксирани стойности на r и z, съответно равни на $\overline{r}$ и $\overline{z}$.



Фигура 5.10: *Кубична интерполация (в радиално направление) за получаване стойностите на приближеното решение в точки, които не принадлежат на мрежата*

На фиг. 5.10 сме показали графичните резултати, генерирани от менюто **Evaluate at given points**. Симулацията е осъществена на височина $z = 0.05$ m за момента от време $t = 100$ s в точките

$$\mathbf{r} = \big[0.00425;\, 0.00693;\, 0.01287;\, 0.01539;\, 0.02453;\, 0.02737;\, 0.0288\big].$$

С помощта на инструмента **Data Cursor** (последния елемент в лентата с инструменти, вж. отново фиг. 5.10) лесно определяме желаните стойности на температурата:

$$u(\mathbf{r}) = \big[608.7;\, 608.6;\, 608.1;\, 607.8;\, 606.2;\, 605.6;\, 516.1\big].$$

В допълнение ще отбележим, че проектираният графичен потребителски интерфейс извършва аритметичните операции с двойна машинна точност. В диалогов режим освен въвеждане на входната информация се осъществява едновременно и автоматична проверка за допустимост на типовете данни.

Глава 6

Изводи от работата по дисертацията. Приноси на дисертационния труд. Апробация на резултатите

6.1 Изводи от работата по дисертацията

1. Разгледан е едномерен математически модел, който описва кристализационния процес на метална сплав. В средата на **MATLAB R2014a** е разработена компютърна програма, реализираща приближеното му решаване по явния метод на Ойлер.

2. Извършени са числени симулации за два вида сплави: алуминиева сплав **AlSi7Mg** със 7 wt%-но съдържание на легиращ елемент силиций **Si** и сив чугун, съдържащ 3,3 wt% въглерод **C**. Резултатите от симулациите показват, че увеличаването на броя на кристализационните центрове намалява преохлаждането.

3. В случая на алуминиева сплав математическият модел е валидиран на базата на известни експериментални данни. Наблюдава се добро съвпадение между резултатите от симулациите и данните от реалния експеримент.

4. Двумерният математически модел за затвърдяването на **AlSi7Mg**, отлята в експериментална форма, е решен приближено чрез използване на линеаризирана чисто неявна локално едномерна схема. Тази схема е абсолютно устойчива и сходяща към неизвестното точно

решение с ред на сходимост 2 по пространствените променливи и 1 — по времето.

5. Имплементирана е съответна компютърна програма, реализираща линеаризираната локално едномерна схема.

6. Конструирани са едномерни тестови примери по всяко от пространствените направления.

7. Чрез сравнение на приближените решения, получени по метода на баланса, с тестовите такива са доказани устойчивостта и сходимостта на избраната линеаризирана локално едномерна схема.

8. Съставени са разчетни таблици за реда на сходимост при различен брой точки в дискретните мрежи по пространството. Този ред на сходимост е пресметнат по правилото на Рунге.

9. Извършено е валидиране на двумерния математически модел чрез сравнение с известни експериментални данни за **AlSi7Mg**. Показано е, че относителната грешка между реално измерените стойности на температурата в точка от отливката и резултатите, генерирани от математическия модел за същата точка, не надминава 4%.

В заключение можем да кажем, че целите на дисертационния труд са постигнати, като са изпълнени поставените задачи.

6.2 Приноси на дисертационния труд

6.2.1 Научни приноси на дисертационния труд

1. Разработен е компютърен алгоритъм за приближено решаване на едномерния математически модел.

2. Разработен е компютърен алгоритъм за приближено решаване на двумерния математически модел.

3. Реализираните компютърни програми позволяват да бъдат определени с добра точност някои от основните величини, характеризиращи процеса на затвърдяване: температурното поле, изотермите $u = u_S$, темпът на кристализация и др.

6.2.2 Научно-приложни приноси на дисретационния труд

1. Имплементиран е собствен графичен потребителски интерфейс **PDE Solve** в средата на **MATLAB R2014a**, с помощта на който задачата за кристализация на сплави, отляти в цилиндрични форми, може да се решава в диалогов решим.

2. Разработеният софтуерен продукт може да се прилага за обучение на специалисти в областта на компютърното симулиране.

3. Потребителският интерфейс ще бъде опорна точка при съставяне на база от данни с резултатите за различни видове сплави. Информацията, съхранявана в базата, ще е от полза при провеждане на лабораторни практикуми, реални експерименти, изготвяне на статии и научни трудове и др.

6.3 Апробация на резултатите

Резултати по темата на дисертационния труд са докладвани на следните научни форуми:

1. *Preparatory Modelling Week. Bulgarian Academy of Science. Sofia, 7 — 11 September 2015*;

2. *Пета национална конференция с международно участие, металознание, хидро- и аеродинамика и национална сигурност. Българска академия на науките. София, 22. 10. 2015*;

3. *XXV всероссийская конференция с международным участием, посвященная 60-летию Института теоретической и прикладной механики им. С. А. Христиановича СО — РАН „Высокоэнергетические процессы в механике сплошной среды“. Новосибирск, Русия, 5 — 9 юни 2017 г.*;

4. *VI International Congress on Energy Fluxes and Radiation Effects (EFRE 2018), Russia, Tomsk, 16 — 22 September 2018.*

По темата на дисертационния труд са публикувани следните научни статии:

1. **E. Ilieva, L. Yovkov, M. Dobreva, P. Iliev, T. Ivanova.** *Mathematical model of crystallization of two-component alloys at presence of nanoparticles.*

Preparatory Modelling Week, booklet, Bulgarian Academy of Science. Sofia, 7 — 11 September 2015; `http://pmw2015.fmi.uni-sofia.bg/Documents/Problem_3_Report.pdf`

2. **Людмил Йовков.** *Числено симулиране на процеса на кристализация на алуминиева сплав.* Пета национална конференция с международно участие, металознание, хидро- и аеродинамика и национална сигурност. Сборник доклади, стр. 117 — 121. София, октомври 2015 г.

3. **P. M. Kuzmanov, S. I. Popov, L. V. Yovkov, R. N. Dimitrova, A. N. Cherepanov, V. K. Manolov.** *Investigation the Effect of Modification with Nano-Powders on Crystallization Process and Microstructure of Some Alloys.* AIP Conference Proceedings 1893, 030104 (2017); `https://doi.org/10.1063/1.5007562`

4. **A. Cherepanov, V. Cherepanova, V. Manolov, L. Yovkov.** *Model of heterogeneous crystallization of a melt modified by infusible nanoparticles of plasma-chemical synthesis.* IOP Publishen Conf. Series. Physics. Journal of Physics: Conference Series (2018)

5. **Черепанов А. Н., Черепанова В. К., Манолов В. К., Йовков Л. В.** *Модель гетерогенной кристаллизации расплава, модифицированного тугоплавкими наночастицами плазмохимического синтеза.* Газоразрядная плазма и ее применение, тезисы докладов XIII Международной конференции, посвященной 100-летию со дня рождения академика М. Ф. Жукова (Россия, Новосибирск, 5 — 7 сент. 2017 г.). Новосибирск: Параллель, 2017, с. 144

6. **Cherepanov A., Cherepanova V., Manolov V. and Yovkov L.** *On crystallization of a metal inoculated with nanoparticles.* Book of Abstracts of 6th International Congress on Energy Fluxes and Radiation Effects (EFRE 2018), Russia, Tomsk, 16 — 22 Sept. 2018. TPU Publishing House, 2018, p. 648. ISBN 978-5-4387-0823-0; `https://iopscience.iop.org/article/10.1088/1742-6596/1115/4/042042/pdf`

По време на следването докторантът Людмил Йовков е бил член на научния колектив, допринесъл за успешното изпълнение на проекта *Теоретично и експериментално изследване на кристализацията на метална сплав с въведени в нея наночастици.* Проектът е финансиран от фонд „Научни изследвания“ по договор № ДН 07/20/15.12.2016.

Благодарности

На финала на този научен труд бих искал да изкажа своите искрени благодарности на научните ми ръководители доц. д-р Валентин Манолов и проф. д-р Татяна Черногорова. Те бяха неотлъчно до мен през цялото време на докторантурата, готови да се борим заедно с трудностите и въпреки всичко да откриваме правилния път към науката.

Сърдечни благодарности изказвам и на проф. дмн Стефка Димова за ценните съвети, забележки и препоръки, които ми е давала през годините на краткото ни познанство.

И на трето място се обръщам сега към теб, драги читателю. Благодаря за това, че с голямото си очакване към мен ти направи този труд много по-значим, отколкото бях предполагал, че може да бъде.

Настоящата докторска работа е подкрепена от:

1. Проект № ДН 07/20/15.12.2016 на тема *Теоретично и експериментално изследване на кристализацията на метална сплав с въведени в нея наночастици*, финансиран от фонд „Научни изследвания“.

2. Bulgarian National Science Fund under Bilateral Project DNTS/Russia 02/12 *Development and investigation of finite difference schemes of higher order of accuracy for solving applied problems of fluid and gas mechanics and ecology*, 2018.

Декларация за оригиналност

Декларирам, че настоящият дисертационен труд *Математическо моделиране на кристализацията на метални сплави* е мое лично дело и че добросъвестно съм посочил всички използвани източници.

Декларирам също така, че съм спазил изискванията за авторско право по отношение на посочените източници и не съм използвал неправомерно чужди текстове в нарушение на авторските им права.

Подпис: .....

Библиография

[1] **А. А. Самарский.** *Об одном экономичном разностном методе решения многомерного параболического уравнения в произвольной области.* Журнал вычислительной математики и математической физики, том 2, номер 5. Москва, октомври 1962

[2] **А. А. Самарский.** *Локально-одномерные разностные схемы на неравномерных сетках.* Журнал вычислительной математики и математической физики, том 3, номер 3. Москва, юни 1963

[3] **А. А. Самарский.** *Введение в теорию разностных схем.* Академический научно-издательский, производственно-полиграфический и книгораспространительский център Российской академии наук „Наука“. Москва, 1971

[4] **А. А. Самарский, Б. Д. Моисеенко.** *Экономичная схема сквозного счета для многомерной задачи Стефана.* Журнал вычислительной математики и математической физики, том 5, номер 5. Москва, септември 1965

[5] **А. А. Самарский, И. М. Соболь.** *Примеры численного расчета температурных волн.* ЖВМ и МФ, 1963, т. 3, №4, стр. 702 — 719

[6] **А. А. Самарский, П. Н. Вабищевич.** *Вычислительная теплопередача.* Издателство „Едиториал УРСС“. Москва, 2002

[7] **А. А. Самарский, Ю. П. Попов.** *Разностные методы решения задач газовой динамики.* Москва, Наука, 1980, стр. 352

[8] **А. М. Мейрманов.** *Задача Стефана.* Издателство „Наука“. Новосибирск, 1986

[9] **А. М. Мейрманов.** *Пример несуществования классического решения задачи Стефана.* Докл. АН СССР, т. 258, №3, стр. 547 — 549. Новосибирск, 1981

[10] **А. Н. Черепанов, В. Н. Попов.** *Моделирование термо- и гидродинамических процессов в модифицированной наночастицами металлической капле при ее соударения с подложкой.* Вестник Удмуртского университета. Ижевск, 2008

[11] **А. Н. Черепанов, В. Н. Попов.** *Численный анализ влияния поверхностно-активного вещества в расплаве на распределение модифицирующих частиц и кристаллизацию при обработке поверхности металла лазерным импульсом.* Списание *Теплофизика и аэромеханика*, том 21, номер 3. Новосибирск, 2014

[12] **А. Н. Черепанов, В. Н. Попов, О. П. Солоненко.** *Объемная кристаллизация капли никеля, содержащей тугоплавкие наночастицы, при соударении с подложкой.* Институт теоретической и прикладной механики. Новосибирск, 2006

[13] **Б. М. Будак, Е. Н. Соловьева, А. Б. Успенский.** *Разностный метод со сглаживанием коэффициентов для решения задачи Стефана.* ЖВМ и МФ, 1965, т. 5, №5, стр. 828 — 840

[14] **В. Манолов, С. Попов.** *Математично моделиране на кристализацията на металнисплави, модифицирани с наномодификатори — алуминиева сплав, стомана, чугун.* София, 2014

[15] **В. Манолов, Р. Лазарова, М. Манчев, С. Попов, С. Станев, Р. Димитрова.** *Изследване влиянието на добавки на прах от TiCN с наноразмери върху свойствата на модифициран сив чугун тип GG25.* International virtual journal for science, technics and innovations for the industry 'Machines, technologies, materials', pp. 9 — 11. София, 2010

[16] **В. И. Мажукин, Ю. А. Повещенко, С. Б. Попов, Ю. П. Попов.** *Об однородных алгоритмах численного решения задачи Стефана.* Препринт ИПМ им. М. В. Кельдша АН СССР, 1985, №122, стр. 23

[17] **В. И. Мажукин, Л. Ю. Такоева.** *Численное решение задачи горения на сетках, динамически адаптирующихся к решенению.* Препринт №74. Москва, 1989

[18] **В. П. Сабуров, А. Н. Черепанов, М. Ф. Жуков и др.** *Плазмохимический синтез ультрадисперсных порошков и их применение для модифицирования металлов и сплавов.* Списание *Наука*, стр. 344. Новосибирск, 1995

[19] **В. Ф. Василевский, В. И. Мажукин.** *Численное решение нестационарной задачи теплопроводности на адаптивной сетке с явным выделением области слабого разрыва.* Препринт №14. Москва, 1989

[20] **В. Д. Дончев.** *Оптимизация на процеса на електронно̀лъчево топене и рафиниране на метали.* Магистърска теза. София, 2014

[21] **Г. И. Баренблатт, И. М. Вишик.** *О конечной скорости распространения в задачах нестационарной фильтрации жидкости и газа.* ПММ, т. 20, №3, стр. 411 — 417. Москва, 1956

[22] **Г. Ф. Баландин.** *Формирование кристаллического строения отливок.* Москва, 1973

[23] **Г. Ф. Баландин.** *Основы теории формирования отливки.* Издателство „Машиностроение“. Москва, 1979

[24] **Д. Лазаров.** *Неорганична химия*, стр. 147 — 159. Университетско издателство „Св. Климент Охридски“. София, 2014

[25] **Д. Рогачева.** *Методи на адаптивните мрежи за числено решаване на класическата задача на Стефан.* Магистърска теза. София, 1994

[26] **Е. Н. Соловьева, А. Б. Успенский.** *Схемы сквозного счета численного решения задач для параболических уравнений с неизвестными границами.* Сб. работ ВЦ МГУ „Вычислительные методы и программирование“, вып. XXIII, 1974

[27] **Ив. Димов, Ц. Рашев, В. Манолов, Бл. Сендов, Р. Лазаров, Б. Чавдаров.** *Изследване на топлообмена при топене на стомана в леяковите системи на металургични машини.* Месечно научно-техническо списание *Металургия*, бр. 4. София, 1981

[28] **Ив. Димов, Ц. Рашев, В. Манолов, Н. Андреев, Бл. Сендов, Р. Лазаров, Ст. Димова, Т. Черногорова.** *Математическо моделиране на затвърдяване на стоманени блокове.* Трудове на Института по черна металургия, том 15, кн. 1. София, 1983

[29] **Ив. Димов, Ц. Рашев, В. Манолов, Н. Андреев, Бл. Сендов, Ст. Димова, Р. Лазаров, Т. Черногорова.** *Математическо моделиране на топлофизическите процеси при получаване на блок от нержда̀ема стомана чрез обработване с газово противоналя̀гане.* Месечно научно-техническо списание *Металургия*, бр. 4. София, 1981

[30] **Ив. Димов, Ц. Рашев, В. Манолов, Т. Черногорова, Ст. Димова.** *Численное решение задачи о кристализации трехмерных слитков.* Численные методы и приложения'84. София, 1985

[31] **Й. Т. Йорданов.** *Mathematica. Преобразувания, изчисления, визуализация.* Издателство „Техника". София, 2013

[32] **Й. Т. Йорданов.** *Matlab 6.7. Преобразувания. Изчисления. Визуализация.* Част 1, 2, 3. Издателство „Техника". София, 2008

[33] **Л. И. Рубинштейн.** *Проблема Стефана.* Рига, Звайгзне, 1976, стр. 457

[34] **Л. С. Лейбензон.** *Собрание трудов.* Изд. АН СССР, т. 4, стр. 399. Москва, 1955

[35] **М. Флемингс.** *Процессы затвердевания.* Издателство „Мир". Москва, 1977

[36] **Н. А. Авдонин.** *Математическое описание процессов кристаллизации.* Издателство „Зинатне". Рига, 1980

[37] **Н. А. Дарьин, В. И. Мажукин.** *Математическое моделирование нестационарной задачи Стефана на адаптивной сетке.* Препринт №52. Москва 1987

[38] **Н. А. Дарьин, В. И. Мажукин.** *Метод построения адаптивных сеток для одномерных кревых задач.* Препринт №33. Москва, 1987

[39] **Н. А. Дарьин, В. И. Мажукин.** *Об одном подходе к построению адаптивных разностных сеток.* ДАН СССР, 1988, т. 298, №1, стр. 64 — 68

[40] **Н. Н. Калиткин.** *Численные методы.* Изд. „Наука", стр. 47 — 51. Москва, 1978

[41] **Р. Лазаров, Ст. Димова, Н. Дренска, Т. Черногорова.** *Математическое моделирование процессов теплообмена и кристаллизации слитков*, vol. 13. Computational Mathematics, Banach Center Publications. Polish Scientifio Publishers. Warsaw, 1984

[42] **Р. Приходанска.** *Числено изследване на процеса на кристализация на рапидна стомана в многослойна кокила.* Дипломна работа. София, 1994

[43] **Т. Черногорова, Н. Андреев, Г. Симеонов, Р. Балчева.** *Кристаллизация инструментальной стали в многослойной изложнице.* Списание *Математическое моделирование*, том 13, номер 4. Российская академия наук, ФГУП „Академиздатцентр Наука“. Москва, 2001

[44] **Т. Черногорова.** *Теория на диференчните схеми* (електронен вариант): `http://www.fmi.uni-sofia.bg/econtent/tds.pdf`. София, 2005

[45] **Т. Черногорова.** *Численное исследование задачи о кристаллизации стального слитка. Одномерное приближение.* Българска академия на науките. Списание *Теоретична и приложна механика*, бр. 1. София, 1987

[46] **Т. Черногорова.** *Числени методи с приложения във финансите* (електронен вариант): `https://intranet.fmi.uni-sofia.bg/index.php/s/BsZlBgb2gZQzqTH`, стр. 80 — 82. София, септември 2016

[47] **Т. П. Черногорова.** *Численное исследование некоторых тепловых и кристаллизационных процессов в металлургии.* Дисертационен труд. Москва, 1987

[48] **Т. П. Черногорова.** *Методът на изчислителния експеримент за изследване на процеси от металургията, лазерната физика, екологията и финансите*, стр. 26 — 44. Хабилитационен труд. София, 2015

[49] **Ф. П. Васильев, А. Б. Успенский.** *Разностный метод решения двухфазной задачи Стефана.* ЖВМ и МФ, 1963, т. 3, №5

[50] **Ф. П. Васильев, А. Б. Успенский.** *О методе конечных разностей для решения двухфазной задачи Стефана для квазилинейного уравнения.* ДАН СССР, 1963, т. 152, №5

[51] **Ф. П. Васильев.** *Разностный метод решения задач типа Стефана для квазилинейного параболического уравнения с разрывными коэффициентами.* ДАН СССР, 1964, т. 157, №6

[52] **Ш. Э. Гусейнов.** *Метод сведения обобщенной задачи Стефана к нелинейному интегро-дифференциальному уравнения типа Волтера.* Computer Modelling and New Technologies, 2006, vol. 10, №2, pp. 57 — 67

[53] **A. N. Kolmogorov.** *On the Statistical Theory of Crystallization of Metals* (in Russian). Izd. Akad. Nauk SSSR. Ser. Mat., No 3, pp. 355 — 359. Moscow, 1937

[54] **A. Gilat.** *MATLAB. An Introduction with Applications.* John Wiley & Sons, Inc. Department of Mechanical Engineering. The Ohio State University. Hoboken, 2011

[55] **B. Mochnacki.** *Numerical modeling of solidification process*

[56] **B. R. Hunt. R. L. Lipsman, J. M. Rosenberg, K. R. Coombes, J. E. Osborn, G. J. Stuck.** *A Guide to Matlab for Beginners and Experienced Users.* Cambridge University Press. Edinburgh, 2001

[57] **G. H. Meyer.** *Multidimensional Stefan Problems.* SIAM J. Numer. Anal., 1973, v. 10, №3, pp. 522 — 538

[58] **G. Lamé, B. P. Clapeyron.** *Memoire sur la solidification par refroidissement d'un globe liquide.* Annales Chimie Physique 47, 1831, pp. 250 — 256

[59] **I. Rafalski, A. Arabey, P. Lushchik, A. S. Chaus.** *Computer modeling of cast alloys solidification by computer-aided cooling curve analysys (CA-CCA).*

[60] **I. S. W. Rogers, A. E. Berger, M. Ciment.** *The alternating phase truncation method for numerical solution of a Stefan Problem.* SIAM J. Numer. Anal., v. 16, №4, pp. 563 — 587. 1979

[61] **Jin-Ju Park, Sung-Mo Hong, Eun-Kwang Park, Kyeong-Yeol Kim, Min-Ku Lee, Chang-Kyu Rhee.** *Microctructure and properties of SA106B carbon steel after treatment of the melt with nanosized TiC particles.* Materials Science and Engineering A 613, pp. 217 – 223. Gongju, 2014

[62] **K. Borodianskiy, M. Zinigrad.** *Mechanical Properties and Microstructure Characterization of Al-Si Cast Alloys Formation Using Carbide Nanoparticles.* Journal of Materials Science and Applications, 1 (3), pp. 85 – 90. New York, 2015

[63] **Maplesoft, a division of Waterloo Maple Inc.** *Maple User Manual.* Waterloo, 2014

[64] **M. Mori.** *Stability and convergence of a finite element method for solving the Stefan problem.* RIMS, Kyoto University, vol. 12, pp. 539 — 563. Kyoto, 1976

[65] **M. Mori.** *A finite element method for solving the two phase Stefan problem in one space dimension.* RIMS, Kyoto University, vol. 13, pp. 723 — 753. Kyoto, 1977

[66] **P. Kuzmanov, A. Velikov, R. Dimitrova, A. Cherepanov, V. Manolov.** *Study of the influence of modification by nanocompositions both on the process of crystallization and on the structure of aluminum alloy AlSi7Mg.* Journal of nanomaterials & molecular nanotechnology, vol. 8, issue 3, 1000271

[67] **P. M., O. Th. Holland.** *Graphics and GUIs with MATLAB*, third edition. Chapman & Hall/CRC Press Company. Washington D.C., 2003

[68] **R. Bonnerot, P. Jamet.** *A second order finite element method for the one-dimensional Stefan problem.* Internat. J. Numer. Methods Engrg., vol. 8, pp. 811 — 820. 1974

[69] **S. Popov et al.** *Journal of Material Science and Technology.* 22, p. 167 – 174. Sofia, 2014

[70] **T. P. Chernogorova, O. P. Iliev.** *An efficient numerical technique for simulating 3D technological solidification processes.* New Science Publisher. New York, 1999

[71] **T. P. Chernogorova, P. N. Vabishchevich.** *Numerical investigation of solidification processes of cylindrical ingots in a metal mould at variable technological circumstances.* International Journal of Heat and Mass Transfer, journal no. 42. Berlin, 1999

[72] **The MathWorks, Inc.** *Creating Graphical User Interface with MatlabR2015b.* Natick, MA, 2015

[73] **The MathWorks, Inc.** *Curve Fitting Toolbox. User's Guide.* Natick, MA, 2018

[74] **The MathWorks, Inc.** *MATLAB. The Language of Technical Computing. Computation. Visualization. Programming.* Natick, MA, 2005

[75] **The MathWorks, Inc.** *Partial Differential Equation Toolbox. For Use with Matlab. User's Guide.* Natick, MA, 1995

[76] **The MathWorks, Inc.** *Partial Differential Equations Toolbox. User's Guide.* Natick, 2016

[77] **V. P. Saburov, E. N. Eremin, A. N. Cherepanov, G. N. Minnehanov.** *Steels and alloys modified with dispersion inoculators.* Publisher of Omsk State Technicak University. Omsk, 2002

[78] **V. Donchev, K. Vutova, T. Chernogorova.** *Economic and Conservative Numerical Scheme for Non-Stationary Heat Model for EBMR.* Sofia, 2014

[79] Вградена функция **deval** за пресмятане на приблизително решение на ОДУ в произволна точка от дефиниционния интервал: `https://www.mathworks.com/help/matlab/ref/deval.html#bu7ixgy-1`

[80] **Задача Стефана:** `https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%A1%D1%82%D0%B5%D1%84%D0%B0%D0%BD%D0%B0`

[81] Многомерна интерполация: `https://en.wikipedia.org/wiki/Bilinear_interpolation`

[82] Научен отчет по проект на тема *Теоретично и експериментално изследване на кристализацията на метална сплав с въведени в нея наночастици*, финансиран от фонд „Научни изследвания“ по договор № ДН 07/20/15.12.2016, етап I: 2016 — 2017

[83] Научен отчет по проект на тема *Изследване на наномодифицирани сплави и тяхното приложение при леене*, финансиран от фонд „Научни изследвания“ по договор № ДО 02.311/19.12.2009, етап II: 2011 — 2013

[84] Продуктова документация на *Simulink*, предоставена от **The MathWorks, Inc**: `https://www.mathworks.com/help/simulink/`

[85] Продуктова документация на *PDE*, предоставена от **The MathWorks, Inc**: `https://www.mathworks.com/help/pde/`

Приложение

I. Скриптов файл, реализиращ пресмятането на функцията $\psi(u)$

```
function res = f_psi(u,u_L,u_S)
C0 = 0.07; u_tilde = 660;
c(1) = 0.0165;
a(1) = c(1) / (u_S - u_tilde);
b(1) = -a(1)*u_tilde;
c(2) = 0.116;
a(2) = c(2) / (u_S - u_tilde);
b(2) = -a(2)*u_tilde;
n = length(u);
for i = 1 : n
    if(u(i)>u_L)
        res(i) = 0;
    elseif(u(i)>=u_S && u(i)<=u_L)
            res(i) = a(2) * (u(i) - u_L) ./ ...
                ((a(2) - a(1)) * u(i) + ...
                a(1) * u_S - a(2) * u_L);
    elseif(u(i)<u_S)
        res(i) = 1;
    end
end
end
```

II. Скриптов файл, реализиращ пресмятането на функцията $\frac{\mathrm{d}\psi}{\mathrm{d}u}$

```
function res = dPsi(u,u_L,u_S)
C0 = 0.07; u_tilde = 660;
c(1) = 0.0165;
a(1) = c(1) / (u_S - u_tilde);
b(1) = -a(1)*u_tilde;
c(2) = 0.116;
a(2) = c(2) / (u_S - u_tilde);
b(2) = -a(2)*u_tilde;
n = length(u);
for i = 1 : n
    if(u(i) > u_L)
```

```
        res(i) = 0;
    elseif(u(i) >= u_S && u(i) <= u_L)
        res(i) = (a(1) * a(2) * (u_S - u_L)) ./ ...
            ((a(2)-a(1)) * u(i) + ...
            a(1) * u_S - a(2) * u_L).^2;
    elseif(u(i) < u_S)
        res(i) = 0;
    end
end
end
```

III. Файл-функция, реализираща пресмятането на параметъра Δ

```
function res = delta_val(var,u,u_S,delta_old)
    u_Left = 0;
    u_Right = 0;
    for i = 2 : length(u)
        if((u(i-1) - u_S) * (u(i) - u_S) <= 0)
            u_Left = u(i-1);
            u_Right = u(i);
            break;
        end
    end
    if(u_Left > u_Right) % decreasing
        if(u_Left - u_S > u_S - u_Right)
            delta = (u_S - u_Right);
            res = delta;
        elseif(u_S - u_Right > u_Left - u_S)
            delta = (u_Left - u_S);
            res = delta;
        end
    elseif(u_Left < u_Right) % increasing
        if(u_S - u_Left < u_Right - u_S)
            delta = (u_S - u_Left);
            res = delta;
        elseif(u_S - u_Left > u_Right - u_S)
            delta = (u_Right - u_S);
            res = delta;
        end
    end % of if
end % of the function
```

IV. Файл-функция, реализираща пресмятането на коефициента на топлообмен $\alpha_k^{(1)}$

```
function res = alpha_k_1_val_u(r,R_k,u, ...
                IndPair,n,u_S,lambda_M,delta_M,delta_0)
%=========================================
% n - the number of the time layer
```

```
%=========================================
%res = 0;
p = lambda_M / delta_M;
value = find_bark_u(r,R_k,u,n,u_S,IndPair);
if(value<delta_0 || isempty(value))
    res = +10^4;
elseif(value>=delta_0)
    res = +p;
end
end
```

V. Файл-функция, реализираща пресмятането на дебелината на твърдата фаза

```
function res = find_bark_u(r,R_k,u,n,u_S,IndPair)
res = [];
%=========================================
N = length(r);
for i = N-1 : -1 : 1
    if((u(i,IndPair,n) - u_S) * (u(i+1,IndPair,n) - u_S) < 0)
        xdata = [r(i),r(i+1)];
        ydata = [u(i,IndPair,n),u(i+1,IndPair,n)];
        %=========================================
        % Linear interpolation
        %=========================================
        r_S = approx_val(xdata,ydata,u_S);
        %=========================================
        % The value of the bark
        %=========================================
        res = R_k - r_S;
    end
end
if(isempty(res))
    res = 0;
end % of if
end % of function
```

VI. Файл-функция, реализираща метода на дясната прогонка

```
function res = Progon(A, d)
% PROGON(A,d) solves the linear system A*x = d
% whose matrix is tridiagonal. This function
% uses the Progonka method for fast and time-save
% solving the system.
%
% INPUT:
% A - the matrix of the linear system A*x = d
% d - the right side of the linear system A*x = d
%
% OUTPUT:
% the solution x of the linear system A*x = d
```

```
13 %
14 % EXAMPLE:
15 % >> A = [-4,2,0,0; ...
16 %           1,3,-1,0; ...
17 %           0,1,-7,-2; ...
18 %           0,0,-9,10];
19 % >> d = [1; 3; -1; 0];
20 % >> x = Progon(A,d);
21 % >> display('Check the obtained answer: ');
22 % >> abs(A*x-d)
23 %     ans =
24 %           1.0e-015 *
25 %
26 %                  0
27 %                  0
28 %             0.2220
29 %             0.4441
30 % See also ITER.
31
32 % b vector
33 b = diag(A);
34 s = size(A);
35 % a vector
36 for j = 1 : s(1, 2)-1
37     a(j) = A(j+1, j);
38 end
39 a = [0, a];
40 % c vector
41 for i = 1 : s(1, 1)-1
42     c(i) = A(i, i+1);
43 end
44 c = [c, 0];
45 alpha(1) = -c(1)/b(1);
46 beta(1) = d(1)/b(1);
47 n = length(d);
48 for k = 2 : n
49     alpha(k) = -c(k)/(b(k)+a(k)*alpha(k-1));
50     beta(k) = (d(k)-a(k)*beta(k-1)) / ...
51                 (b(k)+a(k)*alpha(k-1));
52 end
53 x(n) = beta(n);
54 for k = n-1 : -1 : 1
55     x(k) = alpha(k)*x(k+1)+beta(k);
56 end
57 res = x';
58 end
```

VII. Файл-функция, реализираща пресмятането на нелинейния коефициент $c(u)$ в направление r

```
function res = coefficient_c_r(i,jj,P,Q,r,hr,Sol, ...
                u_L,u_S,c_1,c_2,c_3,L, ...
                psi_E,delta_old)
    my_domain = domain(i,jj,P,Q);
    switch my_domain
        case 1
            L_E = L;
            c_L = c_1;
            c_S = c_1;
            nS = length(find(Sol(1:Q)<=u_S));
            %=======================================
            % The middle solutions
            %=======================================
            if(i==1)
                u_Mid_Left = Sol(1);
                u_Mid_Right = ...
                    (Sol(1) + Sol(2)) / 2;
                r_Left = 0;
                r_Right = r(1)+hr(2)/2;
                step = r_Right - r_Left;
            elseif(i==Q)
                u_Mid_Left = ...
                    (Sol(Q-1) + Sol(Q)) / 2;
                u_Mid_Right = Sol(Q);
                r_Left = r(Q)-hr(Q)/2;
                r_Right = r(Q);
                step = r_Right - r_Left;
            elseif(i>1 && i<Q)
                u_Mid_Left = ...
                    (Sol(i-1) + Sol(i)) / 2;
                u_Mid_Right = ...
                    (Sol(i) + Sol(i+1)) / 2;
                r_Left = r(i)-hr(i)/2;
                r_Right = r(i)+hr(i+1)/2;
                step = r_Right - r_Left;
            end % of the middle solutions
            if(u_Mid_Left >= u_Mid_Right) % decreasing

                %=======================================
                % 1 CASE: nS==0
                %=======================================
                if(nS==0)
                    %=======================================
                    % (1)
                    %=======================================
                    if(u_Mid_Left > u_L && u_Mid_Right > u_L)
                        res = c_L * (r_Right - r_Left);
                        %display('1.1,decr')
                        res = 1/step * res;
```

```
50                      %=========================================
51                      % (2)
52                      %=========================================
53                      elseif(u_Mid_Left > u_L && ...
54                              isInside(u_Mid_Right,u_S,u_L))
55                          %=========================================
56                          % r_L
57                          %=========================================
58                          xdata = [r_Left,r_Right];
59                          ydata = [u_Mid_Left,u_Mid_Right];
60                          r_L = approx_val(xdata,ydata,u_L);
61                          res = c_L * (r_L - r_Left) + ...
62                              c_S * (r_Right - r_L) - ...
63                              L * (r_Right - r_L) / 2 * ...
64                              (dPsi(u_L,u_L,u_S) + ...
65                              dPsi(u_Mid_Right,u_L,u_S));
66                          %display('1.2,decr')
67                          res = 1/step * res;
68                      %=========================================
69                      % (3)
70                      %=========================================
71                      elseif(isInside(u_Mid_Left,u_S,u_L) && ...
72                              isInside(u_Mid_Right,u_S,u_L))
73                          res = c_S * (r_Right - r_Left) - ...
74                              L * (r_Right - r_Left) / 2 * ...
75                              (dPsi(u_Mid_Left,u_L,u_S) + ...
76                              dPsi(u_Mid_Right,u_L,u_S));
77                          %display('1.3,decr')
78                          res = 1/step * res;
79                      end
80                  %=========================================
81                  % 2 CASE: nS!=0 && nS!=N
82                  %=========================================
83                  elseif(nS~=0 && nS~=Q)
84                      delta = ...
85                              delta_val(r(1:Q),Sol(1:Q), ...
86                                      u_S,delta_old);
87                      %=========================================
88                      % (1)
89                      %=========================================
90                      if(u_Mid_Left > u_L && u_Mid_Right > u_L)
91                          res = c_L * (r_Right - r_Left);
92                          %display('2.1,decr')
93                          res = 1/step * res;
94                      %=========================================
95                      % (2)
96                      %=========================================
97                      elseif(u_Mid_Left > u_L && ...
98                              isInside(u_Mid_Right,u_S+delta,u_L))
```

```
99                      %=========================================
100                     % r_L
101                     %=========================================
102                     xdata = [r_Left,r_Right];
103                     ydata = [u_Mid_Left,u_Mid_Right];
104                     r_L = approx_val(xdata,ydata,u_L);
105                     res = c_L * (r_L - r_Left) + ...
106                         c_S * (r_Right - r_L) - ...
107                         L * (r_Right - r_L) / 2 * ...
108                         (dPsi(u_L,u_L,u_S) + ...
109                         dPsi(u_Mid_Right,u_L,u_S));
110                     %display('2.2,decr')
111                     res = 1/step * res;
112                 %=========================================
113                 % (3)
114                 %=========================================
115                 elseif(u_Mid_Left > u_L && ...
116                         isInside(u_Mid_Right,...
117                                  u_S-delta,u_S+delta))
118                     %=========================================
119                     % r_S_Up
120                     %=========================================
121                     if(u_Mid_Right==u_S+delta)
122                         r_S_Up = r_Right;
123                     elseif(u_Mid_Left==u_S+delta)
124                         r_S_Up = r_Left;
125                     else
126                         xdata = [r_Left,r_Right];
127                         ydata = [u_Mid_Left,u_Mid_Right];
128                         r_S_Up = ...
129                                 approx_val(xdata,ydata, ...
130                                            u_S+delta);
131                     end
132                     %=========================================
133                     % r_L
134                     %=========================================
135                     xdata = [r_Left,r_S_Up];
136                     ydata = [u_Mid_Left,u_S+delta];
137                     r_L = approx_val(xdata,ydata,u_L);
138                     res = c_L * (r_L - r_Left) + ...
139                         c_S * (r_S_Up - r_L) - ...
140                         L * (r_S_Up - r_L) / 2 * ...
141                         (dPsi(u_L,u_L,u_S) + ...
142                         dPsi(u_S+delta,u_L,u_S)) + ...
143                         (r_Right - r_S_Up) * ...
144                         (L_E * psi_E / (2 * delta) + ...
145                         c_S - ...
146                         L / (2 * delta) * ...
147                         (f_psi(u_S+delta,u_L,u_S) - ...
```

```
148                         f_psi(u_S,u_L,u_S)));
149                     %display('2.3,decr')
150                     res = 1/step * res;
151                 %=========================
152                 % (4)
153                 %=========================
154                 elseif(u_Mid_Left > u_L && ...
155                         u_Mid_Right < u_S-delta)
156                     %=========================
157                     % r_S_Up
158                     %=========================
159                     if(u_Mid_Right==u_S+delta)
160                         r_S_Up = r_Right;
161                     elseif(u_Mid_Left==u_S+delta)
162                         r_S_Up = r_Left;
163                     else
164                         xdata = [r_Left,r_Right];
165                         ydata = [u_Mid_Left,u_Mid_Right];
166                         r_S_Up = ...
167                                 approx_val(xdata,ydata, ...
168                                         u_S+delta);
169                     end
170                     %=========================
171                     % r_S_Down
172                     %=========================
173                     if(u_Mid_Right==u_S-delta)
174                         r_S_Down = r_Right;
175                     elseif(u_Mid_Left==u_S-delta)
176                         r_S_Down = r_Left;
177                     else
178                         xdata = [r_Left,r_Right];
179                         ydata = [u_Mid_Left,u_Mid_Right];
180                         r_S_Down = ...
181                                 approx_val(xdata,ydata, ...
182                                         u_S-delta);
183                     end
184                     %=========================
185                     % r_L
186                     %=========================
187                     xdata = [r_Left,r_S_Up];
188                     ydata = [u_Mid_Left,u_S+delta];
189                     r_L = approx_val(xdata,ydata,u_L);
190                     res = c_L * (r_L - r_Left) + c_S * ...
191                         (r_S_Up - r_L) - ...
192                         L * (r_S_Up - r_L) / 2 * ...
193                         (dPsi(u_L,u_L,u_S) + ...
194                         dPsi(u_S+delta,u_L,u_S)) + ...
195                         (r_S_Down - r_S_Up) * ...
196                         (L_E * psi_E / (2 * delta) + ...
```

```
197                         c_S - L / (2 * delta) * ...
198                         (f_psi(u_S+delta,u_L,u_S) - ...
199                         f_psi(u_S,u_L,u_S)));
200                     %display('2.4,decr')
201                     res = 1/step * res;
202                 %=========================================
203                 % (5)
204                 %=========================================
205                 elseif(isInside(u_Mid_Left,u_S+delta,u_L) && ...
206                         isInside(u_Mid_Right,u_S+delta,u_L))
207                     res = c_S * (r_Right - r_Left) - ...
208                         L * (r_Right - r_Left) / 2 * ...
209                         (dPsi(u_Mid_Left,u_L,u_S) + ...
210                         dPsi(u_Mid_Right,u_L,u_S));
211                     %display('2.5,decr')
212                     res = 1/step * res;
213                 %=========================================
214                 % (6)
215                 %=========================================
216                 elseif(isInside(u_Mid_Left,u_S+delta,u_L) && ...
217                         isInside(u_Mid_Right, ...
218                                  u_S-delta,u_S+delta))
219                     %=====================================
220                     % r_S_Up
221                     %=====================================
222                     if(u_Mid_Right==u_S+delta)
223                         r_S_Up = r_Right;
224                     elseif(u_Mid_Left==u_S+delta)
225                         r_S_Up = r_Left;
226                     else
227                         xdata = [r_Left,r_Right];
228                         ydata = [u_Mid_Left,u_Mid_Right];
229                         r_S_Up =
230                                 approx_val(xdata,ydata, ...
231                                            u_S+delta);
232                     end
233                     res = c_S * (r_S_Up - r_Left) - ...
234                         L * (r_S_Up - r_Left) / 2 * ...
235                         (dPsi(u_Mid_Left,u_L,u_S) + ...
236                         dPsi(u_S+delta,u_L,u_S)) + ...
237                         (r_Right - r_S_Up) * ...
238                         (L_E * psi_E / (2 * delta) + ...
239                         c_S - ...
240                         L / (2 * delta) * ...
241                         (f_psi(u_S+delta,u_L,u_S) - ...
242                         f_psi(u_S,u_L,u_S)));
243                     %display('2.6,decr')
244                     res = 1/step * res;
245                 %=========================================
```

```
246                     % (7)
247                     %=========================================
248                     elseif(isInside(u_Mid_Left,u_S+delta,u_L) && ...
249                             u_Mid_Right < u_S-delta)
250                         %=========================================
251                         % r_S_Up
252                         %=========================================
253                         if(u_Mid_Right==u_S+delta)
254                             r_S_Up = r_Right;
255                         elseif(u_Mid_Left==u_S+delta)
256                             r_S_Up = r_Left;
257                         else
258                             xdata = [r_Left,r_Right];
259                             ydata = [u_Mid_Left,u_Mid_Right];
260                             r_S_Up = ...
261                                     approx_val(xdata,ydata, ...
262                                                 u_S+delta);
263                         end
264                         %=========================================
265                         % r_S_Down
266                         %=========================================
267                         if(u_Mid_Right==u_S-delta)
268                             r_S_Down = r_Right;
269                         elseif(u_Mid_Left==u_S-delta)
270                             r_S_Down = r_Left;
271                         else
272                             xdata = [r_Left,r_Right];
273                             ydata = [u_Mid_Left,u_Mid_Right];
274                             r_S_Down = ...
275                                     approx_val(xdata,ydata, ...
276                                                 u_S-delta);
277                         end
278                         res = c_S * (r_S_Up - r_Left) - ...
279                             L * (r_S_Up - r_Left) / 2 * ...
280                             (dPsi(u_Mid_Left,u_L,u_S) + ...
281                             dPsi(u_S+delta,u_L,u_S)) + ...
282                             (r_S_Down - r_S_Up) * ...
283                             (L_E * psi_E / (2 * delta) + ...
284                             c_S - ...
285                             L / (2 * delta) * ...
286                             (f_psi(u_S+delta,u_L,u_S) - ...
287                             f_psi(u_S,u_L,u_S))) + ...
288                             c_S * (r_Right - r_S_Down);
289                         %display('2.7,decr')
290                         res = 1/step * res;
291                     %=========================================
292                     % (8)
293                     %=========================================
294                     elseif(isInside(u_Mid_Left, ...
```

```
295                                 u_S-delta,u_S+delta) && ...
296                         isInside(u_Mid_Right, ...
297                                 u_S-delta,u_S+delta))
298                     res = (r_Right - r_Left) * ...
299                         (L_E * psi_E / (2 * delta) + ...
300                         c_S - ...
301                         L / (2 * delta) * ...
302                         (f_psi(u_S+delta,u_L,u_S) - ...
303                         f_psi(u_S,u_L,u_S)));
304                     %display('2.8,decr')
305                     res = 1/step * res;
306                 %=========================================
307                 % (9)
308                 %=========================================
309                 elseif(isInside(u_Mid_Left, ...
310                                 u_S-delta,u_S+delta) && ...
311                                 u_Mid_Right < u_S-delta)
312                     %=====================================
313                     % r_S_Down
314                     %=====================================
315                     if(u_Mid_Right==u_S-delta)
316                         r_S_Down = r_Right;
317                     elseif(u_Mid_Left==u_S-delta)
318                         r_S_Down = r_Left;
319                     else
320                         xdata = [r_Left,r_Right];
321                         ydata = [u_Mid_Left,u_Mid_Right];
322                         r_S_Down = ...
323                                 approx_val(xdata,ydata, ...
324                                         u_S-delta);
325                     end
326                     res = (r_S_Down - r_Left) * ...
327                         (L_E * psi_E / (2 * delta) + ...
328                         c_S - ...
329                         L / (2 * delta) * ...
330                         (f_psi(u_S+delta,u_L,u_S) - ...
331                         f_psi(u_S,u_L,u_S))) + ...
332                         c_S * (r_Right - r_S_Down);
333                     %display('2.9,decr')
334                     res = 1/step * res;
335                 %=========================================
336                 % (10)
337                 %=========================================
338                 elseif(u_Mid_Left < u_S-delta && ...
339                         u_Mid_Right < u_S-delta)
340                     res = c_S * (r_Right - r_Left);
341                     %display('2.10,decr')
342                     res = 1/step * res;
343                 end
```

```
344                 %=========================================
345                 % 3 CASE: nS!=0 && nS==N
346                 %=========================================
347                 elseif(nS~=0 && nS==Q)
348                     res = c_S * (r_Right - r_Left);
349                     %display('3.1,decr')
350                     res = 1/step * res;
351
352                 end % of the cases for nS for decreasing
353
354             elseif(u_Mid_Left < u_Mid_Right) % increasing
355
356                 %=========================================
357                 % 1 CASE: nS==0
358                 %=========================================
359                 if(nS==0)
360                     %=====================================
361                     % (1)
362                     %=====================================
363                     if(isInside(u_Mid_Left,u_S,u_L) && ...
364                             isInside(u_Mid_Right,u_S,u_L))
365                         res = c_S * (r_Right - r_Left) - ...
366                             L * (r_Right - r_Left) / 2 * ...
367                             (dPsi(u_Mid_Left,u_L,u_S) + ...
368                             dPsi(u_Mid_Right,u_L,u_S));
369                         %display('1.1,incr')
370                         res = 1/step * res;
371                     %=====================================
372                     % (2)
373                     %=====================================
374                     elseif(isInside(u_Mid_Left,u_S,u_L) && ...
375                             u_Mid_Right > u_L)
376                         %=================================
377                         % r_L
378                         %=================================
379                         xdata = [r_Left,r_Right];
380                         ydata = [u_Mid_Left,u_Mid_Right];
381                         r_L = approx_val(xdata,ydata,u_L);
382                         res = c_S * (r_L - r_Left) - ...
383                             L * (r_L - r_Left) / 2 * ...
384                             (dPsi(u_Mid_Left,u_L,u_S) + ...
385                             dPsi(u_L,u_L,u_S)) + ...
386                             c_L * (r_Right - r_L);
387                         %display('1.2,incr')
388                         res = 1/step * res;
389                     %=====================================
390                     % (3)
391                     %=====================================
392                     elseif(u_Mid_Left > u_L && ...
```

```
                    u_Mid_Right > u_L)
                res = c_L * (r_Right - r_Left);
                %display('1.3,incr')
                res = 1/step * res;
            end
        %=========================================
        % 2 CASE: nS!=0
        %=========================================
        elseif(nS~=0 && nS~=Q)
            delta = ...
                    delta_val(r(1:Q),Sol(1:Q), ...
                              u_S,delta_old);
            %=========================================
            % (1)
            %=========================================
            if(u_Mid_Left < u_S-delta && ...
                    u_Mid_Right < u_S-delta)
                res = c_S * (r_Right - r_Left);
                %display('2.1,incr')
                res = 1/step * res;
            %=========================================
            % (2)
            %=========================================
            elseif(u_Mid_Left < u_S-delta && ...
                    isInside(u_Mid_Right,...
                             u_S-delta,u_S+delta))
                %=========================================
                % r_S_Down
                %=========================================
                if(u_Mid_Right==u_S-delta)
                    r_S_Down = r_Right;
                elseif(u_Mid_Left==u_S-delta)
                    r_S_Down = r_Left;
                else
                    xdata = [r_Left,r_Right];
                    ydata = [u_Mid_Left,u_Mid_Right];
                    r_S_Down = ...
                                approx_val(xdata,ydata, ...
                                           u_S-delta);
                end
                res = c_S * (r_S_Down - r_Left) + ...
                    (r_Right - r_S_Down) * ...
                    (L_E * psi_E / (2 * delta) + ...
                    c_S - ...
                    L / (2 * delta) * ...
                    (f_psi(u_S+delta,u_L,u_S) - ...
                    f_psi(u_S,u_L,u_S)));
                %display('2.2,incr')
                res = 1/step * res;
```

```
442                     %=========================================
443                     % (3)
444                     %=========================================
445                     elseif(u_Mid_Left < u_S-delta && ...
446                             isInside(u_Mid_Right,u_S+delta,u_L))
447                         %=========================================
448                         % r_S_Down
449                         %=========================================
450                         if(u_Mid_Right==u_S-delta)
451                             r_S_Down = r_Right;
452                         elseif(u_Mid_Left==u_S-delta)
453                             r_S_Down = r_Left;
454                         else
455                             xdata = [r_Left,r_Right];
456                             ydata = [u_Mid_Left,u_Mid_Right];
457                             r_S_Down = ...
458                                     approx_val(xdata,ydata, ...
459                                             u_S-delta);
460                         end
461                         %=========================================
462                         % r_S_Up
463                         %=========================================
464                         if(u_Mid_Right==u_S+delta)
465                             r_S_Up = r_Right;
466                         elseif(u_Mid_Left==u_S+delta)
467                             r_S_Up = r_Left;
468                         else
469                             xdata = [r_Left,r_Right];
470                             ydata = [u_Mid_Left,u_Mid_Right];
471                             r_S_Up = ...
472                                     approx_val(xdata,ydata, ...
473                                             u_S+delta);
474                         end
475                         res = c_S * (r_S_Down - r_Left) + ...
476                             (r_S_Up - r_S_Down) * ...
477                             (L_E * psi_E / (2 * delta) + ...
478                             c_S - ...
479                             L / (2 * delta) * ...
480                             (f_psi(u_S+delta,u_L,u_S) - ...
481                             f_psi(u_S,u_L,u_S))) + ...
482                             c_S * (r_Right - r_S_Up) - ...
483                             L * (r_Right - r_S_Up) / 2 * ...
484                             (dPsi(u_S+delta,u_L,u_S) + ...
485                             dPsi(u_Mid_Right,u_L,u_S));
486                         %display('2.3,incr')
487                         res = 1/step * res;
488                     %=========================================
489                     % (4)
490                     %=========================================
```

```
491                     elseif(u_Mid_Left < u_S-delta && ...
492                             u_Mid_Right > u_L)
493                         %=======================================
494                         % r_S_Down
495                         %=======================================
496                         if(u_Mid_Right==u_S-delta)
497                             r_S_Down = r_Right;
498                         elseif(u_Mid_Left==u_S-delta)
499                             r_S_Down = r_Left;
500                         else
501                             xdata = [r_Left,r_Right];
502                             ydata = [u_Mid_Left,u_Mid_Right];
503                             r_S_Down = ...
504                                 approx_val(xdata,ydata, ...
505                                     u_S-delta);
506                         end
507                         %=======================================
508                         % r_S_Up
509                         %=======================================
510                         if(u_Mid_Right==u_S+delta)
511                             r_S_Up = r_Right;
512                         elseif(u_Mid_Left==u_S+delta)
513                             r_S_Up = r_Left;
514                         else
515                             xdata = [r_Left,r_Right];
516                             ydata = [u_Mid_Left,u_Mid_Right];
517                             r_S_Up = ...
518                                 approx_val(xdata,ydata, ...
519                                     u_S+delta);
520                         end
521                         %=======================================
522                         % r_L
523                         %=======================================
524                         xdata = [r_Right,r_S_Up];
525                         ydata = [u_Mid_Right,u_S+delta];
526                         r_L = approx_val(xdata,ydata,u_L);
527                         res = c_S * (r_S_Down - r_Left) + ...
528                             (r_S_Up - r_S_Down) * ...
529                             (L_E * psi_E / (2 * delta) + ...
530                             c_S - ...
531                             L / (2 * delta) * ...
532                             (f_psi(u_S+delta,u_L,u_S) - ...
533                             f_psi(u_S,u_L,u_S))) + ...
534                             c_S * (r_L - r_S_Up) - ...
535                             L * (r_L - r_S_Up) / 2 * ...
536                             (dPsi(u_S+delta,u_L,u_S) + ...
537                             dPsi(u_L,u_L,u_S)) + ...
538                             c_L * (r_Right - r_L);
539                         %display('2.4,incr')
```

```
540                     res = 1/step * res;
541                 %=========================================
542                 % (5)
543                 %=========================================
544                 elseif(isInside(u_Mid_Left, ...
545                                 u_S-delta,u_S+delta) && ...
546                        isInside(u_Mid_Right, ...
547                                 u_S-delta,u_S+delta))
548                     res = (r_Right - r_Left) * ...
549                         (L_E * psi_E / (2 * delta) + ...
550                         c_S - ...
551                         L / (2 * delta) * ...
552                         (f_psi(u_S+delta,u_L,u_S) - ...
553                         f_psi(u_S,u_L,u_S)));
554                     %display('2.5,incr')
555                     res = 1/step * res;
556                 %=========================================
557                 % (6)
558                 %=========================================
559                 elseif(isInside(u_Mid_Left, ...
560                                 u_S-delta,u_S+delta) && ...
561                        isInside(u_Mid_Right, ...
562                                 u_S+delta,u_L))
563                     %=====================================
564                     % r_S_Up
565                     %=====================================
566                     if(u_Mid_Right==u_S+delta)
567                         r_S_Up = r_Right;
568                     elseif(u_Mid_Left==u_S+delta)
569                         r_S_Up = r_Left;
570                     else
571                         xdata = [r_Left,r_Right];
572                         ydata = [u_Mid_Left,u_Mid_Right];
573                         r_S_Up = ...
574                                 approx_val(xdata,ydata, ...
575                                            u_S+delta);
576                     end
577                     res = (r_S_Up - r_Left) * ...
578                         (L_E * psi_E / (2 * delta) + ...
579                         c_S - ...
580                         L / (2 * delta) * ...
581                         (f_psi(u_S+delta,u_L,u_S) - ...
582                         f_psi(u_S,u_L,u_S))) + ...
583                         c_S * (r_Right - r_S_Up) - ...
584                         L * (r_Right - r_S_Up) / 2 * ...
585                         (dPsi(u_S+delta,u_L,u_S) + ...
586                         dPsi(u_Mid_Right,u_L,u_S));
587                     %display('2.6,incr')
588                     res = 1/step * res;
```

```
589                     %=========================================
590                     % (7)
591                     %=========================================
592                     elseif(isInside(u_Mid_Left, ...
593                                     u_S-delta,u_S+delta) && ...
594                         u_Mid_Right > u_L)
595                         %=========================================
596                         % r_S_Up
597                         %=========================================
598                         if(u_Mid_Right==u_S+delta)
599                             r_S_Up = r_Right;
600                         elseif(u_Mid_Left==u_S+delta)
601                             r_S_Up = r_Left;
602                         else
603                             xdata = [r_Left,r_Right];
604                             ydata = [u_Mid_Left,u_Mid_Right];
605                             r_S_Up = ...
606                                     approx_val(xdata,ydata, ...
607                                                u_S+delta);
608                         end
609                         %=========================================
610                         % r_L
611                         %=========================================
612                         xdata = [r_Right,r_S_Up];
613                         ydata = [u_Mid_Right,u_S+delta];
614                         r_L = approx_val(xdata,ydata,u_L);
615                         res = (r_S_Up - r_Left) * ...
616                             (L_E * psi_E / (2 * delta) + ...
617                             c_S - ...
618                             L / (2 * delta) * ...
619                             (f_psi(u_S+delta,u_L,u_S) - ...
620                             f_psi(u_S,u_L,u_S))) + ...
621                             c_S * (r_L - r_S_Up) - ...
622                             L * (r_L - r_S_Up) / 2 * ...
623                             (dPsi(u_S+delta,u_L,u_S) + ...
624                             dPsi(u_L,u_L,u_S)) + ...
625                             c_L * (r_Right - r_L);
626                         %display('2.7,incr')
627                         res = 1/step * res;
628                     %=========================================
629                     % (8)
630                     %=========================================
631                     elseif(isInside(u_Mid_Left,u_S+delta,u_L) && ...
632                             isInside(u_Mid_Right,u_S+delta,u_L))
633                         res = c_S * (r_Right - r_Left) - ...
634                             L * (r_Right - r_Left) / 2 * ...
635                             (dPsi(u_Mid_Right,u_L,u_S) + ...
636                             dPsi(u_Mid_Left,u_L,u_S));
637                         %display('2.8,incr')
```

```
638                         res = 1/step * res;
639                     %=============================
640                     % (9)
641                     %=============================
642                     elseif(isInside(u_Mid_Left,u_S+delta,u_L) && ...
643                             u_Mid_Right > u_L)
644                         %=============================
645                         % r_L
646                         %=============================
647                         xdata = [r_Left,r_Right];
648                         ydata = [u_Mid_Left,u_Mid_Right];
649                         r_L = approx_val(xdata,ydata,u_L);
650                         res = c_S * (r_L - r_Left) - ...
651                             L * (r_L - r_Left) / 2 * ...
652                             (dPsi(u_Mid_Left,u_L,u_S) + ...
653                             dPsi(u_L,u_L,u_S)) + ...
654                             c_L * (r_Right - r_L);
655                         %display('2.9,incr')
656                         res = 1/step * res;
657                     %=============================
658                     % (10)
659                     %=============================
660                     elseif(u_Mid_Left > u_L && ...
661                             u_Mid_Right > u_L)
662                         res = c_L * (r_Right - r_Left);
663                         %display('2.10,incr')
664                         res = 1/step * res;
665                     end
666                 %=============================
667                 % 3 CASE: nS!=0 && nS==N
668                 %=============================
669                 elseif(nS~=0 && nS==Q)
670                     res = c_S * (r_Right - r_Left);
671                     %display('3.1,incr')
672                     res = 1/step * res;
673                 end % of the cases for nS for increasing
674 
675             end
676         case 2
677             res = c_2;
678         case 3
679             res = c_3;
680     end % of switch
681 end % of the function
```

VIII. Скриптов файл, реализиращ линеаризираната чисто неявна диференчна схема с ред на сходимост $O\big(\|h\|^2+\tau\big)$

```
1 %=================================
2 % Shifted grid, locally 1D scheme
```

```
%=========================================
clear all;
%clc;
tic;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NON-UNIFORM SHIFTED GRID FOR r
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%=========================================
% r, hr
%=========================================
R_k = 0.02865; R = 0.03015;
ar = 0; br = R;
hr(1) = 0; hr(2) = 0.0007;
r(1) = hr(2)/2;
r(2) = r(1) + hr(2);
i = 2;
step = 0.001;
while(r(i)<=R_k)
    i = i+1;
    hr(i) = step;
    r(i) = r(i-1) + hr(i);
end
Q = length(r);
if(r(Q)-r(Q-1)==step)
    r(end) = [];
    hr(end) = [];
    Q = length(r);
    r(Q+1) = r(Q);
    hr(Q+1) = r(Q+1)-r(Q);
else
   r(Q) = R_k; hr(Q) = r(Q)-r(Q-1);
   r(Q+1) = r(Q); hr(Q+1) = r(Q+1)-r(Q);
end
i = Q+1;
step = 0.001;
while(r(i)<=R)
    i = i+1;
    hr(i) =  step;
    r(i) = r(i-1) + hr(i);
end
N = length(r);
if(r(N)-r(N-1)==step)
    r(end) = [];
    hr(end) = [];
    N = length(r);
    hr(N) = r(N)-r(N-1);
else
   r(N) = R; hr(N) = r(N) - r(N-1);
end
```

```
%=========================================
% hLine
%=========================================
hrLine(1) = hr(2);
for i = 2 : N-1
    hrLine(i) = 0.5 * (hr(i) + hr(i+1));
end
hrLine(N) = hr(N)/2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NON-UNIFORM GRID FOR z
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h = 0.015; H = 0.099;
%az = 0; bz = H;
hz(1) = 0; hz(2) = 0.002;
z(1) = 0;
z(2) = z(1) + hz(2);
j = 2;
step = 0.0028;
while(z(j)<=h)
    j = j+1;
    hz(j) = step;
    z(j) = z(j-1) + hz(j);
end
P = length(z);
if(z(P)-z(P-1)==step)
    z(end) = [];
    hz(end) = [];
    P = length(z);
    z(P+1) = z(P);
    hz(P+1) = z(P+1)-z(P);
else
   z(P) = h; hz(P) = z(P)-z(P-1);
   z(P+1) = z(P); hz(P+1) = z(P+1)-z(P);
end
j = P+1;
step = 0.003;
while(z(j)<=H)
    j = j+1;
    hz(j) =  step;
    z(j) = z(j-1) + hz(j);
end
M = length(z);
if(z(M)-z(M-1)==step)
    z(end) = [];
    hz(end) = [];
    M = length(z);
    hz(M) = z(M)-z(M-1);
else
   z(M) = H; hz(M) = z(M) - z(M-1);
```

```
101 end
102 %=========================================
103 % hLine
104 %=========================================
105 hzLine(1) = hz(2)/2;
106 for j = 2 : M-1
107     hzLine(j) = 0.5 * (hz(j) + hz(j+1));
108 end
109 hzLine(M) = hz(M)/2;
110 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
111 % UNIFORM GRID FOR t
112 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113 t0 = 0; tend = 743.6;
114 tau = 0.2;
115 t = t0 : tau : tend;
116 T = length(t);
117 %=========================================
118 % Constants
119 %=========================================
120 index = 0;
121 c_1 = 1.08 * 10^3;
122 c_2 = 1.08 * 10^3;
123 c_3 = 753;
124 c_S = c_1; c_L = c_1;
125 c_vec = [c_1,c_2,c_3];
126 lambda_1 = 104;
127 lambda_2 = 1.28;
128 lambda_3 = 54.5;
129 lambda_vec = [lambda_1,lambda_2,lambda_3];
130 rho_1 = 2.50 * 10^3;
131 rho_2 = 1.65 * 10^3;
132 rho_3 = 7.50 * 10^3;
133 rho_vec = [rho_1,rho_2,rho_3];
134 %=========================================
135 % Heat coefficients
136 %=========================================
137 alpha_k = 10^4;
138 alpha_tilde = 10;
139 alpha_bar = 40;
140 alpha = 41;
141 %=========================================
142 % Temperatures
143 %=========================================
144 u_L = 617 + index * 273.15;
145 u_S = 575 + index * 273.15;
146 T_env = 20 + index * 273.15;
147 T_f = 680 + index * 273.15;
148 C_0 = 0.07; C_E = 0.116; C_B = 0.0165;
149 psi_E = 0.26;
```

```
150 L = 3.69 * 10^5;
151 %=========================================
152 % Initial delta
153 %=========================================
154 delta_old = 0.06;
155 %=========================================
156 % Other parameters
157 %=========================================
158 %lambda_M = 0.0257;
159 lambda_M = 0.0448;
160 delta_M = 0.00015;
161 delta_0 = 0.006;
162 %=========================================
163 % Thermo pair - coordinates
164 %=========================================
165 rCoord = 0;
166 zCoord = 0.0565;
167 %=========================================
168 % At which index of z the value
169 % zCoord is reached
170 %=========================================
171 eps_z = 1.0e-06;
172 index_thermo = find(abs(z-zCoord)<eps_z);
173 if(isempty(index_thermo))
174     eps_z = 5 * eps_z;
175     while(isempty(index_thermo))
176         index_thermo = find(abs(z-zCoord)<eps_z);
177         eps_z = 5 * eps_z;
178     end
179 end
180 IndPair = index_thermo(1);
181 z(IndPair) = zCoord;
182 hz(IndPair) = z(IndPair)-z(IndPair-1);
183 hz(IndPair+1) = z(IndPair+1)-z(IndPair);
184 %=========================================
185 % Approximate solution
186 %=========================================
187 u = zeros(N,M,T);
188 %=========================================
189 % Initial condition
190 %=========================================
191 for i = 1 : N
192     for j = 1 : M
193         if(i>=0 && i<=Q && j>=P+1 && j<=M)
194             u(i,j,1) = T_f;
195         else
196             u(i,j,1) = T_env;
197         end
198     end
```

```
199 end
200 %=====================================
201 % Locally 1D
202 %=====================================
203 for n = 1 : T-1
204     sol = zeros(N,M);
205     %=====================================
206     % 1D for r
207     %=====================================
208     for j = 1 : M
209         adiag = zeros(1,N);
210         bdiag = zeros(1,N);
211         cdiag = zeros(1,N);
212         ddiag = zeros(1,N);
213         %=====================================
214         % i = 1
215         %=====================================
216         c = ...
217             coefficient_c_r(1,j,P,Q,r,hr,u(:,j,n),...
218             u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hrLine(1)
219         rho = rho_val(1,j,P,Q,rho_vec);
220         lambda = lambda_val(1,j,P,Q,lambda_vec);
221         adiag(1) = 0;
222         bdiag(1) = c * rho / tau + ...
223             (r(1)+hr(2)/2) * lambda / ...
224             (r(1) * hrLine(1) * hr(2));
225         cdiag(1) = -(r(1)+hr(2)/2) * lambda / ...
226             (r(1) * hrLine(1) * hr(2));
227         phi = 0;
228         ddiag(1) = c * rho / tau * u(1,j,n) + phi;
229         %=====================================
230         % i = N
231         %=====================================
232         c = ...
233             coefficient_c_r(N,j,P,Q,r,hr,u(:,j,n),...
234             u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hrLine(N)
235         rho = rho_val(N,j,P,Q,rho_vec);
236         lambda = lambda_val(N,j,P,Q,lambda_vec);
237         adiag(N) = -(r(N)-hr(N)/2) * lambda / ...
238             (r(N) * hrLine(N) * hr(N));
239         bdiag(N) = c * rho / tau + alpha / (hrLine(N)) + ...
240             (r(N)-hr(N)/2) * lambda / ...
241             (r(N) * hrLine(N) * hr(N));
242         cdiag(N) = 0;
243         phi = 0;
244         ddiag(N) = c * rho / tau * u(N,j,n) + ...
245             alpha * T_env / (hrLine(N)) + phi;
246         %=====================================
247         % i = 2 : N-1
```

```
248         %=========================================
249         for  ii = 2  : N-1
250              c = ...
251                   coefficient_c_r(ii,j,P,Q,r,hr,u(:,j,n),...
252                   u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hrLine(ii)
253              rho = rho_val(ii,j,P,Q,rho_vec);
254              lambda = lambda_val(ii,j,P,Q,lambda_vec);
255              if(ii==Q)
256                   if(j<=P)
257                        alpha_k_1 = +10^6;
258                   elseif(j>=P+1)
259                      alpha_k_1 = ...
260                           alpha_k_1_val_u(r,R_k,u,IndPair,n,...
261                                u_S,lambda_M,delta_M,delta_0);
262                   end
263                   x = 0;
264                   adiag(Q) = -(r(Q)-hr(Q)/2) * lambda / ...
265                        (r(Q) * hrLine(Q) * hr(Q));
266                   bdiag(Q) = c * rho / tau + ...
267                        (r(Q)-hr(Q)/2) * lambda / ...
268                        (r(Q) * hrLine(Q) * hr(Q)) + ...
269                        alpha_k_1 / (hrLine(Q));
270                   cdiag(Q) = -alpha_k_1 / (hrLine(Q));
271                   phi = 0;
272                   ddiag(Q) = c * rho / tau * u(Q,j,n) - ...
273                        alpha_k_1 * x / (hrLine(Q)) + phi;
274              elseif(ii==Q+1)
275                   if(j<=P)
276                        alpha_k_1 = +10^6;
277                   elseif(j>=P+1)
278                      alpha_k_1 = ...
279                           alpha_k_1_val_u(r,R_k,u,IndPair,n,...
280                                u_S,lambda_M,delta_M,delta_0);
281                   end
282                   x = 0;
283                   adiag(Q+1) = -alpha_k_1 / (hrLine(Q+1));
284                   bdiag(Q+1) = c * rho / tau + ...
285                        alpha_k_1 / (hrLine(Q+1)) + ...
286                        (r(Q+1)+hr(Q+2)/2) * lambda / ...
287                        (r(Q+1) * hrLine(Q+1) * hr(Q+2));
288                   cdiag(Q+1) = -(r(Q+1)+hr(Q+2)/2) * lambda / ...
289                        (r(Q+1) * hrLine(Q+1) * hr(Q+2));
290                   phi = 0;
291                   ddiag(Q+1) = c * rho / tau * u(Q+1,j,n) + ...
292                        alpha_k_1 * x / (hrLine(Q+1)) + phi;
293              else
294                   adiag(ii) = -(r(ii)-hr(ii)/2) * lambda / ...
295                        (r(ii) * hrLine(ii) * hr(ii));
296                   bdiag(ii) = c * rho / tau + ...
```

```
297                     (r(ii)-hr(ii)/2) * lambda / ...
298                     (r(ii) * hrLine(ii) * hr(ii)) + ...
299                     (r(ii)+hr(ii+1)/2) * lambda / ...
300                     (r(ii) * hrLine(ii) * hr(ii+1));
301                 cdiag(ii) = -(r(ii)+hr(ii+1)/2) * lambda / ...
302                     (r(ii) * hrLine(ii) * hr(ii+1));
303                 phi = 0;
304                 ddiag(ii) = c * rho / tau * u(ii,j,n) + phi;
305             end
306         end
307         %=========================================
308         % Matrix A
309         %=========================================
310         A = zeros(N,N);
311         for i0 = 1 : N
312             for j0 = 1 : N
313                 if(i0==j0+1)
314                     A(i0,j0) = adiag(i0);
315                 elseif(j0==i0+1)
316                     A(i0,j0) = cdiag(i0);
317                 elseif(i0==j0)
318                     A(i0,j0) = bdiag(i0);
319                 end
320             end
321         end
322         %=========================================
323         % Progonka for u(n+1/2)
324         %=========================================
325         sol(:,j) = Progon(A,ddiag);
326     end
327     %=========================================
328     % 1D for z
329     %=========================================
330     for i = 1 : N
331         adiag = zeros(1,M);
332         bdiag = zeros(1,M);
333         cdiag = zeros(1,M);
334         ddiag = zeros(1,M);
335         %=========================================
336         % j = 1
337         %=========================================
338         c = ...
339             coefficient_c_z(i,1,P,Q,z,hz,sol(i,:),...
340             u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hzLine
341         rho = rho_val(i,1,P,Q,rho_vec);
342         lambda = lambda_val(i,1,P,Q,lambda_vec);
343         adiag(1) = 0;
344         bdiag(1) = c * rho / tau + ...
345             alpha_tilde / (hzLine(1)) + ...
```

```
346             lambda / (hzLine(1) * hz(2));
347         cdiag(1) = -lambda / (hzLine(1) * hz(2));
348         phi = 0;
349         ddiag(1) = c * rho / tau * sol(i,1) + ...
350             alpha_tilde * T_env / (hzLine(1)) + phi;
351         %=========================================
352         % j = M
353         %=========================================
354         c = ...
355             coefficient_c_z(i,M,P,Q,z,hz,sol(i,:),...
356             u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hzLine
357         rho = rho_val(i,M,P,Q,rho_vec);
358         lambda = lambda_val(i,M,P,Q,lambda_vec);
359         adiag(M) = -lambda / (hzLine(M) * hz(M));
360         bdiag(M) = c * rho / tau + alpha_bar / (hzLine(M))+ ...
361             lambda / (hzLine(M) * hz(M));
362         cdiag(M) = 0;
363         phi = 0;
364         ddiag(M) = c * rho / tau * sol(i,M) + ...
365             alpha_bar * T_env / (hzLine(M)) + phi;
366         %=========================================
367         % j = 2 : M-1
368         %=========================================
369         for jj = 2 : M-1
370             c = ...
371                 coefficient_c_z(i,jj,P,Q,z,hz,sol(i,:),...
372                 u_L,u_S,c_1,c_2,c_3,L,psi_E,delta_old); % hzLine
373             rho = rho_val(i,jj,P,Q,rho_vec);
374             lambda = lambda_val(i,jj,P,Q,lambda_vec);
375             if(jj==P)
376                 adiag(P) = -lambda / (hzLine(P) * hz(P));
377                 bdiag(P) = c * rho / tau + ...
378                     alpha_k / (hzLine(P)) + ...
379                     lambda / (hzLine(P) * hz(P));
380                 cdiag(P) = -alpha_k / (hzLine(P));
381                 phi = 0;
382                 ddiag(P) = c * rho / tau * sol(i,P) + phi;
383             elseif(jj==P+1)
384                 adiag(P+1) = -alpha_k / (hzLine(P+1));
385                 bdiag(P+1) = c * rho / tau + ...
386                     alpha_k / (hzLine(P+1)) + ...
387                     lambda / (hzLine(P+1) * hz(P+2));
388                 cdiag(P+1) = -lambda / (hzLine(P+1) * hz(P+2));
389                 phi = 0;
390                 ddiag(P+1) = c * rho / tau * sol(i,P+1) + phi;
391             else
392                 adiag(jj) = -lambda / (hzLine(jj) * hz(jj));
393                 bdiag(jj) = c * rho / tau + ...
394                     lambda / (hzLine(jj) * hz(jj)) + ...
```

```
395                     lambda / (hzLine(jj) * hz(jj+1));
396                 cdiag(jj) = -lambda / (hzLine(jj) * hz(jj+1));
397                 phi = 0;
398                 ddiag(jj) = c * rho / tau * sol(i,jj) + phi;
399             end
400         end
401         %=========================================
402         % Matrix A
403         %=========================================
404         A = zeros(M,M);
405         for i0 = 1 : M
406             for j0 = 1 : M
407                 if(i0==j0+1)
408                     A(i0,j0) = adiag(i0);
409                 elseif(j0==i0+1)
410                     A(i0,j0) = cdiag(i0);
411                 elseif(i0==j0)
412                     A(i0,j0) = bdiag(i0);
413                 end
414             end
415         end
416         %=========================================
417         % Progonka for u(n+1)
418         %=========================================
419         u(i,:,n+1) = Progon(A,ddiag);
420     end
421 end
422 %=========================================
423 % Collecting data from the solution
424 %=========================================
425 xdata = t(1:1:T);
426 ydata = 0;
427 for idx = 1 : 1 : T
428     ydata = [ydata,u(1,IndPair,idx)];
429 end
430 ydata(1) = [];
431 %=========================================
432 % Load data from file
433 %=========================================
434 [fname,pname] = uigetfile('cyl_data.txt');
435 fullname = strcat(pname,fname);
436 Dat = load(fullname);
437 Xdata = Dat(:,1); Ydata = Dat(:,2);
438 for n = 1 : T
439     Max_exp_err(n) = abs(u(1,IndPair,n)-(Ydata(n)));
440 end
441 [abs_err,t_max_ind] = max(Max_exp_err(50:end));
442 t_max = t(t_max_ind+49);
443 res_norm = norm(Max_exp_err(50:end));
```

```
444 res_norm = sqrt(res_norm);
445 rel_err = abs_err ./ ...
446           max(max(max(Ydata(50:end))));
447 rel_err = rel_err * 100;
448 display(['Maximal difference: ', ...
449             num2str(abs_err), ...
450             ' for t = ', ...
451             num2str(t_max)])
452 display(['Square root of the norm: ', ...
453             num2str(res_norm)])
454 display(['Relative error: ', ...
455             num2str(rel_err),' %'])
456 %=========================================
457 % Plot
458 %=========================================
459 %figure(1)
460 plot(xdata,ydata-0*273.15,'m:','LineWidth',3)
461 hold on, grid on
462 plot([Xdata(1:100:end),Ydata(1:100:end), ...
463         'ko','LineWidth',2])
464 set(gca,'FontSize',12)
465 legend('\bf{Numerical solution}', ...
466     '\bf{Experiment}')
467 title(['\bf{Fractional \psi: u_L =  }',num2str(u_L), ...
468     '\bf{, u_S = }',num2str(u_S), ...
469     '\bf{, \alpha = }',num2str(alpha), ...
470     '\bf{, \alpha bar = }',num2str(alpha_bar), ...
471     '\bf{, \alpha tilde = }',num2str(alpha_tilde), ...
472     '\bf{, relative error = }',num2str(rel_err),'%'])
473 xlabel('\bf{t}')
474 ylabel('\bf{u(t)}')
475 axis([20,743.6,0,720])
476 axis normal
477 %=========================================
478 % Elapsed time
479 %=========================================
480 time = toc;
481 display(['Elapsed time: ',num2str(time)])
```

IX. Callback-функция, реализираща бутона START в средата на приложението *PDE Solve*

```
1 function startBut_Callback(hObject, eventdata, handles)
2 % hObject    handle to startBut (see GCBO)
3 % eventdata  reserved - to be defined in a future version of MATLAB
4 % handles    structure with handles and user data (see GUIDATA)
5
6 % Hint: get(hObject,'Value') returns toggle state of startBut
7     %=========================================
8     % Extracting values from the editable fileds
```

```
9       %=========================================================
10      u0s = get(handles.u0_Dyn,'String');
11      u0 = str2num(u0s);
12
13      hs = get(handles.hDyn,'String');
14      h = str2num(hs);
15
16      Hs = get(handles.HDyn,'String');
17      H = str2num(Hs);
18
19      Rks = get(handles.RkDyn,'String');
20      R_k = str2num(Rks);
21
22      Rs = get(handles.RDyn,'String');
23      R = str2num(Rs);
24
25      t0s = get(handles.tDyn,'String');
26      t0 = str2num(t0s);
27
28      Ts = get(handles.TDyn,'String');
29      tend = str2num(Ts);
30
31      c1s = get(handles.c1_Dyn,'String');
32      c_1 = str2num(c1s);
33
34      c2s = get(handles.c2_Dyn,'String');
35      c_2 = str2num(c2s);
36
37      c3s = get(handles.c3_Dyn,'String');
38      c_3 = str2num(c3s);
39
40      lam1s = get(handles.lam1_Dyn,'String');
41      lambda_1 = str2num(lam1s);
42
43      lam2s = get(handles.lam2_Dyn,'String');
44      lambda_2 = str2num(lam2s);
45
46      lam3s = get(handles.lam3_Dyn,'String');
47      lambda_3 = str2num(lam3s);
48
49      rho1s = get(handles.rho1_Dyn,'String');
50      rho_1 = str2num(rho1s);
51
52      rho2s = get(handles.rho2_Dyn,'String');
53      rho_2 = str2num(rho2s);
54
55      rho3s = get(handles.rho3_Dyn,'String');
56      rho_3 = str2num(rho3s);
57
```

```
58      al_tilde_s = get(handles.al_tilde_Dyn,'String');
59      alpha_tilde = str2num(al_tilde_s);
60
61      al_k_s = get(handles.al_k_Dyn,'String');
62      alpha_k = str2num(al_k_s);
63
64      al_s = get(handles.al_Dyn,'String');
65      alpha = str2num(al_s);
66
67      al_bar_s = get(handles.al_bar_Dyn,'String');
68      alpha_bar = str2num(al_bar_s);
69
70      Ls = get(handles.LDyn,'String');
71      L = str2num(Ls);
72
73      psi_e_s = get(handles.psi_e_Dyn,'String');
74      psi_E = str2num(psi_e_s);
75
76      cl = get(handles.cL_Dyn,'String');
77      c_L = str2num(cl);
78
79      cs = get(handles.cL_Dyn,'String');
80      c_S = str2num(cs);
81
82      ul = get(handles.uL_Dyn,'String');
83      u_L = str2num(ul);
84
85      us = get(handles.uS_Dyn,'String');
86      u_S = str2num(us);
87
88      rs = get(handles.rDyn,'String');
89      rCoord = str2num(rs);
90
91      zs = get(handles.zDyn,'String');
92      zCoord = str2num(zs);
93
94      [handles.r,handles.hr,handles.z,...
95       handles.hz,handles.t,handles.tau,...
96       handles.u,handles.P,handles.Q,...
97       handles.IndPair,handles.time] = ...
98                find_sol(R_k,R,h,H,t0,tend, ...
99                c_1,c_2,c_3,rho_1,rho_2,rho_3, ...
100               lambda_1,lambda_2,lambda_3, ...
101               c_L,c_S,u_L,u_S,L,psi_E, ...
102               alpha_tilde,alpha_k, ...
103     alpha,alpha_bar,rCoord,zCoord,u0);
104
105     helpdlg(['Numerical solution was found! ', ...
106         'Time elapsed: ', ...
```

```
107         num2str(handles.time)], ...
108         'Notification');
109
110     T = length(handles.t);
111     %=================================
112     % Collecting data from the solution
113     %=================================
114     handles.xdata = handles.t(1:1:T);
115     handles.ydata = 0;
116     for idx = 1 : 1 : T
117         handles.ydata = ...
118             [handles.ydata, ...
119              handles.u(1,handles.IndPair, ...
120              idx)];
121     end
122     handles.ydata(1) = [];
123     %=================================
124     % Plot the solution
125     %=================================
126     axes(handles.axMain)
127     p_line = plot(handles.xdata,handles.ydata, ...
128                   'b:','LineWidth',3);
129     set(handles.axMain,'FontSize',12);
130     xlabel('$$ t, $$ [s]','interpreter','latex');
131     ylabel('$$ u(t), \, \, [^\circ C] $$', ...
132            'interpreter','latex');
133     legend('\bf{Model data}');
134     handles.Line = p_line;
135     guidata(gcbo,handles);
136
137     set(handles.dataBut,'Enable','on');
138     set(handles.clearBut,'Enable','on');
139     set(hObject,'Enable','off');
140
141     set(handles.save_menu,'Enable','on');
142     set(handles.save_menu_item,'Enable','on');
143
144     set(handles.plot_menu,'Enable','on');
145     set(handles.plot_menu_item_1,'Enable','on');
146     set(handles.plot_menu_item_2,'Enable','on');
147     set(handles.plot_menu_item_3,'Enable','on');
148     set(handles.plot_menu_item_4,'Enable','on');
149
150     set(handles.plot_menu_item_time,'Enable','on');
151
152     set(handles.interp_menu,'Enable','on');
153     set(handles.bilin_interp,'Enable','on');
154     set(handles.eval_sol,'Enable','on');
155
```

```
156 end
```

X. *Callback*-функция, реализираща билинейна интерполация в средата на приложението *PDE Solve*

```
1  function bilin_interp_Callback(hObject, eventdata, handles)
2  % hObject    handle to bilin_interp (see GCBO)
3  % eventdata  reserved - to be defined in a future version of MATLAB
4  % handles    structure with handles and user data (see GUIDATA)
5      opts.Interpreter = 'tex';
6      x_dlg = inputdlg({'Enter r_{0}: ', ...
7                        'Enter z_0: ', ...
8                        'Enter t_0: '}, ...
9                        'Bilinear interpolation', ...
10                       [1,20; 1,20; 1,20], ...
11                       {'0.01','0.05','60'},opts);
12     r_val = x_dlg{1};
13     z_val = x_dlg{2};
14     t_val = x_dlg{3};
15     if(isempty(r_val) || isempty(z_val) || isempty(t_val) || ...
16             (isempty(r_val) && isempty(z_val)) || ...
17             (isempty(z_val) && isempty(t_val)) || ...
18             (isempty(r_val) && isempty(t_val)) || ...
19             (isempty(r_val) && isempty(z_val) && isempty(t_val)))
20         errordlg(['At least one of the values required is empty!', ...
21             'Try again with correct numerical values!'],'Error');
22     elseif(~isempty(r_val) && ~isempty(r_val) && ~isempty(t_val))
23         r_val = str2num(r_val);
24         z_val = str2num(z_val);
25         t_val = str2num(t_val);
26         if(isempty(r_val)) || isempty(z_val) || isempty(t_val) || ...
27                 (isempty(r_val) && isempty(z_val) || ...
28                 (isempty(z_val) && isempty(t_val)) || ...
29                 (isempty(r_val) && isempty(t_val)) || ...
30                 (isempty(r_val) && isempty(z_val) && isempty(t_val)))
31             errordlg(['At least one of the values required is not', ...
32                 ' numeric. Try again with correct numerical values!'],
    ...
33                 'Error');
34         elseif(~isempty(r_val) && ~isempty(z_val) && ~isempty(t_val))
35             if(handles.r(1) <= r_val && r_val <= handles.r(end) && ...
36                handles.z(1) <= z_val && z_val <= handles.z(end) && ...
37                handles.t(1) <= t_val && t_val <= handles.t(end))
38                idx_r = [1,1];
39                idx_z = [1,1];
40                idx_t = [1,1];
41                for i = 1 : length(handles.r)-1
42                     if(handles.r(i)<r_val && r_val<handles.r(i+1))
43                         idx_r = [i,i+1];
44                     end
```

```
                end
                for j = 1 : length(handles.z)-1
                    if(handles.z(j)<z_val && z_val<handles.z(j+1))
                        idx_z = [j,j+1];
                    end
                end
                for n = 1 : length(handles.t)-1
                    if(handles.t(n)<t_val && t_val<handles.t(n+1))
                        idx_t = [n,n+1];
                    end
                end
                idx_t = round(mean(idx_t));
                Q(1,1) = handles.u(idx_r(1),idx_z(1),idx_t);
                Q(1,2) = handles.u(idx_r(1),idx_z(2),idx_t);
                Q(2,1) = handles.u(idx_r(2),idx_z(1),idx_t);
                Q(2,2) = handles.u(idx_r(2),idx_z(2),idx_t);
                X = 1 / ((handles.r(idx_r(2)) - ...
                          handles.r(idx_r(1))) * ...
                    (handles.z(idx_z(2)) - handles.z(idx_z(1))));
                A = [handles.r(idx_r(2)) - r_val, ...
                     r_val - handles.r(idx_r(1))];
                C = [handles.z(idx_z(2)) - z_val; ...
                     z_val - handles.z(idx_z(1))];
                appr_u = X * A * Q * C;
                msgbox(['The temperature at the point r = ', ...
                        num2str(r_val),', z = ',num2str(z_val),...
                        ', t = ',num2str(t_val),' is u = ', ...
                        num2str(appr_u),'.'],'Evaluation','warn');
            else
                errordlg(['At least one of the values required is', ...
                        ' out of bounds. Try again with correct', ...
                        ' numerical values in the', ...
                        ' respective bounds!'],'Error');
            end
        end
    end
end
```

XI. *Callback*-функция, реализираща пресмятането на приближеното решение в избрани от потребителя точки в средата на приложението *PDE Solve*

```
function eval_sol_Callback(hObject, eventdata, handles)
% hObject    handle to eval_sol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

    list = {'direction r','direction z'};
    str = {'Choose spatial direction', ...
           'to perform interpolation:'};
```

```
[indx,tf] = listdlg('PromptString',str, ...
                    'ListString',list, ...
                    'SelectionMode','single');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INTERPOLATION FOR R
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if(indx==1)
    % we've chosen r-direction
    % choice of z and t - fixed
    prompt = {'Enter a fixed value of z: ', ...
        'Enter a fixed valued of t: '};
    title = 'Input';
    dims = [1,35];
    definput = {'0.05','100'};
    zt_data = inputdlg(prompt,title,dims,definput);
    zdata = zt_data{1};
    tdata = zt_data{2};
    %%%%%%%%%%%%%%%%%%%%%
    % check if zdata and tdata are empty
    if(isempty(zdata) || isempty(tdata) || ...
      (isempty(zdata) && isempty(tdata)))
        errordlg(['At least one of the values required ', ...
                  'is empty! Try again ',...
                  'with correct numerical values!'], ...
                  'Error');
    % if zdata and tdata are not empty
    elseif(~isempty(zdata) && ~isempty(tdata))
        zdata = str2num(zdata);
        tdata = str2num(tdata);
        if(isempty(zdata) || isempty(tdata) || ...
          (isempty(zdata) && isempty(tdata)))
            errordlg(['At least one of the values required ', ...
                      'is not', ...
                      ' numeric. Try again with correct ', ...
                      'numerical values!'], ...
                      'Error');
        elseif(~isempty(zdata) && ~isempty(tdata))
            % check for out of bounds
            % they're in the bounds
            if(handles.z(1)<=min(zdata) && ...
                  min(zdata)<=handles.z(end) && ...
               handles.z(1)<=max(zdata) && ...
                  max(zdata)<=handles.z(end) && ...
               handles.t(1)<=min(tdata) && ...
                  min(tdata)<=handles.t(end) && ...
               handles.t(1)<=max(tdata) && ...
                  max(tdata)<=handles.t(end))
               % interpolation for r!
               % enter the vector for r to interpolate
```

```
58                  r_data = inputdlg('Enter discrete values for r: ');
59                  rr = r_data{1}; pp = str2num(rr);
60                  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61                  if(isempty(rr) || (isnumeric(str2num(rr)) && ...
62                                     strcmp(num2str(pp),'')) || ...
63                                    (isnumeric(str2num(rr)) && ...
64                                     ~strcmp(num2str(pp),'') && ...
65                          (min(str2num(rr)) < handles.r(1) || ...
66                          max(str2num(rr)) > handles.r(end))))
67                      if(isempty(rr))
68                          %display('1')
69                          errordlg(['The input argument must not ',
    ...
70                                    'be empty. ', ...
71                                    'Try again with correct ', ...
72                                    'numerical values!'], ...
73                                    'Error')
74                      elseif(isnumeric(str2num(rr)) && ...
75                             strcmp(num2str(pp),''))
76                          %display('2')
77                          errordlg(['The input arument must be ', ...
78                                    'a number or a numerical ', ...
79                                    'vector. Try again with ', ...
80                                    'correct numerical values!'], ...
81                                    'Error')
82                      elseif(isnumeric(str2num(rr)) && ...
83                             ~strcmp(num2str(pp),'') && ...
84                             (min(str2num(rr)) < handles.r(1) || ...
85                             max(str2num(rr)) > handles.r(end)))
86                          %display('3')
87                          errordlg(['At least one of the ', ...
88                                    'coordinates ', ...
89                                    'of the vector is ', ...
90                                    'out of bounds. Try again ', ...
91                                    'with ', ...
92                                    'correct values in the ', ...
93                                    'respective bounds.'], ...
94                                    'Error')
95                      end
96                  else
97                      % correct values for rr
98                      rr = str2num(rr);
99                      % check for monotnically increasing vector
100                     if(isIncreasing(rr))
101                         r_before = rr(find(rr <= ...
102                                        handles.r(handles.Q)));
103                         r_after = rr(find(rr >= ...
104                                       handles.r(handles.Q)));
105                         if(isempty(r_before))
```

```
106                         r_before = handles.r(handles.Q-1);
107                     end
108                     if(isempty(r_after))
109                         r_after = handles.r(handles.Q+2);
110                     end
111                     idx_z = find(abs(handles.z-zdata)<0.01);
112                     idx_z = idx_z(1);
113                     idx_t = find(abs(handles.t-tdata)<1);
114                     idx_t = idx_t(1);
115                     u_1 = interp1(handles.r(1:handles.Q), ...
116                                   handles.u(1:handles.Q, ...
117                                             idx_z,idx_t), ...
118                                   r_before,'pchip');
119                     u_2 = interp1(handles.r(handles.Q+1:end),
    ...
120                                   handles.u(handles.Q+1:end, ...
121                                             idx_z,idx_t), ...
122                                   r_after,'pchip');
123                     uu = [u_1,u_2];
124                     for ii = 1 : length(rr)
125                        rs = rr(ii);
126                        us = uu(ii);
127                        for i = 1 : length(handles.r)-1
128                            if(handles.r(i) < rs && ...
129                               rs < handles.r(i+1))
130                               % r
131                               rn = [];
132                               rn = handles.r(1:i);
133                               rn = [rn,rs];
134                               rn = [rn, ...
135                                     handles.r(i+1:end)];
136                               r_new = rn; % the finer grid
137                               % for r
138                               % u
139                               un = [];
140                               un = handles.u(1:i,idx_z, ...
141                                              idx_t)';
142                               un = [un,us];
143                               un = [un,handles.u(i+1:end, ...
144                                                  idx_z, ...
145                                                  idx_t)'];
146                               u_new = un; % the finer grid
147                               % for u
148                               % plot
149                               axis(handles.axMain)
150                               clear axes; cla;
151                               set(handles.axMain, ...
152                                   'Fontname','Times')
153                               axis([min(r_new), ...
```

```
154                                                 max(r_new) + 0.1 * ...
155                                                 handles.r(handles.Q),...
156                                                 min(u_new)-5,...
157                                                 max(u_new)+5])
158                                             plot(r_new,u_new, ...
159                                                  'b:','LineWidth',3)
160                                             hold on, grid on
161                                             plot(r_before,u_1,'ko', ...
162                                                  'LineWidth',2)
163                                             plot(r_after,u_2,'ko', ...
164                                                  'LineWidth',2)
165                                             xlabel('\bf{Radius r}')
166                                             ylabel('\bf{Temperature u}')
167                                             legend('\it{Interpolated u(r)}
    ', ...
168                                                    '\it{Nodes}')
169                                         end % of if
170
171                                     end % of the inner for
172                                 end % of the outer for
173                             else % r is not increasing or consists
174                                  % repeating elements
175                                 errordlg('The vector is not ', ...
176                                          'monotonically increasing!', ...
177                                          'Error');
178                             end
179
180                         end
181                         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
182                     else
183                         errordlg(['At least one of the values required is',
    ...
184                          ' out of bounds. Try again with correct ', ...
185                          'numerical values in the', ...
186                          ' respective bounds!'], ...
187                          'Error');
188                     end
189                 end
190             end
191         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
192         % INTERPOLATION FOR Z
193         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
194         elseif(indx==2)
195             % we've chosen z-direction
196             % choice of r and t - fixed
197             prompt = {'Enter a fixed value of r: ', ...
198                 'Enter a fixed valued of t: '};
199             title = 'Input';
200             dims = [1,35];
```

```
201         definput = {'0.001','100'};
202         rt_data = inputdlg(prompt,title,dims,definput);
203         rdata = rt_data{1};
204         tdata = rt_data{2};
205         %%%%%%%%%%%%%%%%%%%%%
206         % check if rdata and tdata are empty
207         if(isempty(rdata) || isempty(tdata) || ...
208                              (isempty(rdata) && ...
209                               isempty(tdata)))
210             errordlg(['At least one of the values ', ...
211                       'required is empty! Try again ',...
212                       'with correct numerical values!'], ...
213                       'Error');
214         % if rdata and tdata are not empty
215         elseif(~isempty(rdata) && ~isempty(tdata))
216             rdata = str2num(rdata);
217             tdata = str2num(tdata);
218             if(isempty(rdata) || isempty(tdata) || ...
219               (isempty(rdata) && isempty(tdata)))
220                 errordlg(['At least one of the values ', ...
221                           'required is not ', ...
222                           'numeric. Try again with correct ', ...
223                           'numerical values!'], ...
224                           'Error');
225             elseif(~isempty(rdata) && ~isempty(tdata))
226                 % check for out of bounds
227                 % they're in the bounds
228                 if(handles.r(1)<=min(rdata) && ...
229                      min(rdata)<=handles.r(end) && ...
230                    handles.r(1)<=max(rdata) && ...
231                      max(rdata)<=handles.r(end) && ...
232                    handles.t(1)<=min(tdata) && ...
233                      min(tdata)<=handles.t(end) && ...
234                    handles.t(1)<=max(tdata) && ...
235                      max(tdata)<=handles.t(end))
236                    % interpolation for z!
237                    % enter the vector for z to interpolate
238                    z_data = inputdlg('Enter discrete values for z: ');
239                    zz = z_data{1}; pp = str2num(zz);
240                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
241                    if(isempty(zz) || (isnumeric(str2num(zz)) && ...
242                       strcmp(num2str(pp),'')) || ...
243                                    (isnumeric(str2num(zz)) && ...
244                                    ~strcmp(num2str(pp),'') && ...
245                             (min(str2num(zz)) < handles.z(1) || ...
246                             max(str2num(zz)) > handles.z(end))))
247                       if(isempty(zz))
248                           errordlg(['The input argument must not ',
    ...
```

```
249                                     'be empty. ', ...
250                                     'Try again with correct ', ...
251                                     'numerical values!'], ...
252                                     'Error')
253                             elseif(isnumeric(str2num(zz)) && ...
254                                    strcmp(num2str(pp),''))
255                                 errordlg(['The input arument must be ', ...
256                                     'a number or a numerical ', ...
257                                     'vector. Try again with ', ...
258                                     'correct numerical values!'], ...
259                                     'Error')
260                             elseif(isnumeric(str2num(zz)) && ...
261                                    ~strcmp(num2str(pp),'') && ...
262                                     (min(str2num(zz)) < handles.z(1) || ...
263                                      max(str2num(zz)) > handles.z(end)))
264                                 errordlg(['At least one of the ', ...
265                                     'coordinates ', ...
266                                     'of the vector is ', ...
267                                     'out of bounds. Try again ', ...
268                                     'with ', ...
269                                     'correct values in the ', ...
270                                     'respective bounds.'], ...
271                                     'Error')
272                             end
273                         else
274                             % correct values for zz
275                             zz = str2num(zz);
276                             % check for monotnically increasing vector
277                             if(isIncreasing(zz))
278                                 z_before = zz(find(zz <= ...
279                                                 handles.z(handles.P)));
280                                 z_after = zz(find(zz >= ...
281                                                   handles.z(handles.P)));
282                                 if(isempty(z_before))
283                                     z_before = handles.z(handles.P-1);
284                                 end
285                                 if(isempty(z_after))
286                                     z_after = handles.z(handles.P+2);
287                                 end
288                                 idx_r = find(abs(handles.r-rdata)<0.01);
289                                 idx_r = idx_r(1);
290                                 idx_t = find(abs(handles.t-tdata)<1);
291                                 idx_t = idx_t(1);
292                                 u_1 = interp1(handles.z(1:handles.P), ...
293                                               handles.u(idx_r, ...
294                                               1:handles.P,idx_t), ...
295                                               z_before,'pchip');
296                                 u_2 = interp1(handles.z(handles.P+1:end),...
297                                               handles.u(idx_r, ...
```

```
298                                     handles.P+1:end,idx_t), ...
299                                     z_after,'pchip');
300                             uu = [u_1,u_2];
301                             for jj = 1 : length(zz)
302                                 zs = zz(jj);
303                                 us = uu(jj);
304                                 for j = 1 : length(handles.z)-1
305                                     if(handles.z(j) < zs && ...
306                                        zs < handles.z(j+1))
307                                         % z
308                                            zn = [];
309                                            zn = handles.z(1:j);
310                                            zn = [zn,zs];
311                                            zn = [zn, ...
312                                                  handles.z(j+1:end)];
313                                            z_new = zn; % the finer grid
314                                            % for z
315                                            % u
316                                            un = [];
317                                            un = handles.u(idx_r,1:j, ...
318                                                           idx_t);
319                                            un = [un,us];
320                                            un = [un, ...
321                                                  handles.u(idx_r, ...
322                                                  j+1:end, ...
323                                                  idx_t)];
324                                            u_new = un; % the finer grid
325                                            % for u
326                                            % plot
327                                            axis(handles.axMain)
328                                            clear axes; cla;
329                                            set(handles.axMain, ...
330                                                'Fontname','Times')
331                                            axis([min(z_new), ...
332                                                max(z_new) + 0.1 * ...
333                                                handles.z(handles.P),...
334                                                min(u_new)-5,...
335                                                max(u_new)+5])
336                                            plot(z_new,u_new, ...
337                                                 'm:','LineWidth',3)
338                                            hold on, grid on
339                                            plot(z_before,u_1, ...
340                                                 'ko','LineWidth',2)
341                                            plot(z_after,u_2, ...
342                                                 'ko','LineWidth',2)
343                                            xlabel('\bf{Height z}')
344                                            ylabel('\bf{Temperature u}')
345                                            legend('\it{Interpolated u(z)}
    ', ...
```

```
346                                                 '\it{Nodes}')
347                                     end % of if
348
349                                 end % of the inner for
350                             end % of the outer for
351                         else % z is not increasing or consists
352                              % repeating elements
353                             errordlg('The vector is not ', ...
354                                      'monotonically ', ...
355                                      'increasing!', ...
356                                      'Error');
357                         end
358
359                     end
360                     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
361                 else
362                     errordlg(['At least one of the values ', ...
363                               'required is ', ...
364                               'out of bounds. Try again with ', ...
365                               'correct numerical values in the ', ...
366                               'respective bounds!'], ...
367                               'Error');
368                 end
369             end
370         end
371     end
```