БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ
ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

Явор Иванов Вутов

ПАРАЛЕЛНИ ИТЕРАЦИОННИ МЕТОДИ ЗА НЕКОНФОРМНИ КРАЙНИ ЕЛЕМЕНТИ

Д И С Е Р Т А Ц И Я

за присъждане на
образователната и научна степен „Доктор“

Научна специалност: 01.01.13 – „Математическо моделиране и приложение на математиката“

Професионално направление: 4.5 – „Математика“

Научен ръководител: чл.-кор. Светозар Маргенов

София, 2015г.

Съдържание

Използвани означения и съкращения **5**

Увод **7**

1 Въведение **15**
1.1 Метод на крайните елементи 15
1.1.1 Описание на метода 15
1.1.2 Крайни елементи на Ранахер и Турек 18
1.2 Линейна теория на еластичността 20
1.3 Метод на спрегнатия градиент с преобуславяне 22
1.4 MIC(0) факторизация 25
1.5 Алгебричен мултигрид 26
1.6 Бележки относно програмната реализация 29

2 Паралелен MIC(0) преобусловител **31**
2.1 Постановка на задачата 31
2.2 Дискретизация 32
2.3 Конструкция на метода за преобуславяне 32
2.3.1 Идея на паралелния алгоритъм 32
2.3.2 Локално оптимизирани паралелни MIC(0) преобусловители 34
2.4 Локален анализ на числото на обусловеност 40
2.5 Паралелна реализация 44
2.5.1 Описание на алгоритъма 44
2.5.2 Програмна реализация 48
2.6 Оценки за паралелните времена 53
2.6.1 Изчислителна сложност 53
2.6.2 Времена 55
2.7 Числени експерименти 58
2.7.1 Сходимост 58
2.7.2 Паралелни времена 61

3 Паралелен метод за системата на Ламе **65**
3.1 Постановка на задачата 65
3.2 Преобуславяне чрез разделяне по преместванията 66
3.3 Паралелен MIC(0) преобусловител 67
3.4 Паралелен мултигрид преобусловител 69
3.5 Реализация 69

3.6 Оценки за паралелните времена 71
3.6.1 Изчислителна сложност 71
3.6.2 Времена 72
3.7 Числени експерименти 74
3.7.1 Сходимост 74
Сравнение с аналитично решение 74
Сходимост за задача със скок в коефициентите 75
3.7.2 Паралелни времена 77

4 Приложения **81**
4.1 Симулации на линейни еластични системи 82
4.1.1 Пилотни фундаменти 82
Числени експерименти 82
4.1.2 Костни микроструктури 86
Числени експерименти 87
4.2 Числена хомогенизация 90
4.2.1 Описание на метода 91
4.2.2 Главни направления на анизотропия 93
4.2.3 Сходимост на МСГП за полуопределени матрици 94
4.2.4 Алгоритмична реализация 95
4.2.5 Числени експерименти 95
Хомогенизация на трабекуларни тъкани 95
Хомогенизация на полимерни нанокомпозити 99
Сравнение с пакета GeoDict 101
4.2.6 Сравнение с аналитични оценки 103

Използвана литература **107**

Често използвани означения и съкращения

$\mathbb{R}$ — Множеството на реалните числа

α, γ — С малки букви от гръцката азбука са означени скаларни стойности или функции

A, C — С главни латински букви са означени матрици

$\mathbf{b}, \mathbf{x}$ — Векторите са означени с удебелени латински букви

$\mathbf{e}$ — Единичен вектор с подразбираща се размерност

$\mathbf{0}$ — Нулев вектор с подразбираща се размерност

$\begin{bmatrix} a_{1,1} & & \\ & a_{2,2} & \\ & & a_{3,3} \end{bmatrix}$ — Често при записа на матрици нулевите елементи са пропускани

МСГ — Метод на спрегнатия градиент

МСГП — Метод на спрегнатия градиент с преобуславяне

МКЕ — Метод на крайните елементи

МКР — Метод на крайните разлики

СЛАУ — Система линейни алгебрични уравнения

MIC(0) — Модифицирана непълна факторизация на Холецки от нулев ред

δ_{ij} — Символ на Кронекер: $\delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$

$l\left(\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\right)$ — Линейна обвивка на вектори или функции.

$diag(a_{1,1}, a_{2,2}, \ldots, a_{n,n})$ — Диагонална матрица с изброени елементи по главния диагонал

Увод

Актуалност и съвременно състояние на темата

Математическото моделиране е сложен интердисциплинарен процес, който има за цел изучаването, разбирането и описването на процесите и явленията в заобикалящият ни свят. То включва:

1. Събиране на познание чрез наблюдения и експерименти;
2. Определяне на имащите отношение към разглежданите процеси фактори и величини;
3. Изясняване на количествените връзки между тях, формулирайки физични и математични закони.

Веднъж изведени тези закони биват прилагани и при други ситуации. Когато математическите модели биват реализирани, като компютърни програми, извършващи симулации на различните физични процеси, говорим за *компютърно моделиране*. То ни позволява да извършваме нови, виртуални експерименти и да разбираме по-добре реалните такива. Например при една термосимулация с компютърен експеримент можем да получим топлинното поле във всяка една точка от изследваната област, дори в точки вътрешни за твърди тела. Компютърното моделиране е незаменим инструмент в редица случаи, сред които:

- Мащабите на експеримента, не позволяват провеждането му – например гравитационна симулация на галактика;
- Цената на оборудването и консумативите за извършване на реален експеримент е много голяма;
- Трябва висока точност – количествена, пространствена и времева – на симулираните величини.

Математическите модели на физичните процеси са обикновено интегрални или диференциални уравнения. В някои идеализирани частни случаи и при прости геометрии те могат да се решат аналитично. Това често не е възможно за реални задачи. Тогава се прибягва до тяхното числено решаване. Първа стъпка от численото решение е *дискретизацията*. Това е апроксимацията на непрекъснатата задача с дискретна такава. Основни методи за числено решаване на диференциални уравнения са методът на крайните разлики (МКР) и

методът на крайните елементи (МКЕ) [69]. При дискретизация на гранични задачи се получават системи линейни алгебрични уравнения (СЛАУ) в следния вид.

$$A\mathbf{x} = \mathbf{b}.$$

Матриците в тези системи са *разредени* и често с голяма размерност – достигаща милиони и милиарди неизвестни.

Дефиниция. *Разредени матрици наричаме такива матрици, при които броят на ненулевите елементи на всеки ред е ограничен с константа независеща от размера на матрицата.*

Методите за решаване на СЛАУ най-общо се делят на два типа: преки и итерационни [46, 88, 93, 94]. Преките методи дават точно решение (при точна аритметика), като броят на операциите е ограничен и предварително известен. Такива методи са

- Гаусовата елиминация – привеждане на матрицата на системата A в триъгълен вид;
- методът на квадратния корен, известен още като метод на Холецки;
- LU факторизация – преобразуване на матрицата A като произведение на долно- и горнотриъгълни такива;
- метод на вложените сечения [50].

При итерационните методи, решението се търси като последователност от приближения, като всяко следващо приближение се намира от предишните едно или повече приближения. Този процес се повтаря до достигането на определен критерий за сходимост. В Таблица 1 са дадени порядъците на броя на операциите и необходимата памет за различни преки и итерационни методи, като са вписани порядъците на броя на операциите и необходима оперативна памет. Както се вижда итерационните методи базирани на метода на спрегнатия градиент (МСГ), както и многонивовите и мултигрид методи имат явни предимства за този клас задачи пред преките, както по отношение на използваната оперативна памет, така и по броя на операциите им. Тяхната сходимост силно зависи от числото на обусловеност на матрицата A:

Дефиниция. *Число на обусловеност на матрицата A бележим с $\kappa(A)$, като*

$$\kappa(A) = ||A||.||A^{-1}||$$

за някоя норма $//.//$.

Забележка. Когато нормата $||.||$ е Евклидовата, а матрицата A е симетрична и положително определена, тогава числото на обусловеност може да бъде изразено чрез най-голямата λ_N и най-малката λ_1 собствени стойности на матрицата A:

$$\kappa(A) = ||A||_2.||A^{-1}||_2 = \frac{\lambda_N}{\lambda_1}.$$

Тогава наричмаме $\kappa(A)$ *спектрално число на обусловеност*. В тази работа под число на обусловеност навсякъде разбираме спектралното такова.

Таблица 1: Брой на операциите и необходима оперативна памет за различните методи за решаване СЛАУ със симетрична матрица с размери $N \times N$, получена при решаване на гранична елиптична задача от втори ред

Метод	Размерност	Операции	Памет
Гаусова елиминация	2D, 3D	$O(N^3)$	$O(N^2)$
Метод на Холецки	2D	$O(N^2)$	$O(N^{3/2})$
Метод на Холецки	3D	$O(N^{13/9})$	$O(N^{5/3})$
Метод на вложените сечения	2D	$O(N^{3/2})$	$O(N \log N)$
Метод на вложените сечения	3D	$O(N^2)$	$O(N^{4/3})$
МСГ	2D	$O(N^{3/2})$	$O(N)$
МСГ	3D	$O(N^{4/3})$	$O(N)$
МСГ с MIC(0) преобусловител	2D	$O(N^{5/4})$	$O(N)$
МСГ с MIC(0) преобусловител	3D	$O(N^{7/6})$	$O(N)$
МСГ с мултигрид преобусловител	2D,3D	$O(N)$	$O(N)$
МСГ с многонивов преобусловител	2D,3D	$O(N)$	$O(N)$

При метода на спрегнатия градиент броят на итерациите необходим за достигане на зададена точност е пропорционален на квадратен корен от числото на обусловеност $\kappa(A)$. Числото на обусловеност има отношение и към преките методи, където при решаване на системи с лошо обусловените матрици (а това са матриците с голямо число на обусловеност) се получават много големи грешки при закръгления (при работа с числа с крайна точност).

Сходимостта на метода на спрегнатия градиент може да се подобри чрез преобуславяне, което по същество е решаване на друга система с по-добре обусловена матрица B

$$B\mathbf{y} = \mathbf{c},$$

като търсеното от нас решение $\mathbf{x}$ може лесно (и бързо) да бъде намерено от полученото такова $\mathbf{y}$.

Времето за решаването на СЛАУ, след избиране на подходящ метод, може допълнително да бъде намалено, като се използва *паралелизация*. Това означава алгоритъмът да се изпълнява *едновременно* на повече от един процесор. Това може да доведе до ускорение на изпълнението на алгоритмите пропорционално на броя на процесорите. Още повече при съвременните компютърни системи, скоростта на процесорите с общо предназначение, към настоящия момент (2015 година) е спряла да се покачва и паралелизацията е единствения начин за по-бързото изпълнение на даден алгоритъм.

В настоящата работа разработваме паралелен алгоритъм за решаване на тримерни елиптични гранични задачи от втори ред по метода на спрегнатия градиент с преобусловител модифицирана непълна факторизация на Холецки от нулев ред (MIC(0)). Това е логическо продължение на работите на колегите Иван Георгиев, разработил алгоритми за преобуславяне на тримерни задачи, и на Гергана Бенчева, разработила подобен паралелен алгоритъм, но за двумерни задачи[11, 12].

Въпреки неоптималния си характер, показваме, че преобусловителя на ба-

зата на непълната факторизация може успешно да се конкурира с модерни, асимптотично оптимални мултигрид алгоритми.

Цели на дисертационния труд

Основните цели на изследванията в дисертацията са:

- Разработване и изследване на паралелен MIC(0) преобусловител за тримерни елиптични задачи, дискретизирани чрез неконформни крайни елементи на Ранахер–Турек.

- Разработване и изследване на паралелен блочен MIC(0) преобусловител от тип разделяне по преместванията за тримерната система на Ламе и приложението му за числено решаване на еластични задачи върху вокселни структури.

- Разработване и изследване на паралелен алгоритъм за числена хомогенизация на вокселни структури.

Методология на изследванията

При решаването на задачи с голяма размерност от основно значение са времето за изпълнение на програмата и ресурсите, които тя заема – оперативна памет, брой на използваните процесори. Тези величини имат директно отношение, както към цените на компютърните системи, така и към разходите за експлоатацията им (например потребявана електрическа енергия). Специално внимание е обърнато на минималното ползване на оперативна памет – това позволява решаването на по-големи задачи. Времето за изпълнение на програмите също има значение за използването им в практиката. Например, никой не би чакал дни за компютърна проверка на правописа.

От друга страна времето е пропорционално на броя на извършваните операции за даден алгоритъм. За минимизиране на този брой са необходими ефективни алгоритми. Основно средство за решаване на големи СЛАУ е МСГ. Броят на операциите за изпълнение на алгоритъма е пропорционален на корен квадратен на числото на обусловеност на матрицата $\kappa(A)$ и на броя на неизвестните N. В разработените в дисертацията преобусловители основна цел е създаването на преобусловител C намаляващ относителното число на обусловеност $\kappa\left(C^{-1}A\right)$.

Разработените в дисертацията алгоритми на базата на модифицирана непълна факторизация на Холецки имат брой на операциите $O(N^{7/6})$. Времената за изпълнението им са сравнени с мултигрид алгоритъм с оптимална изчислителна сложност $O(N)$.

За паралелните програми времето за изпълнение *силно* зависи и от паралелната ефективност. Тя е оценка за това каква част от процесорното време (на всички процесори) се използва от програмата. Разработените алгоритми имат оптимална асимптотична паралелна ефективност.

Съдържание на дисертацията

Дисертацията се състои от увод и четири глави.

В първа глава са представени общи сведения за методите и алгоритмите използвани в дисертацията. Това включва метода на крайните елементи и крайните елементи на Ранахер-Турек, общи понятия от линейната теория на еластичността, метода на спрегнатия градиент с преобуславяне за решаване на системи линейни алгебрични уравнения, непълната модифицирана факторизация на Холецки MIC(0), алгебричния мултигрид, както и някои бележки върху програмната реализация на разработените алгоритми.

Във втора глава е разработен паралелен алгоритъм за преобуславяне на елиптични задачи, дискретизирани чрез крайни елементи на Ранахер-Турек използвайки MIC(0) факторизация. Изведени са оценки за сходимостта, както и за паралелните времена на алгоритъма.

В трета глава е разработен паралелен алгоритъм за решаване на уравненията на Ламе върху вокселни структури, на базата на разделяне по премествания и използвайки алгебричен мултигрид или разработения в Глава 2 алгоритъм.

В заключителната четвърта глава са изследвани някои приложения на алгоритмите включващи компютърна симулация на напрегнатото и деформирано състояние на трабекуларна кост и пилотни фундаменти, като и числена хомогенизация за композитни материали.

Съществено място в Глави 2,3 и 4 е отделено и на числените експерименти. Те демонстрират сходимостта, ефективността и приложимостта на разработените методи.

В интернет могат да бъдат намерени текста на дисертацията в електронна форма, изходния код на разработените програми и инструменти, както и други материали по дисертацията [89].

Списък на публикациите по дисертацията

Резултати, включени в дисертацията, са публикувани в [2, 32, 53, 54, 56, 60–63, 84, 85]:

- Y. Vutov. Parallel incomplete factorization of 3D NC FEM elliptic systems. *Numerical Methods and Applications*, *LNCS* **4310**, 114–121. Springer-Verlag, 2007

- I. Lirkov and Y. Vutov. Comparative analysis of high performance solvers for 3D elliptic problems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 483–492, 2007

- P. Arbenz, S. Margenov, and Y. Vutov. Parallel MIC(0) preconditioning of 3D elliptic problems discretized by Rannacher–Turek finite elements. *Computers and Mathematics with Applications*, 55(10):2197–2211, 2008

- I. Georgiev, E. Ivanov, S. Margenov, and Y. Vutov. Numerical homogenization of epoxy-clay composite materials. *Numerical Methods and Applications*, *LNCS* **8962**, 130–137. Springer Berlin Heidelberg, 2015

- S. Margenov, S. Stoykov, and Y. Vutov. Numerical homogenization of heterogeneous anisotropic linear elastic materials. *Large-Scale Scientific Computing*, LNCS, 347–354. Springer Berlin Heidelberg, 2014

- S. Margenov and Y. Vutov. Parallel MIC(0) preconditioning for numerical upscaling of anisotropic linear elastic materials. *Large-Scale Scientific Computing*, *LNCS* **5910**, 805–812. Springer, 2010

- I. Lirkov, Y. Vutov, M. Paprzycki, and M. Ganzha. Parallel performance evaluation of MIC(0) preconditioning algorithm for voxel μFE simulation. *Parallel Processing and Applied Mathematics*, *LNCS* **6068**, 135–144. Springer, 2010

- S. Margenov and Y. Vutov. Parallel PCG algorithms for voxel FEM elasticity systems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 517–526, 2007

- Y. Vutov. Parallel DD-MIC(0) Preconditioning of Nonconforming Rotated Trilinear FEM Elasticity Systems. *Large-Scale Scientific Computing*, *LNCS* **4818**, 745–572. Springer-Verlag, 2008

- I. Lirkov, Y. Vutov, M. Ganzha, and M. Paprzycki. Comparative Analysis of High Performance Solvers for 3D Elasticity Problems. *Numerical Methods and Applications*, *LNCS* **5434**, 392–399. Springer-Verlag, 2009

- S. Margenov and Y. Vutov. Preconditioning of voxel FEM elliptic systems. *TASK Quarterly*, 11(1-2):117–128, 2007

Апробация на резултатите

Резултати от дисертацията са докладвани на семинари в бившия Институт по Паралелна Обработка на Информацията и на редица международни конференции, сред които

- Large Scale Scientific Computations, Созопол – 2007, 2009, 2011, 2013;

- International Conference on Parallel Processing and Applied Mathematics, Торун, Полша – 2011

- Enumath, Упсала, Швеция – 2009

- International Workshop on Parallel Matrix Algorithms and Applications PMAA Нюшател, Швейцария – 2008, Базел, Швейцария – 2010

- Numerical Methods and Applications, Боровец – 2006, 2010, 2014

Участие в научни проекти

Участвал съм в редица научни проекти, сред които:

- ДО02-115/2008 – Център за върхови научни постижения "Суперкомпютърни приложения";
- ДО02-147/2008 – Методи, алгоритми и софтуерни средства за задачи с голяма размерност и йерархични компютърни модели;
- ДФНИ И01/5 – Числени методи за свързани системи и компютърно моделиране в биомедицината и екологията;
- BG161PO003-1.1.06-0004-C0001 -– ОП „Конкурентоспособност” - Иновативни технологични решения за радиочестотна термоаблация;
- ДЦВП02/1 – Развитие на център за върхови научни постижения "Суперкомпютърни приложения";
- BIS-21++ – Bulgarian IST Centre of Competence in 21 Century;
- PRACE – Partnership for Advanced Computing in Europe.

Основни научни и научноприложни приноси

- Разработен е паралелен MIC(0) преобусловител за тримерни елиптични задачи, дискретизирани чрез неконформни крайни елементи на Ранахер–Турек. Изведени са оценки за скоростта на сходимост на метода и за паралелните времена.
- Разработен е паралелен блочен MIC(0) преобусловител от тип разделяне по преместванията за тримерната система на Ламе и приложението му за числено решаване на еластични задачи върху вокселни структури. Изведени са оценки за паралелните времена.
- Разработен е паралелен алгоритъм за числена хомогенизация на композитни материали.
- Програмно са реализирани разработените методи и алгоритми. Проведени са числени експерименти потвърждаващи качествата и ефективността на разработените алгоритми.
- Разработените програмни средства са приложени за важни класове задачи с голяма размерност от практиката.

Благодарности

Тази работа нямаше да бъде реалност без всеотдайната подкрепа на много хора:

- жена ми,

- приятелите ми,
- колегите ми,
- роднините ми,
- и разбира се, на научния ми ръководител.

Сърдечно ви благодаря!

Също така искам да благодаря и на програмата HPC-Europa, благодарение на която на два пъти посетих суперкомпютърния център „CINECA“, Италия, където свърших по-голямата част от работата по глави 2 и 3. Благодаря и на „Българският център за суперкомпютърни приложения“ за предоставеният ми достъп до компютъра Blue Gene/P.

Глава 1

Въведение

В тази глава са включени основни сведения за задачите, методите и алгоритмите засегнати и използвани в дисертацията. Това са: метода на крайните елементи и в частност неконформните крайни елементи на Ранахер и Турек; метода на спрегнатия градиент с преобуславяне; модифицираната непълна факторизация; алгебричния мултигрид, както и формулировка на задачи от линейната еластичност.

1.1 Метод на крайните елементи

При числено решаване на диференциални уравнения, непрекъсната задача се дискретизира, след което се решава получената система от линейни алгебрични уравнения. Един от съвременните подходи за дискретизация е методът на крайните елементи (МКЕ) [7, 16, 43, 95]. Името на метода произлиза от идеята решенията на диференциалните уравнения да се приближават върху елементи с краен размер, а не безкрайно малки, както при построенията в анализа. При МКЕ решението се търси като линейна комбинация от функции, чийто носител е ограничен върху малък брой (съседни) елементи. Най-често тези функции са на части полиномиални.

МКЕ е предпочитан метод в следните случаи:

- при решаването на задачи в области със сложна геометрия,
- при променящи се с времето области,
- при желана различна точност в различни части от областта.

1.1.1 Описание на метода

Нека разгледаме следната елиптична гранична задача:

$$-\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{1.1.1а}$$

$$u(\mathbf{x}) = 0, \qquad \mathbf{x} \in \Gamma_D, \tag{1.1.1б}$$

$$(a(\mathbf{x})\nabla u(\mathbf{x})) \cdot \mathbf{n} = 0, \qquad \mathbf{x} \in \Gamma_N, \tag{1.1.1в}$$

където Ω е (ограничена, едносвързана) област в $\mathbb{R}^3$, с граница $\partial\Omega$, разделена на две части – Γ_D, с ненулева мярка, където се налагат гранични условия на Дирихле, и Γ_N, където се налагат гранични условия на Нойман, $\Gamma_D \cap \Gamma_N = \emptyset$, $\partial\Omega = \Gamma_D \cup \Gamma_N$, а с $a(\mathbf{x}) = [a_{ij}(\mathbf{x})]_{i,j=1}^3$ е означена симетричната и положително определена коефициентна матрица от на части гладки функции върху Ω, $a_{ij}(\mathbf{x}) \in \mathcal{C}^0$.

Като първа стъпка в МКЕ, диференциалната задача (1.1.1) се поставя в еквивалентна вариационна формулировка. Това става по следния начин: Умножаваме (1.1.1а) с пробна функция $v \in \mathcal{V}$, $\mathcal{V} = H_D^1 = \{v \in H^1 : v = 0$ върху $\Gamma_D\}$ и след това интегрираме върху Ω. Тук с H^1 е означено соболевото пространство $W^{1,2}$ – пространството от функции с крайна L^2 норма, които имат и крайна L^2 норма на първата си производна (в слаб смисъл). След интегриране по части получаваме вариационната формулировка на задачата:

$$\mathcal{A}(u,v) = \mathcal{B}(v) \quad \text{за} \quad \forall v \in \mathcal{V}, \quad \text{където} \tag{1.1.2а}$$

$$\mathcal{A}(u,v) = \int_\Omega a(\mathbf{x})\nabla u(\mathbf{x})\nabla v(\mathbf{x})d\mathbf{x}, \tag{1.1.2б}$$

$$\mathcal{B}(v) = \int_\Omega f(\mathbf{x})v(\mathbf{x})d\mathbf{x}. \tag{1.1.2в}$$

Дефиниция 1.1.1. *Нека областта Ω е разделена на краен брой подобласти, наречени* крайни елементи *или за краткост* елементи*. Множеството от тези елементи $\mathcal{T}_h$ се нарича триангулация на Ω.*

Обикновено при МКЕ се използват еднотипни елементи, например тетраедри или шестостени. Възможни са обаче и постановки с елементи от различен тип, както и елементи с криви стени и ръбове. Ако за дадена триангулация $\mathcal{T}_h$, никой от върховете не лежи върху стена или ръб на елемент, то такава триангулация наричаме *крайноелементна мрежа* и можем да я използваме в МКЕ. Тук с h е означен максималния диаметър на елементите. Параметърът h е важна характеристика на мрежата. При прости области генерирането на крайноелемнтни мрежи е тривиално. В общия случай триангулацията се извършва от *мрежови генератори*. Това са компютърни програми, които получават за вход описание на геометрията на областта, а като изход мрежа, съставена от координати на възли, и тяхната свързаност. Обикновено в получените мрежи се маркират с различни номера отделните области и граници. Това способства за правилната дискретизация и за налагането на граничните условия. Някои популярни мрежови генератори са:

- Triangle [79],
- Gmsh [33],
- Netgen [77],
- Tetgen [80].

От съществено значение е качеството на получените мрежи, което зависи от формата на получените елементи [47, 48, 59, 78].

При метода на крайните елементи се търси приближено решение на задачата (1.1.2), в подходящо избрано крайномерно пространство $\mathcal{V}_h$. Нека $\{\phi_1, \phi_2, \ldots, \phi_N\}$ е базис на пространството $\mathcal{V}_h$. Тогава всеки две функции v_h и u_h от $\mathcal{V}_h$ могат да се представят по следния начин:

$$v_h = \sum_{i=1}^{N} v_i \phi_i, \tag{1.1.3а}$$

$$u_h = \sum_{i=1}^{N} u_i \phi_i. \tag{1.1.3б}$$

Тук v_i и u_i са реални коефициенти. Така получаваме следната задача: Да се намери функция $u_h \in V_h$ такава, че

$$\mathcal{A}_h(u_h, v_h) = \mathcal{B}_h(v_h) \quad \text{за} \quad \forall v_h \in \mathcal{V}_h, \quad \text{където} \tag{1.1.4а}$$

$$\mathcal{A}_h(u_h, v_h) = \sum_{e \in \mathcal{T}_h} \int_e a(e) \nabla u_h(\mathbf{x}), \nabla v_h(\mathbf{x}) d\mathbf{x} \tag{1.1.4б}$$

$$\mathcal{B}_h(v_h) = \int_\Omega f(\mathbf{x}) v_h(\mathbf{x}) d\mathbf{x}. \tag{1.1.4в}$$

Тук матрицата $a(\mathbf{x})$ е заменена с на части константна симетрична и положително определена матрица $a(e)$ по следния начин:

$$a(e) = \frac{1}{|e|} \int_e a(\mathbf{x}) d\mathbf{x}. \tag{1.1.5}$$

Тази постановка позволява значителни скокове на коефициентите. Като заместим (1.1.3) в (1.1.4) получаваме следната система от линейни алгебрични уравнения:

$$\left| \sum_{i=1}^{N} \mathcal{A}_h(\phi_i, \phi_j) u_i = \mathcal{B}_h(\phi_j), \quad \text{за } j = 1, \ldots, N. \right. \tag{1.1.6}$$

Системата (1.1.6) може да се запише в матричен вид:

$$A\mathbf{u} = \mathbf{b}, \tag{1.1.7}$$

където A е матрица $N \times N$ с елементи $a_{ij} = \mathcal{A}_h(\phi_i, \phi_j)$, а $\mathbf{B}$ е вектор стълб с елементи $b_i = \mathcal{B}_h(\phi_j)$. Основната черта на МКЕ е начина по който се избират базисните функции ϕ_i, и съответно и пространството V_h. Когато функциите ϕ_i се избират, така че носителят им – множеството, $\{\mathbf{x} : \phi_i(\mathbf{x}) \neq 0\}$ – да е ограничен върху малък брой елементи, тогава голяма част от елементите в матрицата A стават нула. Най-често в МКЕ се използват базиси от на части полиномиални функции (функциите са алгебрични полиноми от сравнително ниска степен, във всеки един елемент). Използват се също и други базиси – например такива от на части полиномиални сплайни, експоненциални полиноми, NURBS (неравномерни рационални Б-сплайни).

Едно от основните свойства и предимства на МКЕ е, че при неговото прилагане се получават СЛАУ с разредени матрици. Това свойство е важно, защото

системи с разредени матрици и голяма размерност могат да се решават ефективно използвайки съвременните итерационни методи.

При МКЕ крайноелементното пространство V_h, както и самата линейна система 1.1.7 се конструират на елементно ниво. За целта обикновено се използва референтен елемент. Нека $\hat{e}$ е референтния ни елемент и с L_e е означена матрицата на линейното изображение от глобалния вектор с неизвестни $\{u_i\}_{i=1}^{N}$ към вектора с неизвестни за текущия елемент e $\hat{u}_{i=1}^{N_{\hat{e}}}$. С $N_{\hat{e}}$ е означен броя на степените на свобода за елемента e. Нека също, $\{\hat{\phi}_i\}_{i=1}^{N_{\hat{e}}}$ е базис в $\hat{e}$, а с Ψ_e е означено биективното изображение от референтния елемент $\hat{e}$ в текущия e. Тогава можем да запишем матрицата A в следния вид:

$$A = \sum_{e \in \mathcal{T}_h} L_e^T A_e L_e. \tag{1.1.8}$$

Матрицата A_e с размери $N_{\hat{e}} \times N_{\hat{e}}$ се нарича елементна матрица на коравина с елементи $a_{i,j}^e$:

$$a_{i,j}^e = a(e) \int_e \nabla \left(\hat{\phi}_i \circ \Psi_e^{-1} \right) \nabla \left(\hat{\phi}_j \circ \Psi_e^{-1} \right) de.$$

Със символа $'\circ'$ е означена суперпозицията на функциите. Обикновено първо се избира типа на референтния елемент, базиса $\{\hat{\phi}_i\}_{i=1}^{N_{\hat{e}}}$, а след това с подходящи условия за непрекъснатост се конструира и пространството $\mathcal{V}_h$.

Дефиниция 1.1.2. *Когато пространството $\mathcal{V}_h$ е подпространство на $\mathcal{V}$, МКЕ се нарича* конформен. *И обратното, когато $\mathcal{V}_h$ не е подпространство на $\mathcal{V}$, МКЕ се нарича* неконформен.

.

1.1.2 Крайни елементи на Ранахер и Турек

Неконформните крайни елементи базирани на завъртени (дву)трилинейни функции на формата са предложени от Ранахер и Турек [71], като клас от опростени елементи за решаването на задачата на Стокс. Бързо се развиват през последните години методите за ефективно решаване на системи получени чрез неконформни крайни елементи. Това развитие е мотивирано от техните свойства – те са устойчиво средство за дискретизация за лошо обусловени задачи.

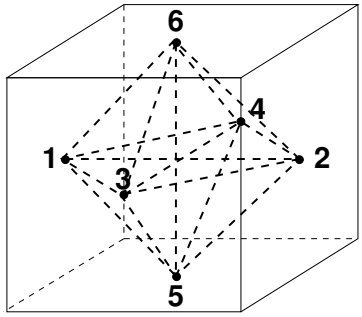
Нека T_h е триангулация на Ω съставена от шестостени.

Забележка 1.1.1. Тук, и по-нататък в тази дисертация под шестостен ще разбираме такъв изпъкнал шестостен имащ шест четириъгълни стени.

За референтен елемент $\hat{e}$ е избран кубът $[-1,1]^3$ (вижте Фиг. 1.1). Той се използва за дефиниране на елемента $e \in \mathcal{T}_h$.

Нека $\Psi_e : \hat{e} \to e$ е трилинейната трансформация изобразяваща референтния елемент $\hat{e}$ в e. Тогава базисните функции се дефинират по следния начин:

$$\begin{array}{rcl} \{\phi_i\}_{i=1}^{6} & = & \{\hat{\phi}_i \circ \Psi_e^{-1}\}_{i=1}^{6}, \\ \{\hat{\phi}_i\}_{i=1}^{6} & \in & l\left(1, x_1, x_2, x_3, x_2^2 - x_1^2, x_1^2 - x_3^2\right). \end{array} \tag{1.1.9}$$



Фигура 1.1: Номерация на възлите на референтния елемент на Ранахер-Турек $\hat{e}$ и шаблон на свързаността на съответната елементна матрица на коравина A_e.

Има два естествени варианта за избор на възловите интерполационни условия, които водят до две различни крайноелементни пространства. При първият – MP (mid point – средна точка), $\{\hat{\phi}_i\}_{i=1}^6$ са дефинирани от поточковите интерполационни условия

$$\hat{\phi}_i(b_\Gamma^j) = \delta_{ij},$$

където $b_\Gamma^j, j = 1, 2, \ldots, 6$ са центровете на маса на стените на куба $\hat{e}$. Така получаваме следните базисни функции.

$$\begin{aligned}
\hat{\phi}_1(\mathbf{x}) &= (1 - 3x_1 + 2x_1^2 - x_2^2 - x_3^2)/6, \\
\hat{\phi}_2(\mathbf{x}) &= (1 + 3x_1 + 2x_1^2 - x_2^2 - x_3^2)/6, \\
\hat{\phi}_3(\mathbf{x}) &= (1 - x_1^2 - 3x_2 + 2x_2^2 - x_3^2)/6, \\
\hat{\phi}_4(\mathbf{x}) &= (1 - x_1^2 + 3x_2 + 2x_2^2 - x_3^2)/6, \\
\hat{\phi}_5(\mathbf{x}) &= (1 - x_1^2 - x_2^2 - 3x_3 + 2x_3^2)/6, \\
\hat{\phi}_6(\mathbf{x}) &= (1 - x_1^2 - x_2^2 + 3x_3 + 2x_3^2)/6.
\end{aligned}$$

При другия вариант – MV (mean value – средна стойност) се използва усредняващ интерполационен функционал във вида:

$$|\Gamma_{\hat{e}}^j|^{-1} \int_{\Gamma_{\hat{e}}^j} \hat{\phi}_i = \delta_{ij},$$

където $\Gamma_{\hat{e}}^j, j = 1, 2, \ldots, 6$ са стените на куба $\hat{e}$, базисните функции се получават в следния вид:

$$\begin{aligned}
\hat{\phi}_1(\mathbf{x}) &= (2 - 6x_1 + 6x_1^2 - 3x_2^2 - 3x_3^2)/12, \\
\hat{\phi}_2(\mathbf{x}) &= (2 + 6x_1 + 6x_1^2 - 3x_2^2 - 3x_3^2)/12, \\
\hat{\phi}_3(\mathbf{x}) &= (2 - 3x_1^2 - 6x_2 + 6x_2^2 - 3x_3^2)/12, \\
\hat{\phi}_4(\mathbf{x}) &= (2 - 3x_1^2 + 6x_2 + 6x_2^2 - 3x_3^2)/12, \\
\hat{\phi}_5(\mathbf{x}) &= (2 - 3x_1^2 - 3x_2^2 - 6x_3 + 6x_3^2)/12, \\
\hat{\phi}_6(\mathbf{x}) &= (2 - 3x_1^2 - 3x_2^2 + 6x_3 + 6x_3^2)/12.
\end{aligned}$$

И при двата варианта на интерполационни условия, степените на свобода са асоциирани със стените на елемента. Това е съществено, защото дава

възможност за единна програмна реализация, при която като параметър се задават различните функции на формата.

Трябва да отбележим, че и при двата варианта, базисните функции са трилинейни, в координатната система с оси правите свързващи срещуположните върхове на $\hat{e}$. Затова крайните елементи на Ранахер и Турек се наричат също и *завъртени трилинейни*.

1.2 Линейна теория на еластичността

В този раздел ще представим накратко някои основни сведения от линейната теория на еластичността. За повече сведения вижте [8, 64, 91].

Нека Ω е затворена област в $\mathbb{R}^3$ с граница $\Gamma = \Gamma_D \cup \Gamma_N$. В областта Ω е разположено тяло, на което по границата Γ_N са приложени сили, а по границата Γ_D са предписани известни премествания. Тогава под действието на силите тялото се деформира, до достигане на равновесие между външните сили и напреженията породени от деформациите.

Ако дадена точка R от тяло е с радиус-вектор $\mathbf{r}_0$ в нормалното (ненатоварено) състояние на тялото и с радиус-вектор $\mathbf{r}$ в равновесното натоварено състояние на тялото, то с $\mathbf{u} = \mathbf{r} - \mathbf{r}_0$, $\mathbf{u} = \{u_1, u_2, u_3\}$ означаваме вектора на преместванията.

Компонентите на тензора на малките деформации са

$$\epsilon_{ij} = \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad 1 \leq i, j \leq 3. \tag{1.2.10}$$

Фигура 1.2: Крива на натоварване за стомана [72]: 1. Максимална коравина; 2. Граница на пропорционалност.

В линейната теория на еластичността се приема допускането, че напреженията зависят линейно от деформациите. Това допускане дава много добро приближение при малки премествания за много материали. Това са материали, като стомана, бетон, гума и други. На Фигура 1.2 е изобразена кривата на натоварване за типична стомана (зависимостта между напреженията и деформациите). На нея се вижда, че при малки деформации зависимостта е линейна. Точката в която зависимостта спира да е линейна се нарича *граница на пропорционалност* или *граница на еластичност*.

От горното допускане получаваме следната връзка между тензора на напреженията σ и деформациите:

$$\sigma_{ij} = \sum_{k=1}^{3} \sum_{l=1}^{3} c_{ijkl} \epsilon_{kl}, \tag{1.2.11}$$

където коефициентите c_{ijkl} определят поведението на материала. Равенството (1.2.11) е известно като закон на Хук.

Когато материалът е изотропен (във всички направления свойствата са му еднакви) свойствата му могат да се опишат само с два независими коефициента:

$$c_{iiii} = \lambda + 2\mu \qquad i = 1, 2, 3, \tag{1.2.12}$$

$$c_{iijj} = \lambda \qquad i, j = 1, 2, 3; i \neq j, \tag{1.2.13}$$

$$c_{ijij} = c_{ijji} = \mu \qquad i, j = 1, 2, 3; i \neq j, \tag{1.2.14}$$

останалите коефициенти c_{ijkl} са нули. Коефициентите λ и μ се наричат коефициенти на Ламе. Други два такива коефициента са модулът на еластична деформация E, известен също като модул на Юнг и коефициентът на Поасон ν. В сила са следните връзки:

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \tag{1.2.15а}$$

$$\nu = \frac{\lambda}{2(\lambda + \mu)}, \tag{1.2.15б}$$

$$\lambda = \frac{E\nu}{(1 + \nu)(1 - 2\nu)}, \tag{1.2.15в}$$

$$\mu = \frac{E}{2(1 + \nu)}. \tag{1.2.15г}$$

В случай на изотропен материал законът на Хук може да се напише по следния начин:

$$\sigma_{ij} = \lambda \sum_{k=1}^{3} \epsilon_{kk} \delta_{ij} + 2\mu\epsilon_{ij} \quad i, j = 1, 2, 3. \tag{1.2.16}$$

В горното равенство с δ_{ij} е означен символът на Кронекер.

От втория закон на Нютон за движение могат да се изведат следните равенства [81]:

$$\sum_{j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} + F_i = \rho \frac{\partial^2 u_i}{\partial t^2} \quad i = 1, 2, 3, \tag{1.2.17}$$

където с t е означено времето, а с $\mathbf{F}$ са означени обемните сили. Когато ни интересува равновесното положение на тялото, можем да заместим ускоренията в дясната част на (1.2.17) с нула. Ако заместим (1.2.16) и (1.2.10) в (1.2.17) получаваме системата на Навие-Ламе:

$$(\lambda+\mu)\sum_{j=1}^{3}\frac{\partial^2 u_j}{\partial x_i \partial x_j}+\mu\sum_{j=1}^{3}\frac{\partial^2 u_i}{\partial x_j^2}=F_i \quad i=1,2,3 \tag{1.2.18}$$

със следните гранични условия

$$u_i(\mathbf{x})=g_i(\mathbf{x}) \qquad \mathbf{x}\in\Gamma_D, \tag{1.2.19}$$

$$\sum_{j=1}^{3}\sigma_{ij}(\mathbf{x})n_j=h_i(\mathbf{x}) \qquad \mathbf{x}\in\Gamma_N, \tag{1.2.20}$$

където с n_j са означени компонентите на нормалния вектор $\mathbf{n}$ в точката $\mathbf{x}$ на границата сочещ вън от Ω.

1.3 Метод на спрегнатия градиент с преобусловяне

Най-добрият известен пряк метод за решаване на СЛАУ е този на вложените сечения. Неговата изчислителна сложност в двумерния случай е $O(N^{3/2})$, където N е броят на неизвестните. Разработват се също така и итерационни методи [90, 92]. При тях решението се получава като граница на последователни приближения $x^0, x^1, \ldots$, всяко следващо от които се конструира с помощта на предишните (най-често само чрез последното). Най-добрият известен итерационен метод за системи с положително определени и симетрични матрици е *методът на спрегнатия градиент* (МСГ)[6]. Той има следния вид:

Алгоритъм 1.1. *Метод на спрегнатия градиент* $\mathbf{x} = МСГ(A, \mathbf{b})$*:*

$$\begin{aligned}
\mathbf{x}^0 &= 0,\ \mathbf{g}^0 = A\mathbf{x}^0 - \mathbf{b},\ \mathbf{d}^0 = -\mathbf{h}^0,\ \gamma_0 = \left(\mathbf{g}^0, \mathbf{h}^0\right)\\
k &= 0,\ 1,\ 2, \ldots\\
\mathbf{t}^k &= A\mathbf{d}^k\\
\tau_k &= \frac{\gamma_k}{(\mathbf{d}^k, \mathbf{t}^k)}\\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \tau_k \mathbf{d}^k\\
\mathbf{g}^{k+1} &= \mathbf{g}^k + \tau_k \mathbf{t}^k\\
\gamma_{k+1} &= \left(\mathbf{g}^{k+1}, \mathbf{g}^{k+1}\right)\\
\beta_k &= \frac{\gamma_{k+1}}{\gamma_k}\\
\mathbf{d}^{k+1} &= \beta_k \mathbf{d}^k - \mathbf{g}^{k+1}
\end{aligned}$$

Нека припомним, че ако $\{\lambda_1, \ldots, \lambda_N\}$ са подредените в нарастващ ред собствени стойности на симетричната и положително определената матрица A (те

са реални и положителни), то спектралното число на обусловеност можем да изразим по следния начин:

$$\kappa(A) = \frac{\lambda_N}{\lambda_1}.$$

Тогава относно сходимостта на МСГ е в сила следната теорема [90]:

Теорема 1.3.1. *Нека $p(\epsilon)$ е най-малкото цяло положително число k, за което е изпълнено неравенството*

$$||\mathbf{x}^k - \hat{\mathbf{x}}||_A \leq \epsilon ||\mathbf{x}^0 - \hat{\mathbf{x}}||_A.$$

В сила е следната оценка за $p(\epsilon)$:

$$p(\epsilon) \leq \frac{1}{2}\sqrt{\kappa(A)} ln\left(\frac{2}{\epsilon}\right) + 1.$$

□

Тоест броят на итерациите за достигане на желаната от нас точност е пропорционален на квадратния корен от числото на обусловеност на матрицата на системата:

$$\mathcal{N}_{it} = O(\sqrt{\kappa(A)}). \tag{1.3.21}$$

След дискретизация на тримерна елиптична задача, за числото на обусловеност на матрицата е в сила оценката

$$\kappa = O(N^{2/3}),$$

където N е броят на неизвестните.

На всяка итерация при разредена матрица се извършват $O(N)$ операции. Броят на итерациите при прилагане на метода на спрегнатия градиент е $O(N^{1/3})$. Следователно общата изчислителна сложност е $O(N^{4/3})$.

Този резултат може да бъде подобрен още чрез преобуславяне. Идеята на този подход е, вместо системата

$$Ax = b,$$

да решим системата

$$E^{-1^T} A E^{-1} y = E^{-1^T} b,$$

където $y = Ex$, а E е неособена матрица и числото на обусловеност на матрицата $E^{-1^T} A E^{-1}$ е по-малко от числото на обусловеност на A. По този начин намаляваме броя на итерациите. Този метод се нарича метод на спрегнатия градиент с преобуславяне (МСГП). При него на всяка итерация е необходимо решаване на системи с матрица $C = E^T E$. Тази матрица се нарича *преобусловител*.

Алгоритъмът на спрегнатия градиент с преобусловител за системата $Ax = b$ може да се запише по следния начин [76, 90]:

Алгоритъм 1.2. *Метод на спрегнатия градиент с преобуславяне* $\mathbf{x} = МСГП(A, \mathbf{b})$*:*

$$\begin{aligned}
\mathbf{x}^0 &= 0,\ \mathbf{g}^0 = A\mathbf{x}^0 - \mathbf{b},\ \mathbf{h}^0 = C^{-1}\mathbf{g}^0,\ \mathbf{d}^0 = -\mathbf{h}^0,\ \gamma_0 = \left(\mathbf{g}^0, \mathbf{h}^0\right) \\
k &= 0,\ 1,\ 2, \ldots \\
\mathbf{t}^k &= A\mathbf{d}^k \\
\tau_k &= \frac{\gamma_k}{(\mathbf{d}^k, \mathbf{t}^k)} \\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \tau_k \mathbf{d}^k \\
\mathbf{g}^{k+1} &= \mathbf{g}^k + \tau_k \mathbf{t}^k \\
\mathbf{h}^{k+1} &= C^{-1}\mathbf{g}^{k+1} \\
\gamma_{k+1} &= \left(\mathbf{g}^{k+1}, \mathbf{h}^{k+1}\right) \\
\beta_k &= \frac{\gamma_{k+1}}{\gamma_k} \\
\mathbf{d}^{k+1} &= \beta_k \mathbf{d}^k - \mathbf{h}^{k+1}
\end{aligned}$$

Дефиниция 1.3.1 *Изчислителна сложност*
Под изчислителна сложност на даден израз или алгоритъм, разбираме броя на аритметичните операции, необходими за пресмятането му. С $\mathcal{N}(X)$ означаваме изчислителната сложност на израза X.

Общата стратегията за конструиране на ефективни преобусловители е да съсъществува ефективен алгоритъм за решаване на системи с преобусловителя C и числото на обусловеност на $C^{-1}A$ да е много по-малко от това на A, тоест:

$$\mathcal{N}\left(C^{-1}x\right) << \mathcal{N}\left(A^{-1}x\right)$$

и

$$\kappa\left(C^{-1}A\right) << \kappa\left(A\right)$$

На всяка итерация в МСГП се извършва едно умножение на матрицата A с вектор, две скаларни произведения на вектори, три операции от тип умножение на скалар с вектор плюс вектор, едно решаване на система с преобуславящата матрица C и две деления. Тоест:

$$\mathcal{N}_{it}^{PCG}\left(A^{-1}\mathbf{b}\right) = \mathcal{N}\left(A\mathbf{x}\right) + 2\mathcal{N}\left((\mathbf{a}, \mathbf{b})\right) + 3\mathcal{N}\left(\alpha\mathbf{a} + \mathbf{b}\right) + \mathcal{N}\left(C^{-1}\mathbf{x}\right) + 2$$

Забележка 1.3.2. При реализация на така разписания Алгоритъм 4.2, освен векторите на дясната част $\mathbf{b}$ и вектора с неизвестните $\mathbf{x}$ са необходими още 4 помощни вектора – $\mathbf{d}, \mathbf{g}, \mathbf{h}, \mathbf{t}$. Лесно се вижда, че може да се използва един и същ физически вектор за $\mathbf{h}$ и $\mathbf{t}$. Също така, често дясната част $\mathbf{b}$ не е необходима след решаването на линейната система. В такъв случай, може векторите $\mathbf{b}$ и $\mathbf{d}$ да бъдат разположени на едно и също място в паметта. Тези оптимизации водят до съществено намаляване на необходимата оперативна памет за реализация на алгоритъма, което е важно при решаване на задачи с голяма размерност. И двете оптимизации се използват от програмите в дисертацията. Първата – винаги, а втората по избор.

1.4 MIC(0) факторизация

Един от широко използваните методи за конструиране на преобусловители е чрез така наречената непълна факторизация (ILU от английски *incomplete LU*). Добре известен факт е, че в общия случай, обратната матрица A^{-1} е плътна, дори когато матрицата A е разредена. Това създава много проблеми, които правят непрактични използването на пълни факторизации, за големи разредени матрици.

Общ подход за конструирането на непълни факторизации на матрица A е намирането на долнотриъгълна разредена матрица L и такава горнотриъгълна U, така че резидула $R = LU - A$ да изпълнява определени критерии [74].

В тази работа разглеждаме непълна факторизация от нулев ред. Това означава, че матриците L и U имат ненулеви елементи само на местата на които елементите от съответните долно- и горно-триъгълни части на изходната матрица A са различни от нула. Когато матрицата A е симетрична, тогава $L = U^T$ и говорим за непълна факторизация на Холецки (IC - *incomplete Cholecki*).

Основен инструмент в дисертацията е преобусловителят от тип $MIC(0)$ (модифицирана непълна факторизация на Холецки). Модификацията се изразява в допълнителното изискване за еднакви суми по редовете, на изходната матрица и на приближената ѝ факторизация.

При прилагането на метода на спрегнатия градиент с $MIC(0)$ преобусловител за задачата (1.1.1) броят на операциите е $O(N^{7/6})$ [36]. Въпреки че съществуват и по-добри преобусловители, този се откроява със съотношението между простота при реализацията и сравнително неголямата си изчислителна сложност.

Да припомним някои факти за факторизацията от тип MIC(0). Нека A е симетрична и положително определена реална матрица с размери $N \times N$. Записваме матрицата A в следния вид

$$A = D - L - L^T,$$

където D е диагоналната част на матрицата A, а $(-L)$ е строго долно триъгълната част на A. Тогава ще търсим MIC(0) преобусловител C във вида:

$$C = (X - L)\, X^{-1} \left(X - L^T\right). \tag{1.4.22}$$

Диагоналната матрица X ще определим от условието:

$$A\mathbf{e} = C\mathbf{e},\ \mathbf{e}^T = (1, 1, \ldots, 1).$$

Тъй като ще използваме матрицата C за преобусловител, се интересуваме от случая, когато $x_{i,i} > 0$. В сила е следната теорема:

Теорема 1.4.1. *Нека $A = (a_{i,j})$ е симетрична и положително определена реална матрица с размери $N \times N$, $A = D - L - L^T$, като са в сила неравенствата (в поелементен смисъл)*

$$L \geq 0, \tag{1.4.23}$$

$$A\mathbf{e} \geq \mathbf{0}, \tag{1.4.24}$$

$$A\mathbf{e} + L^T\mathbf{e} > \mathbf{0}\ ,\ \ \mathbf{e} = (1, 1, \ldots, 1)^T \in R^N. \tag{1.4.25}$$

Тогава може да бъде намерена диагонална матрица X*, такава че* $x_{i,i} > 0$ *по следния начин:*

$$x_{i,i} = a_{i,i} - \sum_{k=1}^{i-1} \frac{a_{i,k}}{x_{k,k}} \sum_{j=k+1}^{N} a_{k,j}, \qquad \textit{за } i\textit{=1, . . . , N.}$$

□

Доказателството на тази теорема може да бъде намерено в [14].

Забележка 1.4.2. При всички числени експерименти в дисертацията се използва пертурбиран вариант на MIC(0) алгоритъма, където факторизацията се прилага към матрица $\tilde{A} = A + \tilde{D}$. Диагоналната пертурбация $\tilde{D} = \tilde{D}(\xi) = diag(\tilde{d}_1, \ldots \tilde{d}_N)$ се дефинира по следния начин:

$$\tilde{d}_i = \begin{cases} \xi a_{ii}, & \text{if } a_{ii} \geq 2w_i, \\ \xi^{1/2} a_{ii}, & \text{if } a_{ii} < 2w_i, \end{cases}$$

където

$$w_i = -\sum_{j>i} a_{ij},$$

а $0 < \xi < 1$ е константа от порядъка на най-малката собствена стойност на матрицата A. При числените експерименти е използван $\xi = h^{-2}$ при крайни елементи на Ранахер-Турек от тип MP и $\xi = 16h^{-2}$ при такива от тип MV, където h е параметъра на дискретизация.

Нека A е матрицата на коравина за тримерната задача:

$$-\Delta u \;=\; f. \tag{1.4.26}$$

Тогава е в сила следната оценка за числото на обусловеност на матрицата $C^{-1}A$

$$\kappa\left(C^{-1}A\right) = O\left(N^{\frac{1}{3}}\right),$$

където C е пертурбираната MIC(0) факторизацията на A [14].

Както се вижда от (1.4.22), решаването на системи с матрица C се свежда до решаване на две системи с триъгълни матрици и една с диагонална. Следователно, ако матрицата A е разредена, то за решаването на система с матрицата C трябва се извършват $O(N)$ операции. Трябва да подчертаем, че в общия случай, ако не знаем нищо за структурата на A, решаването на получаващите се триъгълни системи в C в същността си е рекурентно и не се поддава на ефективно разпаралеляване.

1.5 Алгебричен мултигрид

Мултигрид методите (MG) (от английски *multigrid* – много-мрежов) водят началото си от работи на *Радий Федоренко* и *Николай Бахвалов* през шейсетте години на двадесети век [9, 26]. Обаче чак *Ахи Брандт* пръв показва тяхната практическа полза за широк кръг задачи през седемдесетте години на същия

век [17, 18]. Повече информация относно MG, може да бъде намерена в [22, 39, 40].

В основата на MG методи е наблюдението, че сходимостта на методите за релаксация (на Якоби, на Гаус-Зайдел, SOR, виж. [93, 94]) след едва няколко итерации силно се влошава. Този ефект става още по-добре изразен ако се използват по-фини мрежи. Оказва се, че тези методи работят много по-добре на по-груби мрежи. При изследване на причините за това се оказва, че релаксационните методи успяват много бързо да намалят високочестотните компоненти на грешката и обратно, нискочестотни такива намаляват много бавно. На практика те "изглаждат" вектора на грешката. От там и идва основната идея на мултигрид методите, и по-специално на геометричните такива: бавно променящата се грешка да бъде намалявана на все по-груби мрежи.

Напоследък станаха популярни алгебрични мултигрид методи (AMG), при които не е нужна геометрична информация, за построяването на грубите мрежи [19, 20].

Да скицираме общия алгоритъм на алгебричен мултигрид. По-подробно описание може да се намери например в [73]. Разглеждаме следната система линейни уравнения:

$$A\mathbf{u} = \mathbf{f}, \tag{1.5.27}$$

където A е симетрична и положително определена разредена матрица $n \times n$ с елементи a_{ij}. За удобство, ще асоцииряме индексите с точките от мрежата, така с u_i сме означили стойността на $\mathbf{u}$ в точката i, и мрежата е означена с $\Omega = \{1, 2, \ldots, n\}$. Както отбелязахме по-рано, основната идея във всички мултигрид методи е "гладката грешка" $\mathbf{g}$, която не се намалява от релаксацията, да бъде редуцирана чрез корекция от по-грубите мрежи. Това се постига решавайки системата $A\mathbf{g} = \mathbf{r}$ на по-груба мрежа, където $\mathbf{r} = \mathbf{f} - A\mathbf{u}$ е резидула. След това интерполирайки грешката от грубата мрежа върху по-фината, поправяме текущото приближение $\mathbf{u} \leftarrow \mathbf{u}+\mathbf{g}$. С горни индекси са означени нивата, като с 1 означаваме нивото на най-фината мрежа. Така, че $A^1 = A$ и $\Omega^1 = \Omega$. Общият мултигрид алгоритъм се състои от следните части:

- „Мрежи": $\Omega^1 \supset \Omega^2 \supset \ldots \supset \Omega^M$;
- Мрежови оператори : $A^1, A^2, \ldots, A^M$;
- Оператори за преход между мрежите:
  Интерполация I_{k+1}^k, $k = 1, 2, \ldots, M-1$
  Рестрикция I_k^{k+1}, $k = 1, 2, \ldots, M-1$;
- Метод за релаксация за всяко ниво.

Ако съставните части са определени, можем да дефинираме рекурсивния мултигрид алгоритъм, така наречения V цикъл (названието идва от формата на латинската буква "V" – която илюстрира движението на алгоритъма от най-финото ниво, към най-грубото и обратно):

Алгоритъм 1.3. *$MV^k(\mathbf{u}^k, \mathbf{f}^k)$ – (μ_1, μ_2) V-цикъл*
Ако $k = M$ пресмятаме $\mathbf{u}^M = (A^M)^{-1}\mathbf{f}^M$

в противен случай:
Релаксираме μ_1 *пъти* $A^k\mathbf{u^k} = \mathbf{f^k}$
Правим корекция от по-грубата мрежа:
$\mathbf{u}^{k+1} = 0$, $\mathbf{f}^{k+1} = I_k^{k+1}(\mathbf{f}^k - A^k\mathbf{u}^k)$
Решаваме $MV^{k+1}(\mathbf{u}^{k+1}, \mathbf{f}^{k+1})$ *на нивото* $k+1$.
Поправяме решението $\mathbf{u}^k \leftarrow \mathbf{u}^k + I_{k+1}^k\mathbf{u}^{k+1}$
Релаксираме μ_2 *пъти* $A^k\mathbf{u^k} = \mathbf{f^k}$

Изборът на съставните части при алгебричния мултигрид става, на отделна "инициализираща" стъпка:

Алгоритъм 1.4. *Инициализация при AMG*
(1) Инициализираме $k = 1$.
(2) Разделяме Ω^k *на две непресичащи се множества* C^k *и* F^k.
(а) Полагаме $\Omega^{k+1} = C^k$
(б) Дефинираме интерполацията I_{k+1}^k.
(3) Означаваме $I_k^{k+1} = (I_{k+1}^k)^T$ *и* $A^{k+1} = I_k^{k+1}A^kI_{k+1}^k$.
(4) Ако Ω^{k+1} *е достатъчно малко, то* $M = k+1$ *и спираме,*
в противен случай $k = k+1$ *и се връщаме към стъпка (2)*

От голямо значение за ефективността на метода е начина по който избираме множествата от по-грубите мрежи C^k (загрубяването).

Мултигрид методите имат оптимална изчислителна сложност за широк клас задачи. Те могат да бъдат използвани, както директно, така и като преобусловители. В тази работа, ние сме използвали паралелния мултигрид "BoomerAMG", част от софтуерния пакет "Hypre", разработван в Лорансовата национална лаборатория в Ливърмор, САЩ. Нашите цели са, както да изследваме приложимостта на мултигрид методите върху разглежданите класове задачи (в това число за конструиране на ефективни блочни алгоритми), така и за сравнение с разработените в дисертацията преобусловители на базата на непълна факторизация. BoomerAMG съдържа последователна и паралелна реализация на AMG. В [41, 87] могат да се намерят подробно описание на използваните алгоритми за загрубяване, интерполация, както и числени експерименти. На разположение са следните методи за загрубяване:

- загрубяване на Клири-Люби-Джоунс-Пласман (CLJP),
- редица варианти на класическите загрубявания на Руге-Щубен (RS), както и
- загрубяване на Фалгот, което е комбинация на CLJP и RS алгоритмите.

Могат да се използват, един или няколко метода на релаксация от:

- релаксация на Якоби,
- релаксация на Гаус-Зайдел,
- хибридна Гуас-Зайдел/Якоби релаксация,
- симетрична хибридна Гуас-Зайдел/Якоби релаксация.

1.6 Бележки относно програмната реализация

Съвременните паралелни архитектури включват:

- Многопроцесорни системи с обща памет;
- Графични ускорители (видеокарти);
- Системи с разпределена памет (клъстери).

Възможни са и се използват хибридни варианти, които са комбинации от горните варианти. Всяка една от горните архитектури има своите плюсове и минуси. Когато програмите имат достъп до едно и също адресно пространство говорим за система с *обща памет*. В противен случай говорим за системи с *разпределена памет*.

Системите с обща памет позволяват най-голяма свобода при програмиране, могат да се използват вече готови последователни програми и библиотеки, от които да се "сглобяват" паралелни такива. Недостатъците са, че паралелизма е ограничен до сравнително малко на брой процесори. Също така размерът на решаваните задачи е ограничен от размера на общата памет, която достига десетки гигабайти. Цените на системите започват бързо да растат с увеличаване на броя на процесорите и количеството оперативна памет.

Все по-голяма популярност придобива използването на графични ускорители за изчисления. Те се наричат така поради исторически причини – възникнали са (както и продължават да се използват) за реализация на тримерна графика и компютърни игри, като постепенно са се превърнали в мощни векторни процесори. Към днешна дата те имат най-добро отношение „изчислителна мощност/цена," както и много добро отношение „потребявана енергия/изчислителна мощност." Основния им недостатък е трудността за програмиране. Всеки ускорител има ограничен размер собствена памет (няколко гигабайта), като централният процесор трябва да прехвърля към и от ускорителя данните и резултатите от изчисленията. Има два различни, макар и подобни, начина за програмиране – системата CUDA [65] и стандарта OpenCL [35]. Това ограничава преносимостта на програмите и увеличава необходимите за програмиране ресурси.

Третият вид системи – тези с разпределена памет позволяват решаването на огромни задачи. Съвременните паралелни компютри с разпределена памет представляват съвкупност от системи с обща памет с по няколко процесора (например с 2, с 8 или с 16) и няколко гигабайта оперативна памет. Възможно е, обаче, да се използват голям брой такива системи – така се получават машини с огромен изчислителен капацитет. Техен недостатък е трудността на програмиране – изискват се специализирани алгоритми. Също така комуникациите, които се налагат могат да намалят ефективността на програмите.

За реализация на алгоритмите в тази дисертация са използвани следните програмни инструменти:

MPI Библиотека и стандарт за писане на паралелни програми върху системи с разпределена памет. Акронимът произхожда от „Message Passing Interface" – система за предаване на съобщения [27, 28, 82].

C++ Език за програмиране с общо предназначение.

При MPI се стартират n на брой програми, обикновено всяка на различен процесор. Процесорите, могат да са както на една, така и на различни машини.

Стандартът MPI е *естественият* избор при реализация на паралелни изчислителни програми, реализиращи алгоритми с голяма изчислителна сложност. Практически всички производители на високопроизводителни компютърни системи поддържат този стандарт, имат специализирани и оптимизирани реализации (драйвер + библиотека) на стандарта. MPI се базира на размяната на съобщения. Друго предимство е, че MPI програмите могат да бъдат изпълнявани без модификация, и на системи с обща памет.

Езикът C++ е избран защото той е един от трите езика, които стандартът MPI поддържа директно (другите два са езиците C и FORTRAN). Друга причина е наличието на така наречените шаблони в този език.

Шаблоните позволяват една единствена реализация на даден алгоритъм или структура от данни, да може да бъде използвана върху различни типове данни. Това е изключително мощно средство [1], което както ще видим в Глава 2 и в Глава 3 помага много при решаването на някои технически проблеми.

Забележка 1.6.1. Разработването и използването на шаблони освен плюсове, има и някои недостатъци: кодът, използващ шаблони, често е по-трудно разбираем, времето за компилиране на програми използващи шаблони е по-голямо (в пъти) от същото за *аналогични програми*, които не използват шаблони.

Глава 2

Паралелен MIC(0) преобусловител за елиптични задачи

Тази глава е посветена на конструирането на паралелен алгоритъм за MIC(0) преобуславяне на скаларни елиптични задачи. Изведени са теоретични оценки за сходимостта на предложения алгоритъм, както и за паралелната ефективност. Представени са и числени експерименти илюстриращи получените оценки.

Резултатите, включени в тази глава са публикувани в [2, 53, 55, 84]:

- Y. Vutov. Parallel incomplete factorization of 3D NC FEM elliptic systems. *Numerical Methods and Applications*, *LNCS* **4310**, 114–121. Springer-Verlag, 2007
- I. Lirkov and Y. Vutov. Comparative analysis of high performance solvers for 3D elliptic problems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 483–492, 2007
- P. Arbenz, S. Margenov, and Y. Vutov. Parallel MIC(0) preconditioning of 3D elliptic problems discretized by Rannacher–Turek finite elements. *Computers and Mathematics with Applications*, 55(10):2197–2211, 2008
- I. Lirkov, Y. Vutov, M. Paprzycki, and M. Ganzha. Benchmarking Performance Analysis of Parallel Solver for 3D Elasticity Problems. *Large-Scale Scientific Computing*, *LNCS* **4818**, 705–7012. Springer-Verlag, 2008

2.1 Постановка на задачата

Разглеждаме следната елиптична гранична задача:

$$\begin{aligned} -\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= 0, \qquad \mathbf{x} \in \Gamma_D, \\ (a(\mathbf{x})\nabla u(\mathbf{x})) \cdot \mathbf{n} &= 0, \qquad \mathbf{x} \in \Gamma_N, \end{aligned} \tag{2.1.1}$$

където Ω е област в $\mathbb{R}^3$, $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$, $a(\mathbf{x}) = [a_{ij}(\mathbf{x})]_{i,j=1}^3$ е симетрична и положително определена матрица от на части гладки функции върху Ω.

2.2 Дискретизация

Вариационната формулировка на задачата (2.1.1) е както следва: при дадена функция $f \in L^2(\Omega)$, да се намери функция $u \in \mathcal{V} \equiv H_D^1(\Omega) = \{v \in H^1(\Omega) : v = 0|_{\Gamma_D}\}$, удовлетворяваща уравнението

$$\mathcal{A}(u,v) = (f,v) \quad \forall v \in H_D^1(\Omega), \quad \text{където} \tag{2.2.2}$$

$$\mathcal{A}(u,v) = \int_\Omega a(\mathbf{x}) \nabla u(\mathbf{x}) \cdot \nabla v(\mathbf{x}) d\mathbf{x}. \tag{2.2.3}$$

Нека $\mathcal{T}$ е разделяне на областта Ω на шестостени. Ще предполагаме, че разделянето $\mathcal{T}$ е съгласувано с прекъсванията на коефциентната матрица $a(\mathbf{x})$, тоест за всеки елемент $e \in \mathcal{T}$, $a(\mathbf{x})$ е непрекъсната функция. Вариационната задача (2.2.2) се дискретизира по метода на крайните елементи. Непрекъснатото пространство $\mathcal{V}$ се заменя от крайно-мерно пространство $\mathcal{V}_h$. По този начин получаваме следната постановката: да се намери $u_h \in \mathcal{V}_h$, удовлетворяваща

$$\mathcal{A}_h(u_h, v_h) = (f, v_h) \quad \forall v_h \in \mathcal{V}_h, \text{ където} \tag{2.2.4}$$

$$\mathcal{A}_h(u_h, v_h) = \sum_{e \in \mathcal{T}} a(e) \int_e \nabla u_h \cdot \nabla v_h d\mathbf{x}.$$

Тук $a(e)$ е на части константна функция, дефинирана чрез интегрално усредняване на $a(\mathbf{x})$ върху всеки елемент на $\mathcal{T}$. Важно е да отбележим, че този подход допуска големи скокове на коефициентите по границите между елементите. Дискретната задача, която получаваме е системата линейни уравнения

$$A_h \mathbf{u}_h = \mathbf{f}_h, \tag{2.2.5}$$

където с A_h, $\mathbf{u}_h$, и $\mathbf{f}_h$ са означени съответно матрицата на коравина, вектора на неизвестните степени на свобода и дясната част. Тук h е параметърът на дискретизацията за разделянето $\mathcal{T}$ на Ω. Нашата основна цел е разработването на ефективни методи, алгоритми и програмни средства за решаване на системата (2.2.5), която от сега нататък ще записваме съкратено във вида

$$A\mathbf{x} = \mathbf{b}. \tag{2.2.6}$$

2.3 Конструкция на метода за преобуславяне

Ще конструираме паралелен MIC(0) преобусловител за системата (2.2.6).

2.3.1 Идея на паралелния алгоритъм

От вида на MIC(0) факторизацията (1.4.22) се вижда, че решаването на СЛАУ с така факторизирана матрица се свежда до решаването на други три СЛАУ: Една с долнотриъгълна матрица, една с диагонална и една с горнотриъгълна матрица. Решаването на диагонални системи е тривиално. Решаването

на триъгълни системи се свежда до прилагане на обратния ход на метода на Гаус. Системата

$$L\mathbf{x} = \mathbf{b}, \tag{2.3.7}$$

където L е долнотриъгълна матрица $N \times N$ с елементи $\{l_{i,j}\}$ се решава по следния начин:

$$x_i = \frac{1}{l_{i,i}} \left(b_i - \sum_{j=1}^{i-1} l_{j,i} x_j \right), \quad \text{за } i\text{=1,...,N.} \tag{2.3.8}$$

В (2.3.8) имаме рекурентна зависимост, като за пресмятането на x_i е необходимо всички $\{x_j : j < i\}$ да са вече пресметнати.

Нека напишем системата (2.3.7) в блочен вид:

$$\begin{bmatrix} L^{1,1} & & & \\ L^{2,1} & L^{2,2} & & \\ \vdots & \vdots & \ddots & \\ L^{m,1} & L^{m,2} & \cdots & L^{m,m} \end{bmatrix} \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^m \end{bmatrix} = \begin{bmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \\ \vdots \\ \mathbf{b}^m \end{bmatrix}, \tag{2.3.9}$$

където блоковете $L^{k,l}$ са матрици с елементи $\left\{ l_{i,j}^{k,l} \right\}$ и размер $N^k \times N^l$, векторите $\mathbf{x}^k$ и $\mathbf{b}^k$ са с размер N^k и с елементи съответно $\left\{ x_i^k \right\}$ и $\left\{ b_i^k \right\}$. Тогава равенството (2.3.8) приема следната блочна форма:

$$\mathbf{x}^k = \left(L^{k,k} \right)^{-1} \left(\mathbf{b}^k - \sum_{l=1}^{k-1} L^{l,k} \mathbf{x}^l \right), \quad \text{за } k\text{=1,...,m.} \tag{2.3.10}$$

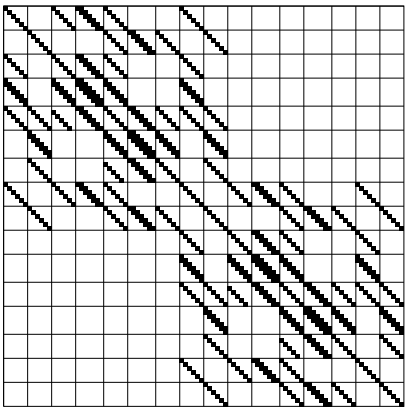
Ако даден блок $L^{k,k}$ е диагонална матрица, то елементите $\left\{ x_i^k \right\}_{i=1}^{N^k}$ могат да бъдат намерени по следния начин:

$$x_i^k = \frac{1}{l_{i,i}^{k,k}} \left(b_i^k - \sum_{l=1}^{k-1} \sum_{j=1}^{N^l} l_{i,j}^{l,k} x_j^l \right). \tag{2.3.11}$$

От горното равенство се вижда, че x_i^k не зависи от никое x_j^k, $j \neq i$. Това значи, че всичките елементи на частта от решението $\mathbf{x}^k$ могат да се намерят едновременно, независимо един от друг. По този начин могат да бъдат използвани N^k различни процесори.

На Фиг 2.1 е изобразена структурата на ненулевите елементи на матрица на коравина A. Използвана е лексикографска номерация на възлите относно координатите им. Матрицата може да се раздели на блокове, както е показано на фигурата. Вижда се, че част от блоковете по диагонала са диагонални матрици, а останалите са *почти* диагонални.

Решаването на системи с MIC(0) преобусловители обикновено се смята за последователен процес. Това е така поради рекурентния характер на процеса на решаване на триъгълните системи, които участват във факторизацията. Както показахме по-горе, ако матрицата която факторизираме има такава блочна структура, че диагоналните блокове са диагонални матрици, този недостатък може да бъде преодолян.



Фигура 2.1: Структура на ненулевите елементи на матрицата на коравина A, за разделяне на Ω на $2 \times 2 \times 6$. Ненулевите елементи са изобразени с малки квадратчета.

Идеята на нашия подход е да приложим MIC(0) факторизацията към спомагателна матрица B, която има специална блочна структура. Целта е тази структура да позволява ефективна паралелна реализация. Идеята за това, черпим от опита за паралелни MIC(0) алгоритми за двумерни задачи (виж [11, 12, 38, 49]). Нашият алгоритъм, както и тези в двумерния случай са вдъхновени от структурата на матрицата на коравина, получена при дискретизация с елементи на Ранахер и Турек. Алгоритъмът се базира на модификации на елементните матрици на коравина, (виж [37]).

2.3.2 Локално оптимизирани паралелни MIC(0) преобусловители

Ще конструираме спомагателната матрицата B на елементно ниво. Следвайки стандартната процедура в МКЕ за асемблиране, матрицата на коравина A може да бъде записана по следния начин:

$$A = \sum_{e \in \mathcal{T}} T_e^T A_e T_e, \tag{2.3.12}$$

където A_e е елементната матрица на коравина, а T_e е матрицата изобразяваща глобалния вектор на неизвестните в локалния, за текущия елемент e. Матрицата A_e е плътна и може да бъде записана в следния вид:

$$A_e = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix},$$

където номерацията и шаблона на свързаност са изобразени на Фиг. 1.1.

Лема 2.3.1. *Нека въведем спомагателната матрица B в следния вид:*

$$B = \sum_{e \in \mathcal{T}} \lambda_e^{(1)} T_e^T B_e T_e, \tag{2.3.13}$$

където B_e е симетрична и положително полуопределена матрица с неположителни извъндиагонални елементи, такава че $B_e\mathbf{e} = A_e\mathbf{e}$, $\mathbf{e}^T = (1,1,1,1,1,1)$, и където $\{\lambda_e^{(i)}\}_{i=1}^5$ са ненулевите обобщени собствени стойности на задачата $A_e\mathbf{v} = \lambda B_e\mathbf{v}$ подредени в нарастващ ред. Тогава:

(i) матрицата B удовлетворява условията на Теорема 1.4.1 за устойчива MIC(0) факторизация;

(ii) B сила е следната локална оценка на числото на обусловеност,

$$\kappa(B^{-1}A) \leq \max_e \kappa(B_e^{-1}A_e).$$

Доказателство. Доказателството на (i) следва директно от конструирането на B, при лексикографска глобална номерация на възлите. За второто твърдение, нека $\mathbf{v} \in \mathbb{R}^N$, N е размера на глобалния вектор от неизвестните и $\mathbf{v}_e \in \mathbb{R}^6$ е рестрикцията на $\mathbf{v}$ върху текущия елемент $e \in \mathcal{T}$. Тогава,

$$(B\mathbf{v}, \mathbf{v}) = \sum_e \lambda_e^{(1)} (B_e\mathbf{v}_e, \mathbf{v}_e) \leq \sum_e (A_e\mathbf{v}_e, \mathbf{v}_e) = (A\mathbf{v}, \mathbf{v}),$$

следователно за минималната собствена стойност λ_m е в сила неравенството

$$\lambda_m(B^{-1}A) \geq 1.$$

Аналогично,

$$\begin{aligned}
(B\mathbf{v}, \mathbf{v}) &= \sum_e \lambda_e^{(1)} (B_e\mathbf{v}_e, \mathbf{v}_e) \\
&\geq \sum_e \frac{\lambda_e^{(1)}}{\lambda_e^{(5)}} (A_e\mathbf{v}_e, \mathbf{v}_e) \\
&\geq \min_e \frac{\lambda_e^{(1)}}{\lambda_e^{(5)}} \sum_e (A_e\mathbf{v}_e, \mathbf{v}_e) \\
&= \min_e \frac{\lambda_e^{(1)}}{\lambda_e^{(5)}} (A\mathbf{v}, \mathbf{v})
\end{aligned} \tag{2.3.14}$$

и следователно за максималната собствена стойност на $B^{-1}A$ е в сила оценката

$$\lambda_M(B^{-1}A) \leq \max_e \frac{\lambda_e^{(5)}}{\lambda_e^{(1)}},$$

с което доказателството е завършено. □

Сега ще представим структурата на два варианта за локална апроксимация B_e, при условие, че са спазени условията на Лема 2.3.1.

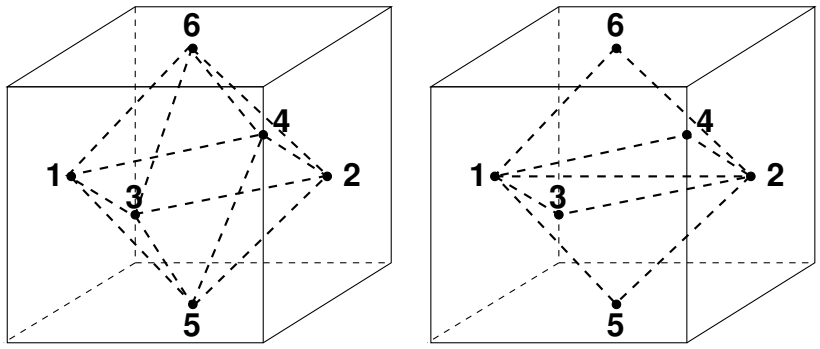
Вариант 1:

$$B_e^1 = \begin{bmatrix} b_{11} & & b_{13} & b_{14} & b_{15} & b_{16} \\ & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & & b_{35} & b_{36} \\ b_{41} & b_{42} & & b_{44} & b_{45} & b_{46} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & \\ b_{61} & b_{62} & b_{63} & b_{64} & & b_{66} \end{bmatrix} \tag{2.3.15}$$

Вариант 2:

$$B_e^2 = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & & & \\ b_{41} & b_{42} & & b_{44} & & \\ b_{51} & b_{52} & & & b_{55} & \\ b_{61} & b_{62} & & & & b_{66} \end{bmatrix} \tag{2.3.16}$$

Дефинициите на въведените апроксимации B_e съответстват на локалната номерация, показана на Фиг. 2.2. Тук с пунктирани линии е изобразен шаблона на свързаност за Вариант 1 (вляво) и Вариант 2 (вдясно). Нека отбележим, че диагонална матрица B_e не би могла да удовлетвори условието $B_e\mathbf{e} = A_e\mathbf{e}$.



(а) (б)

Фигура 2.2: Локална номерация и схема на свързаност на степените на свобода на матриците B_e: (а) Вариант 1; (б) Вариант 2.

С помощта на въведената спомагателна матрица B, дефинираме паралелен MIC(0) преобусловител C за матрицата A във вида

$$C = C_{\mathrm{MIC(0)}}(B).$$

Нека припомним, че дефиницията на B осигурява устойчива MIC(0) факторизация. В следващия раздел ще бъдат представени локално оптимизирани конструкции на матрицата B_e за двата Варианта 1 и 2 и за двата вида елементи на Ранахер-Турек – MV и MP.

Структурата на ненулевите елементи на B_e, ще позволи паралелизация на метода на спрегнатия градиент с представения преобусловител. Помощната матрица B има специална блочна структура, в която блоковете по диагонала

са диагонални матрици. Те съответстват на линии и равнини, определени от възлите на мрежата.

Нека разгледаме задачата (2.2.4) в случая на коефициентна и мрежова изотропия, тоест, в случая, когато всички елементи $e \in \mathcal{T}$ са кубове със страна h и локалните коефициентни матрици $a(e)$ са диагонални, с равни елементи по диагонала:

$$a(e) = \begin{bmatrix} a_e & & \\ & a_e & \\ & & a_e \end{bmatrix}$$

Тогава, елементните матрици на коравина за елементите на Ранахер и Турек са както следва:

$$A_e^{MP} = \frac{2ha_e}{9} \begin{bmatrix} 17 & -1 & -4 & -4 & -4 & -4 \\ -1 & 17 & -4 & -4 & -4 & -4 \\ -4 & -4 & 17 & -1 & -4 & -4 \\ -4 & -4 & -1 & 17 & -4 & -4 \\ -4 & -4 & -4 & -4 & 17 & -1 \\ -4 & -4 & -4 & -4 & -1 & 17 \end{bmatrix} \tag{2.3.17}$$

в случая на MP дискретизация и

$$A_e^{MV} = 2ha_e \begin{bmatrix} 3 & 1 & -1 & -1 & -1 & -1 \\ 1 & 3 & -1 & -1 & -1 & -1 \\ -1 & -1 & 3 & 1 & -1 & -1 \\ -1 & -1 & 1 & 3 & -1 & -1 \\ -1 & -1 & -1 & -1 & 3 & 1 \\ -1 & -1 & -1 & -1 & 1 & 3 \end{bmatrix} \tag{2.3.18}$$

в случая на MV дискретизация.

Ще конструираме елементни матрици изпълняващи условията на Лема 2.3.1, такива че относителните числа на обусловеност

$$\kappa\left(\left(B_e^{MP}\right)^{-1} A_e^{MP}\right) \text{ и } \kappa\left(\left(B_e^{MV}\right)^{-1} A_e^{MV}\right)$$

са минимални. Следните две леми ще ни послужат за анализа.

Лема 2.3.2. *Нека A и B са симетрични и положително полуопределени матрици, P е симетрична пермутационна матрица, за която $P^TP = P^2 = I$, а $\mathcal{M}$ е класът матрици с неположителни извъндиагонални елементи, за които пермутационната матрица P не променя нулевите елементи. Тоест за всяка матрица M от $\mathcal{M}$ е вярно, че M и PMP имат нулеви елементи на едни и същи места. Нека също така:*

(а) $\ker(A) = \ker(B)$*;*

(б) $PAP = A$*;*

(в) B е матрицата с минимално относително число на обусловеност $\kappa(B^{-1}A) = \kappa_m$ в класа на матриците $\mathcal{M}$.

Тогава съществува матрица $\tilde{B}$ *от класа* $\mathcal{M}$*, такава, че* $P\tilde{B}P = \tilde{B}$ *и* $\kappa(\tilde{B}^{-1}A) = \kappa_m$.

Доказателство. Нека $A\mathbf{v} = \lambda B\mathbf{v}$, тогава от допускането, че $PAP = A$ следва $A\hat{\mathbf{v}} = \lambda(PBP)\hat{\mathbf{v}}$, $\hat{\mathbf{v}} = P\mathbf{v}$. Нека означим с λ_m и λ_M минималната и максималната собствена стойност на обобщената задача за собствени стойности

$$A\mathbf{v} = \lambda B\mathbf{v}, \qquad \mathbf{v} \notin \ker(A).$$

Тогава от обобщената мин-макс теорема на Курант-Фишер [5] следват неравенствата

$$\begin{aligned}\lambda_m B \leq & A \leq \lambda_M B,\\ \lambda_m PBP \leq & A \leq \lambda_M PBP,\\ \lambda_m \tilde{B} \leq & A \leq \lambda_M \tilde{B},\end{aligned}$$

където

$$\tilde{B} = \frac{B + PBP}{2}\ .$$

От последното неравенство следва, че

$$\kappa(\tilde{B}^{-1}A) = \frac{\lambda_M}{\lambda_m} = \kappa_m.$$

Тук неравенствата трябва да се разбират в смисъл на положителна полуопределеност на матриците. Тоест под $A \leq B$ разбираме, че неравенството

$$\mathbf{v}^T A\mathbf{v} \leq \mathbf{v}^T B\mathbf{v}$$

е изпълнено за всеки вектор $\mathbf{v}$.

Тъй като извъндиагоналните елементи на $\tilde{B}$ са неположителни, за завършване на доказателството остава да отбележим, че:

$$P\tilde{B}P = \frac{PBP + PPBPP}{2} = \frac{B + PBP}{2} = \tilde{B}.$$

□

Лема 2.3.3. *Нека*

$$A_e = \begin{bmatrix} 2b_{12}+a_1 & -a_1 & -b_1 & -b_1 & -b_2 & -b_2 \\ -a_1 & 2b_{12}+a_1 & -b_1 & -b_1 & -b_2 & -b_2 \\ -b_1 & -b_1 & 2b_{13}+a_2 & -a_2 & -b_3 & -b_3 \\ -b_1 & -b_1 & -a_2 & 2b_{13}+a_2 & -b_3 & -b_3 \\ -b_2 & -b_2 & -b_3 & -b_3 & 2b_{23}+a_3 & -a_3 \\ -b_2 & -b_2 & -b_3 & -b_3 & -a_3 & 2b_{23}+a_3 \end{bmatrix}, \tag{2.3.19}$$

където $b_{ij} = b_i + b_j$. *Тогава*

$$A_e V_e = V_e L_e, \tag{2.3.20}$$

където

$$V_e = \begin{bmatrix} 1 & 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 2 & -1 & 0 & 0 \\ 1 & \sqrt{3} & -1 & 0 & 1 & 0 \\ 1 & \sqrt{3} & -1 & 0 & -1 & 0 \\ 1 & -\sqrt{3} & -1 & 0 & 0 & 1 \\ 1 & -\sqrt{3} & -1 & 0 & 0 & -1 \end{bmatrix}$$

и

$$L_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_{12}+4b_3 & \sqrt{3}(b_2-b_1) & 0 & 0 & 0 \\ 0 & \sqrt{3}(b_2-b_1) & 3b_{12} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(b_{12}+a_1) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2(b_{13}+a_2) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2(b_{23}+a_3) \end{bmatrix}.$$

Доказателство. Лемата се доказва, чрез непосредствена проверка. □

Забележка 2.3.1. Вторият и третият вектор-стълб на V_e са избрани така, че извънднагоналните елементи на L_e да се анулират при $b_1 = b_2$. Тогава, прилагайки лемата, за елементните матрици A_e получаваме собствените стойности на A_e в диагонала на L_e.

За елементната матрица A_e^{MP}, като положим

$$b_1 = b_2 = b_3 = 4\frac{2ha_e}{9} \quad \text{и} \quad a_1 = a_2 = a_3 = \frac{2ha_e}{9}$$

получаваме

$$A_e^{MP}V_e = V_e\, L_e^{MP} \qquad L_e^{MP} = \frac{2ha_e}{9}\begin{bmatrix} 0 & & & & & \\ & 24 & & & & \\ & & 24 & & & \\ & & & 18 & & \\ & & & & 18 & \\ & & & & & 18 \end{bmatrix}. \tag{2.3.21}$$

За матрицата A_e^{MV} прилагаме Лема 2.3.3 полагайки

$$b_1 = b_2 = b_3 = 2ha_e \quad \text{и} \quad a_1 = a_2 = a_3 = -2ha_e$$

и получаваме

$$A_e^{MV}V_e = V_e\, L_e^{MV} \qquad L_e^{MV} = 2ha_e\begin{bmatrix} 0 & & & & & \\ & 6 & & & & \\ & & 6 & & & \\ & & & 2 & & \\ & & & & 2 & \\ & & & & & 2 \end{bmatrix}. \tag{2.3.22}$$

2.4 Локален анализ на спектралното число на обусловеност

От Лема 2.3.2 следва, че при конструиране на локално оптимални приближения на A_e с определена структура на ненулевите елементи е достатъчно да разгледаме такива матрици B_e, които запазват изотропията на A_e.

Нека разгледаме пермутационните матрици

$$P_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

$$P_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Лесно се проверява, че са изпълнени равенствата:

$$P_i^T A_e^{MP} P_i = A_e^{MP} \quad \text{и} \quad P_i^T A_e^{MV} P_i = A_e^{MV} \quad \text{за } i = 1, \ldots, 5. \tag{2.4.23}$$

Нека разгледаме симетричната матрицата B_e

$$B_e = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} \end{bmatrix}. \tag{2.4.24}$$

Ако е изпълнено равенството $P_1^T B_e P_1 = B_e$, то от симетрията на B_e следва, че са в сила равенствата:

$$\begin{aligned} b_{11} = b_{22}, \quad & b_{13} = b_{23} = b_{31} = b_{32}, \\ & b_{14} = b_{24} = b_{41} = b_{42}, \\ & b_{15} = b_{25} = b_{51} = b_{52}, \\ & b_{16} = b_{26} = b_{61} = b_{62}. \end{aligned} \tag{2.4.25}$$

Аналогично, от допусканията $P_2^T B_e P_2 = B_e$, $P_3^T B_e P_3 = B_e$, $P_4^T B_e P_4 = B_e$ и $P_5^T B_e P_5 = B_e$ следват съответно равенствата:

$$\begin{aligned} b_{33} = b_{44}, \quad & b_{31} = b_{41} = b_{13} = b_{14}, \\ & b_{32} = b_{42} = b_{23} = b_{24}, \\ & b_{35} = b_{45} = b_{53} = b_{54}, \\ & b_{36} = b_{46} = b_{63} = b_{64}; \end{aligned} \tag{2.4.26}$$

$$\begin{aligned} b_{55} = b_{66}, \quad & b_{53} = b_{63} = b_{35} = b_{36}, \\ & b_{54} = b_{64} = b_{45} = b_{46}, \\ & b_{51} = b_{61} = b_{15} = b_{16}, \\ & b_{52} = b_{62} = b_{25} = b_{26}; \end{aligned} \tag{2.4.27}$$

$$\begin{aligned} b_{33} = b_{55}, \quad & b_{13} = b_{15} = b_{31} = b_{51}, \\ b_{44} = b_{66}, \quad & b_{14} = b_{16} = b_{41} = b_{61}, \\ b_{34} = b_{56}, \quad & b_{23} = b_{25} = b_{32} = b_{52}, \\ b_{43} = b_{65}, \quad & b_{24} = b_{26} = b_{42} = b_{62}; \end{aligned} \tag{2.4.28}$$

и

$$\begin{aligned} b_{11} = b_{55}, \quad & b_{13} = b_{53} = b_{31} = b_{53}, \\ b_{22} = b_{66}, \quad & b_{23} = b_{63} = b_{32} = b_{36}, \\ b_{12} = b_{56}, \quad & b_{14} = b_{54} = b_{41} = b_{45}, \\ b_{21} = b_{65}, \quad & b_{24} = b_{64} = b_{42} = b_{64}. \end{aligned} \tag{2.4.29}$$

Трансформация с матриците P_i, $i = 1, \ldots, 4$ запазва структурата на ненулевите елементи на матриците B_e^1 (2.3.15) и B_e^2 (2.3.16). За Вариант 1 структурата се запазва и от трансформация с пермутационната матрица P_5.

Вариант 1:

Разглеждаме матрицата B_e^1 от (2.3.15). От Лема 2.3.2 и равенствата (2.4.25)—(2.4.29) следва, че всички извъндиагонални елементи на матрицата са равни. Тогава, като вземем пред вид условието $B_e^1 \mathbf{e} = A_e \mathbf{e} = \mathbf{0}$ получаваме следния вид за локално оптималната матрица B_{opt}^1:

$$B_{opt}^1 = \begin{bmatrix} 4b & & -b & -b & -b & -b \\ & 4b & -b & -b & -b & -b \\ -b & -b & 4b & & -b & -b \\ -b & -b & & 4b & -b & -b \\ -b & -b & -b & -b & 4b & \\ -b & -b & -b & -b & & 4b \end{bmatrix}, \qquad b > 0. \tag{2.4.30}$$

Лема 2.4.1. *За Вариант 1 матрицата B_{opt}^1 е локално оптимално приближение на A_e^{MP} и A_e^{MV}. За всяко положително b е в сила оценката*

$$\kappa\left(\left(B_{opt}^1\right)^{-1} A_e^{MP}\right) = 9/8, \quad \textit{and} \quad \kappa\left(\left(B_{opt}^1\right)^{-1} A_e^{MV}\right) = 2. \tag{2.4.31}$$

Доказателство. Прилагаме Лема 2.3.3 при $a_1 = a_2 = a_3 = 0$ и $b_1 = b_2 = b_3 = b$ и получаваме

$$B^1_{opt}V_e = V_e \begin{bmatrix} 0 & & & & & \\ & 6b & & & & \\ & & 6b & & & \\ & & & 4b & & \\ & & & & 4b & \\ & & & & & 4b \end{bmatrix}, \qquad b > 0,$$

Оттук, като вземем предвид (2.3.21), ненулевите собствените стойности на обобщената задачата за собствени стойности $A^{MP}_e\mathbf{v} = \lambda B^1_{opt}\mathbf{v}$ (с точност до множител $2ha_e/9$) са:

$$\frac{4}{b}, \frac{4}{b}, \frac{9}{2b}, \frac{9}{2b}, \frac{9}{2b}$$

и следователно

$$\kappa(A^{MP}_e, B^1_{opt}) = 9/8$$

.

Аналогично, собствените стойности на задачата $A^{MV}_e\mathbf{v} = \lambda B^1_{opt}\mathbf{v}$ (с точност до множител $2ha_e$) са

$$\frac{1}{b}, \frac{1}{b}, \frac{1}{2b}, \frac{1}{2b}, \frac{1}{2b}.$$

Тогава получаваме

$$\kappa(A^{MV}_e, B^1_{opt}) = 2$$

. □

Вариант 2:

Разглеждаме матрицата B^2_e от (2.3.16). От Лема 2.3.2 и (2.4.25)—(2.4.28) следват равенствата:

$$\begin{gathered} b_{1,3} = b_{1,4} = b_{1,5} = b_{1,6} = \\ b_{2,3} = b_{2,4} = b_{2,5} = b_{2,6} = \\ b_{3,1} = b_{4,1} = b_{5,1} = b_{6,1} = \\ b_{3,1} = b_{4,1} = b_{5,1} = b_{6,1} \\ \text{и} \\ b_{1,2} = b_{2,1}. \end{gathered}$$

От горните равенства и от условието $B^2_e\mathbf{e} = A_e\mathbf{e} = \mathbf{0}$ получаваме матрицата B^2_{opt} в следния вид:

$$B^2_{opt} = \begin{bmatrix} 4b+a & -a & -b & -b & -b & -b \\ -a & 4b+a & -b & -b & -b & -b \\ -b & -b & 2b & & & \\ -b & -b & & 2b & & \\ -b & -b & & & 2b & \\ -b & -b & & & & 2b \end{bmatrix}, \qquad a \geq 0, \quad b > 0. \qquad (2.4.32)$$

В сила е следната лема:

Лема 2.4.2. *За Вариант 2 локално оптималните приближения на матриците A_e^{MP} и A_e^{MV} имат вида на матрицата B_{opt}^2 в (2.4.32), като са в сила следните равномерни оценки на числото на обусловеност:*

$$\kappa\left(\left(B_e^{MP}\right)^{-1} A_e^{MP}\right) \leq 3 \quad и \quad \kappa\left(\left(B_e^{MV}\right)^{-1} A_e^{MV}\right) \leq 6. \tag{2.4.33}$$

Доказателство. Локално оптималното приближение B_{opt}^2 съответства на общия вид на матрицата (2.3.19) при $b_1 = b_2 = b$, $a_1 = a$ и $b_3 = a_2 = a_3 = 0$. Като приложим Лема (2.3.3) получаваме

$$B_{opt}^2 V_e = V_e \begin{bmatrix} 0 & & & & & & \\ & 2b & & & & & \\ & & 6b & & & & \\ & & & 4b+2a & & & \\ & & & & 2b & & \\ & & & & & 2b & \end{bmatrix}. \tag{2.4.34}$$

Стойностите b и a се определят така, че числото на обусловеност $\kappa((B_{opt}^2)^{-1} A_e^{MP})$ да е минимално. Собствените стойности на задачата $A_e^{MP}\mathbf{v} = \lambda B_{opt}^2 \mathbf{v}$ (с точност до множител $2ha_e/9$) имат вида:

$$\frac{12}{b}, \frac{9}{a+2b}, \frac{4}{b}, \frac{9}{b}, \frac{9}{b}. \tag{2.4.35}$$

Следователно

$$\kappa((B_{opt}^2)^{-1} A_e) = \max\left(3, \frac{4(a+2b)}{3b}\right) = \max\left(3, \frac{8}{3} + \frac{4a}{3b}\right). \tag{2.4.36}$$

Ако изберем $a = 0$ получаваме, че числото на обусловеност е равно на 3 за всяко $b > 0$. С това доказателството на лемата за случая MP е завършено.

По аналогичен начин изследваме случая MV. Собствените стойности λ на задачата $A_e^{MV}\mathbf{v} = \lambda B_{opt}^2 \mathbf{v}$ (с точност до множител $2ha_e$) са

$$\frac{3}{b}, \frac{1}{b}, \frac{1}{2b+a}, \frac{1}{b}, \frac{1}{b}.$$

Следователно,

$$\kappa((B_{opt}^2)^{-1} A_e^{MV}) = \max\left(3, \frac{3(a+2b)}{b}\right) = 6 + \frac{3a}{b}.$$

Така при $a = 0$ получаваме минимум $\kappa(B_e^{-1} A_e^{MV}) = 6$ за всяко положително b. С това лемата е доказана. □

Вече можем да докажем следната теорема:

Теорема 2.4.1. *Нека B^{MP} и B^{MV} са дефинирани чрез (2.3.13), където локалните (елементни) матрици B_e са равни на B_{opt}^1 или B_{opt}^2. Тогава са в сила следните оценки.*

$$\kappa\left(\left(B^{MP}\right)^{-1} A^{MP}\right) \leq 3, \qquad \kappa\left(\left(B^{MV}\right)^{-1} A^{MV}\right) \leq 6.$$

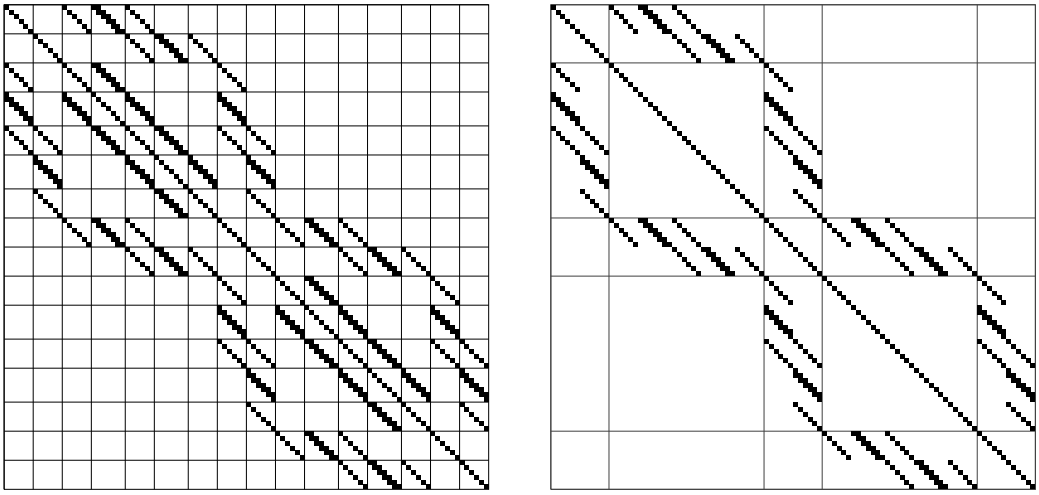
Тези оценки са равномерни, както по отношение на параметъра на дискретизация h, така и спрямо скокове на коефициента $a(e)$.

Доказателство. Вярността на теоремата следва от локалните оценки в Лема 2.4.1 и Лема 2.4.2 след прилагане на Лема 2.3.1. □

2.5 Паралелна реализация

2.5.1 Описание на алгоритъма

За решаване на системата от линейни уравнения (2.2.6) използваме метода на спрегнатия градиент с преобусловител MIC(0) факторизацията на B. Нека предположим, че паралелепипедалната област Ω е разделена на $n_1 \times n_2 \times n_3$ шестостенни елемента. И за двата варианта на дискретизацията степените на свобода са асоциирани със стените на шестостените. За определеност, от тук нататък всички разглеждания са за случая MP.



(а) (б)

Фигура 2.3: Структура на ненулевите елементи на матрицата B, за разделяне на Ω на $2 \times 2 \times 6$ елемента: (а) Вариант 1; (б) Вариант 2. Ненулевите елементи са изобразени с малки квадратчета.

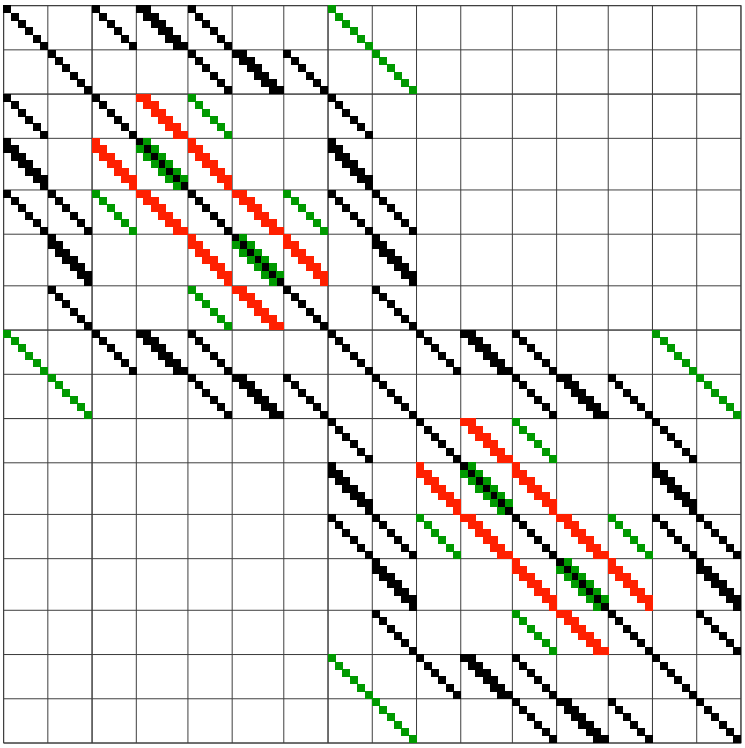
Структурата на ненулевите елементи на матрицата B е показана на Фиг. 2.3 и за двата Варианта - 1 и 2. Използвана е лексикографска номерация на възлите. За сравнение, на Фиг. 2.4 е показана структурата на оригиналната матрица A. Нека отбележим, че и в двата случая блоковете стоящи по диагонала на матрицата B са диагонални. Размерите на тези блокове варират. Те са съответно n_3 или $(n_3 + 1)$ за Вариант 1 и $n_2 n_3$ или $n_2(n_3 + 1) + (n_2 + 1)n_3$ за Вариант 2.

За решаването на системи с MIC(0) преобусловителя, тоест

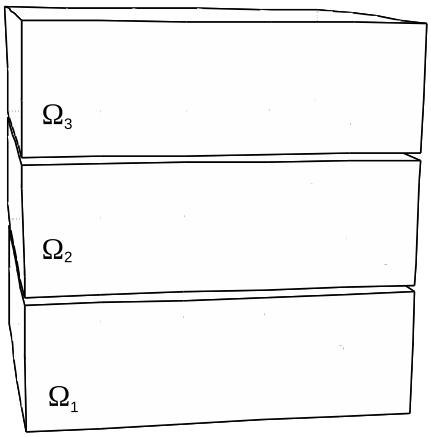
$$C_{MIC(0)}(B)\mathbf{w} \equiv (X - L)\, X^{-1} \left(X - L^T\right) \mathbf{w} = \mathbf{v} \tag{2.5.37}$$

трябва да се решат системите

$$\begin{aligned} \tilde{L}\mathbf{y} \equiv (X - L)\mathbf{y} &= \mathbf{v}, \\ X^{-1}\mathbf{z} &= \mathbf{y} \\ \text{и} & \\ \tilde{L}^T\mathbf{w} &= \mathbf{z}, \end{aligned}$$



Фигура 2.4: Структура на ненулевите елементи на матрицата A, за разделяне на Ω на $2 \times 2 \times 6$ елемента. Със зелено са означени елементите, които отпадат при Варианти 1 и 2, а с червено – тези които отпадат само при Вариант 2.



Фигура 2.5: Разделяне на областта Ω на подобласти при използване на 3 процесора.

където L е строго долно триъгълната част от матрицата B. Триъгълните системи се решават както в обратния ход в метода на Гаус. Това може да стане на $k_{B1} = n_1(2n_2 + 1) + n_2$ стъпки за Вариант 1 и на $k_{B2} = 2n_1 + 1$ стъпки за Вариант 2.

Да разгледаме решаването на долнотриъгълната система. Записваме я в

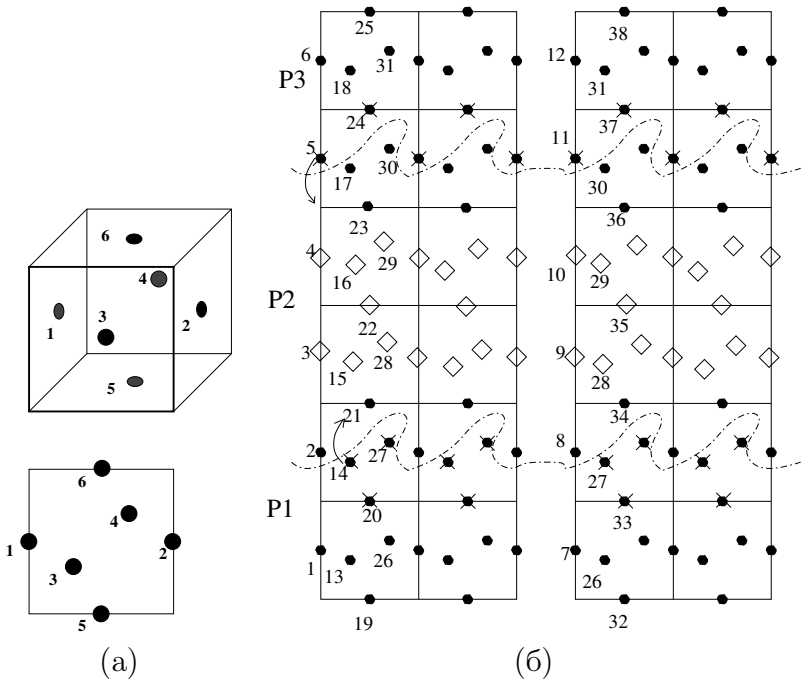
блочен вид, така че блоковете $\tilde{L}_{ii}$ са диагонални матрици:

$$\begin{bmatrix} \tilde{L}^{1,1} & \mathbf{0} & \cdots & \mathbf{0} \\ \tilde{L}^{2,1} & \tilde{L}^{2,2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{L}^{k,1} & \tilde{L}^{k,2} & \cdots & \tilde{L}^{k,k} \end{bmatrix} \begin{bmatrix} \mathbf{y}^1 \\ \mathbf{y}^2 \\ \vdots \\ \mathbf{y}^k \end{bmatrix} = \begin{bmatrix} \mathbf{v}^1 \\ \mathbf{v}^2 \\ \vdots \\ \mathbf{v}^k \end{bmatrix}. \tag{2.5.38}$$

На i-тата стъпка се намира блокът $\mathbf{y}^i$. Тъй като блоковете $\tilde{L}_{ii}$, са диагонални, изчисленията за всеки елемент $\mathbf{y}^i$ могат да се пресметнат паралелно.

Нека имаме $p \leq n_3 + 1$ процесора, означени с $P_i|_{i=1}^{p}$. Разделяме областта Ω на p на брой подобласти $\Omega_i|_{i=1}^{p}$, както е показано на Фиг. 2.5. Всеки процесор съхранява част от елементите на векторите, участващи в метода на спрегнатия градиент с преобуславяне, съответстваща на възли от мрежата от определена подобласт. По този начин всеки вектор се разпределя по всички процесори. При Вариант 1 всеки блок $\mathbf{y}^i$ съответства на една вертикална линия от възли на мрежата, а при Вариант 2 – на възли разположени в една равнина.

На Фиг. 2.6 (б) е изобразено разпределението на възлите от мрежата върху 3 процесора. За целта е използвана двумерна проекция на всеки елемент, виж. Фиг. 2.6 (а). По същия начин са разпределени и елементите от всички вектори и редовете на всички матрици, които участват в алгоритъма на МСГП. Разпределението е едно и също и за двата варианта.



Фигура 2.6: (а) Двумерна проекция на референтния елемент. (б) Разпределение на данните: $p = 3$, $n_1 = 2$, $n_2 = 2$, $n_3 = 6$; Схема на комуникациите за умножението на матрица по вектор.

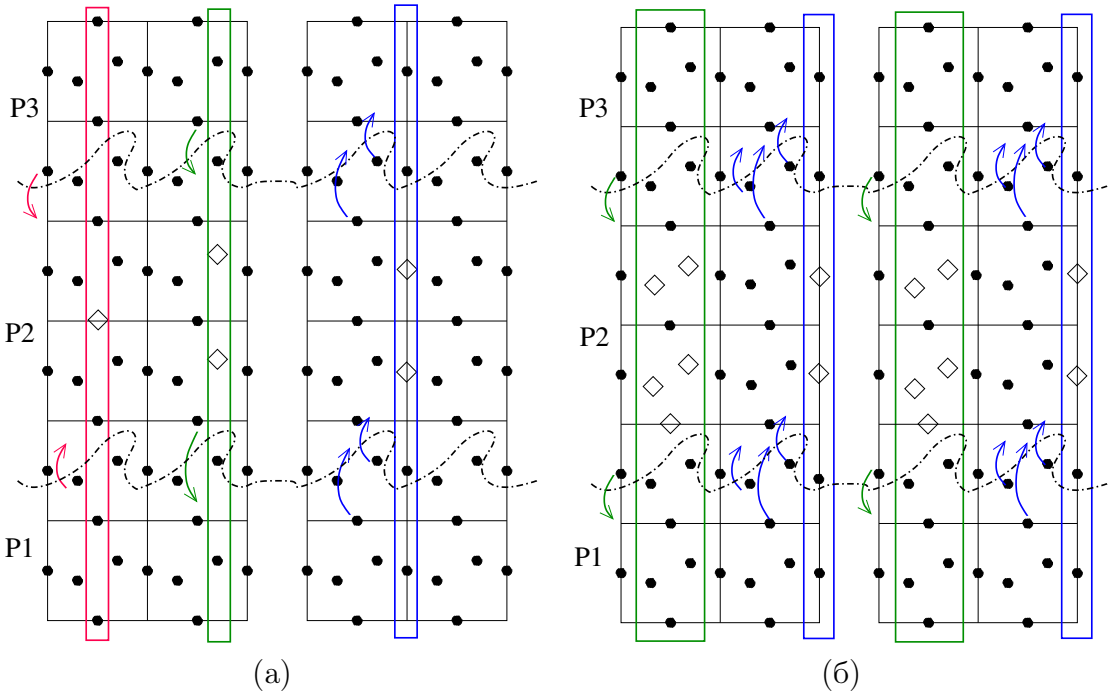
Ще опишем на кратко как се извършват различните операции в МСГП и какви комуникации са необходими за изпълнението им.

На всяка итерация, освен решаването на системата с преобуславящата матрица $C_{MIC(0)}(B)$, трябва да бъдат извършени едно умножение на матрицата на коравина A с вектор, две скаларни произведения, и три операции от типа

$$\mathbf{v} := \alpha\mathbf{v} + \mathbf{u}.$$

За свързаните векторни операции, всеки процесор пресмята своята част от вектора $\mathbf{v}$. Не се налагат никакви комуникации. За пресмятане на скаларните произведения, след пресмятането на частите, съответстващи на локалните вектори, се налага изпълнението на една глобална комуникация за получаването на сумата от всичките процесори.

Нека отбележим, че възлите лежащи върху равнините разделящи подобластите Ω_i и Ω_{i+1} са разпределени по равно между процесорите P_i и P_{i+1} (виж. Фиг. 2.6 (б)). За получаване на елементите от умножението на матрица по вектор $A\mathbf{v}$, за които процесорът P_i е отговорен, е необходимо прехвърлянето на някои елементи от вектора $\mathbf{v}$ от съседните нему процесори P_{i-1} и P_{i+1}. Броят на тези елементи е $2n_1n_2+n_1$ прехвърляни от процесора P_{i-1} и $2n_1n_2+n_2$ – прехвърляни от процесора P_{i+1}. На Фиг. 2.6 (б), елементите, които трябва да се прехвърлят в процесора P_2 са означени със знака $\times$. Докато са в ход тези комуникации могат да бъдат пресмятани компонентите на резултата $A\mathbf{v}$, които не зависят от тези на $\mathbf{v}$, разположени в съседните процесори. Тези елементи са означени със знака $\diamond$.



Фигура 2.7: Схема на комуникациите, при решаване на система с долнотриъгълните матрици от преобусловителите: (а) Вариант 1; (б) Вариант 2.

Нека се върнем към решаването на системата с преобусловителя (2.5.37). Решаването на системи с диагонална матрица е тривиално и може да се извърши без никакви комуникации. Решаването на долнотриъгълните системи може

да бъде разделено на k_{B1} стъпки за Вариант 1 и на k_{B2} – за Вариант 2. На i-тата стъпка се пресмята частта $\mathbf{y}^i$ от решението. За изчисляването на всяка стъпка е необходимо прехвърлянето на определени компоненти от съседните процесори.

Три различни шаблона на тези комуникации са необходими за Вариант 1 и два – за Вариант 2. За решаването на долнотриъгълна система тези шаблони са изобразени на Фиг. 2.7. Различните шаблони са изобразени чрез различни цветове. Правоъгълниците с еднакъв цвят ограждат елементите, които трябва да се пресметнат на определена стъпка. Комуникациите са обозначени с цветни стрелки. Те трябва да бъдат завършени преди да се извършат изчисленията за съответния правоъгълник от възли. За Варианта 1 е достатъчно прехвърлянето на една или две компоненти между всяка двойка съседни процесори в двете посоки за всяка една вертикална линия от възли. За Вариант 2 е необходимо прехвърляне на n_2 или $3n_2 + 2$ елемента между стъпките. Тук пресмятането на елементите означени със знака $\diamond$ може да бъде извършено едновременно с комуникациите.

По аналогичен начин става решаването на системата с горнотриъгълна матрица.

2.5.2 Програмна реализация

Много важен елемент от програмната реализация е структурата от данни за съхраняване на векторите.

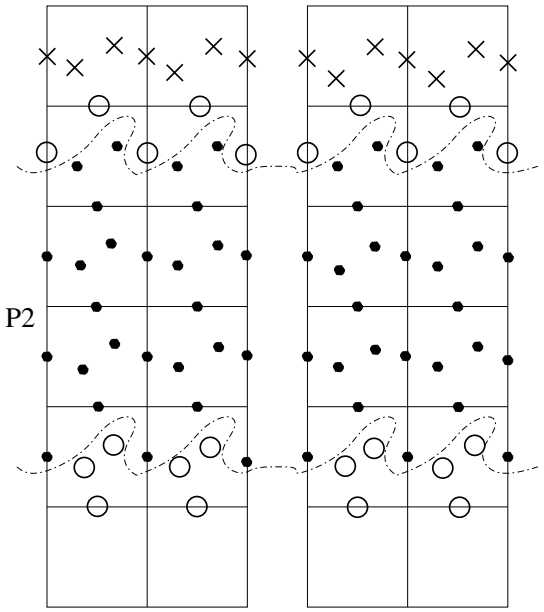
Векторите участващи в МСГП в нашата реализация се съхраняват в четиримерни масиви:

$$\mathbf{v} = \{v^l_{i,j,k}\}.$$

Тук индексите i, j и k определят елемент от крайноелементната мрежа, а последният индекс l = 0,1,2 съответства на възлите разположени съответно на лявата, предната и долната стена от крайния елемент.

При разделяне на мрежата на $n_1 \times n_2 \times n_3$ елемента индексите i и j се менят съответно от 0 до n_1 и от 0 до n_2. По границата на областта, не всички от елементите на масива се използват. По стената с индекс $i = n_1$ се ползва само елемента с индекс $l = 0$, а по стената с индекс $j = n_2$ се ползва само елемента с индекс $l = 1$. По аналогичен начин (виж. Фиг. 2.6 (б)), в най-долния и най-горния слой по направлението с индекс k се използват само част от елементите. Добавени са също така редове, които съответстват на елементи, за чиито изчисления са отговорни съседните два процесора. На тези места се прехвърлят елементи от съседите, при извършването на операциите умножение на матрица по вектор и решаването на системата с преобусловителя. Този подход е често използван в практиката при разработката на паралелни програми. Добавените елементи се наричат ореол (на английски “halo cells” или “ghost cells”). Например, на Фиг. 2.6 (б) процесорът $P2$ е отговорен за изчисленията на елементите с индекси (ще изброим само тези индекси изобразени на фигурата): 2, 3, 4, 8, 9, 10, 15, 16, 17, 21, 22, 23, 28, 29, 30, 34, 35 и 36. Освен тях, има места за елементите с индекси 14, 20, 27, 33, които биват прехвърляни от

процесор P1, и за елементи с индекси 5, 11, 24, 37 от процесор P3. От Фиг. 2.8 се вижда, че за втория процесор индексът k приема 5 стойности - от 0 до 4.



Фигура 2.8: Елементи съхранявани от процесор P2. Със запълнени кръгчета са отбелязани елементите, за чиито пресмятане е отговорен P2; с празни кръгчета са отбелязани, ореолните елементи – тези прехвърляни от съседните процесори; с кръстче са отбелязани "запълващи елементи", които заемат памет, но не се използват.

Матрицата на системата A се съхранява, подобно на векторите, в многомерен масив. Поради симетрията на матрицата, съхраняваме само горната триъгълна част от нея. Добавено е още едно измерение, към четирите измерения използвани за векторите:

$$A = \{a_{i,j,k}^{l,m}\}.$$

Индексите i, j, k и l определят реда в матрицата (по същия начин, както при векторите), а последният индекс $m = 0, \ldots, 5$ определя колоната ($m = 0$ съответства на диагоналния елемент).

Умножението на матрицата по вектор

$$\mathbf{r} = A\mathbf{v}$$

се извършва по следния алгоритъм:

Алгоритъм 2.1. *Умножение на матрица по вектор:*
(1) $\mathbf{r} = \mathbf{0}$
(2) Започни изпращането и получаването на елементи от $\mathbf{v}$ *за и от съседните процесори*
(3) За всички четворки (i,j,k,l), за които изчисленията не зависят от елементите на $\mathbf{v}$ *в съседните процесори:*
(4) $r_{i,j,k}^{l} = r_{i,j,k}^{l} + a_{i,j,k}^{l,0} v_{i,j,k}^{l}$

(5) за m=1, . . . , 5
(6а) намери от Таблица 2.1 по (l, m), индексите (i^0, j^0, k^0, l^0)
(6б) $r^l_{i,j,k} = r^l_{i,j,k} + a^{l,m}_{i,j,k} v^{l^0}_{i+i^0,j+j^0,k+k^0}$
(6в) $r^{l^0}_{i+i^0,j+j^0,k+k^0} = r^{l^0}_{i+i^0,j+j^0,k+k^0} + a^{l,m}_{i,j,k} v^l_{i,j,k}$
(7) Изчакай приключване на комуникациите, започнати в стъпка (2) (8) За останалите четворки (i,j,k,l) изпълни стъпките (4)(5) и (6).

Таблица 2.1: Връзка между колони на матрицата и позиция на елемент

(l, m)	i^0	j^0	k^0	l^0
(0, 1)	0	0	0	1
(0, 2)	0	0	0	2
(0, 3)	0	0	1	2
(0, 4)	0	1	0	1
(0, 5)	1	0	0	0
(1, 1)	0	0	0	2
(1, 2)	0	0	1	2
(1, 3)	0	1	0	1
(1, 4)	1	-1	0	0
(1, 5)	1	0	0	0
(2, 1)	0	0	1	2
(2, 2)	0	0	-1	1
(2, 3)	0	1	0	1
(2, 4)	1	0	-1	0
(2, 5)	1	0	0	0

Таблица 2.1 помага да открием елемента, който отговаря на съответната колона от матрицата A.

Тъй като цикълът (5)-(6) се явява най-вътрешен, скоростта на неговото изпълнение определя скоростта на целия алгоритъм. В началото той беше реализиран както е даден – използвайки таблица. В опити да ускорим цикъла, използвахме switch конструкции за реализиране на търсенето в таблица. Това даде почти същата производителност. След това се възползвахме от средствата на езика C++ за метапрограмиране. Използвайки шаблони, списъци от типове (в които бяха кодирани таблиците) [1] и рекурсия цикълът (5)(6) бе развит (английският термин е unrolled) по време на компилацията от компилатора на C++. В резултат получихме по-ефективна реализация, почти три пъти по-бърза от тази с таблици.

Забележка 2.5.1. Съществено е, че от индексите l и $\{i^0, j^0, k^0, l^0\}$ еднозначно може да се определи отместването на интересуващия ни елемент в паметта, спрямо текущия, което на практика извършва компилатора на C++.

Нека отбележим, че параметърът b в матриците B^1_{opt} и B^2_{opt} може да се избере така, че ненулевите извъндиагонални елементи от тези матрици, скалирани с най-малката собствена стойност, да бъдат същите, като извъндиагоналните

елементи в съответните елементни матрици на коравина. Това означава, че и ненулевите извъндиагонални елементи в глобалната спомагателна матрица B са равни на тези от глобалната матрица на коравина A. В следствие на това, можем да спестим съхранението на извъндиагоналните елементи на B. Диагоналните елементи от B са необходими само за конструирането на преобусловителя, но те могат да бъдат пресметнати от извъндиагоналните, така че и те не се съхраняват.

Системите с MIC(0) преобусловител решаваме на два етапа. Първо се решава системата

$$(L - X)X^{-1}\mathbf{y} = \mathbf{b}, \tag{2.5.39}$$

а след това, системата

$$(L^T - X)\mathbf{x} = \mathbf{y}. \tag{2.5.40}$$

Алгоритъм 2.2 решава системата (2.5.39).

Алгоритъм 2.2. *Решаване на системата* $(L - X)X^{-1}\mathbf{y} = \mathbf{b}$*, Вариант 1. Векторът* $\mathbf{p}$ *съдържа реципрочните стойности на диагоналните елементи от* X

Подпрограма П1(i, j, k, l)
(0)от l *намери в Таблица 2.2 множеството* M*.*
(1а) $y^l_{i,j,k}$ *вече съдържа решението на системата* $(L - X)X^{-1}\mathbf{y} = \mathbf{b}$ *(!)*
(1б) $r = y^l_{i,j,k} * p^l_{i,j,k}$*;* r *е съответният елемент от решението на системата* $(L - X)\mathbf{r} = \mathbf{b}$
(2а) за всяко $m \in M$*:*
(2б) намери от Таблица 2.1 по (l, m), индексите (i^0*, *j^0*, *k^0*, *l^0*)*
(2в) $y^{l^0}_{i+i^0,j+j^0,k+k^0} = y^{l^0}_{i+i^0,j+j^0,k+k^0} - a^{l,m}_{i,j,k} * r$

Подпрограма П2(i, j)
(0) Започни прехвърляне на елементите от и към съседните процесори:
(1) За всички вътрешни индекси k *изпълни П1(i, j, k, 0);*
(2) Изчакай приключване на комуникациите от стъпка (0)
(3) Изпълни П1(i, j, k, 0) за останалите k*.*

Подпрограма П3(i, j)
(0) Започни прехвърляне на елементите от и към съседните процесори:
(1) За всички вътрешни индекси k *изпълни П1(i, j, k, 1);*
(2) Изчакай приключване на комуникациите от стъпка (0)
(3) Изпълни П1(i, j, k, 1) за останалите k*.*

Подпрограма П4(i, j)
(0) Започни прехвърляне на елементите от и към съседните процесори:
(1) За всички вътрешни индекси k *изпълни П1(i, j, k, 2);*
(2) Изчакай приключване на комуникациите от стъпка (0)
(3) Изпълни П1(i, j, k, 2) за останалите k*.*

Главна програма
(0) $\mathbf{y} = \mathbf{b}$*;*
(1) За $i = 0$ *до* $n_1 - 1$
(2а) *За* $j = 0$ *до* $n_2 - 1$
(2б) *Изпълни П2(i, j)*
(3а) *За* $j = 0$ *до* $n_2 - 1$
(3б) *Изпълни П3(i, j)*
(3в) *Изпълни П4(i, j)*
(3г) *Изпълни П3(i,* n_2*)*
(4) За $i = n_1$ *изпълни (2а)(2б)*

Таблица 2.2: Връзки между елементи в матрицата B, Вариант 1

l	M
0	{1, 2, 3, 4}
1	{1, 2, 4, 5}
2	{2, 3, 4, 5}

Таблица 2.3: Връзки между елементи в матрицата B, Вариант 2

l	M
0	{1, 2, 3, 4}
1	{4, 5}
2	{4, 5}

Таблица 2.2 ни дава ненулевите извъндиагонални елементи от матрицата B. Подпрограма П2, П3 и П4 изпращат и получават различни елементи (за това не са обобщени).

За реализирането на комуникациите се конструират със средствата на библиотеката MPI специални типове данни (MPI_Datatype) които определят кои елементи се изпращат/получават. Конструирането на такъв тип от данни става, като се започне от даден базов тип данни. За вградените типове данни на езика C++ има предефинирани MPI типове. Например число с плаваща запетая с единична или двойна точност (MPI_FLOAT и MPI_DOUBLE) или целочислено число (MPI_INT). От тях могат да се съставят други съставни MPI типове, които представляват структури, и вектори. Важно е, че тези типове могат да съдържат и дупки, тоест елементи, които не се пращат или получават. Така, например ако имаме тримерен масив, можем да конструираме MPI типове данни за колона, от възли (три типа, за колона по всяко едно от направленията). После от тези типове за колони, могат да се конструират типове за равнини от възли.

В началото на изпълнение на програмата се създават тези MPI типове данни за различните шаблони на комуникация. А в подпрограмите П2, П3 и

П4 се изпълняват по една (неблокираща) операция изпращане или получаване с подходящ тип, от и на подходящото място.

По подобен на Алгоритъм 2.2 начин се реализира и алгоритъма за решаване на горнотриъгълни системи (2.5.40). Разликата е, че циклите по i и j се изпълняват в обратна посока, и се променя П0.

По аналогичен начин се конструират и алгоритмите за Вариант 2 на преобусловителя. Използва се Таблица 2.3 вместо Таблица 2.2. Освен това обхождането става по равнини от възли, а не по колони, както е при Вариант 1.

Директния подход използващ обикновен (едномерен) масив за векторите *не бе* избран поради няколко причини:

- Това би наложило създаването на друга локална номерация, както и изработването на функции, за преход от локална към глобална номерация и обратното. Това макар възможно и лесно осъществимо, би довело до много излишни преходи от и към глобалните индекси – тоест би забавило изпълнението.

- Както видяхме пазенето на индекси за колоните в матрицата A може да се избегне. Това води до използване на един път и половина по-малко памет за матрицата (ако използваме числа с двойна точност и 32 битови индекси), в сравнение със стандартния компресиран поредов формат.

- Би се загубила важна геометрична информация, което би усложнило извършването на комуникациите.

- Както показахме по-горе, със средствата на езика C++ използването на малки таблици с индекси може да доведе до съществени ускорения.

Забележка 2.5.2. Авторът съзнава, че при тази реализация на векторите се въвежда друга, неявна линейна локална номерация. Тя обаче е по-особена, с това, че номера се приписват и на несъществуващи елементи. Тъй като тази неявна номерация, не е била използвана явно, както и авторът не е обмислял алгоритмите в нейните термини, предпочетохме да не я въвеждаме формално, което би било напълно възможно.

2.6 Оценки за паралелните времена

В този раздел са изведени оценки на паралелните времена.

2.6.1 Изчислителна сложност

При разделяне на областта Ω на $n_1 \times n_2 \times n_3$ елемента общият брой неизвестни N е

$$N = 3n_1n_2n_3 + n_1n_2 + n_1n_3 + n_2n_3.$$

Забележка 2.6.1. В този раздел ще броим умножението с натрупване (умножение плюс събиране) като *една операция.* Този избор е оправдан от два

факта: първо, при всички използвани векторни операции, както и при умножение на матрица по вектор и при решаването на системи с преобусловителя след всяко умножение следва натрупване и второ, при съвременните процесори събирането може, и се изпълнява *едновременно* с умножението и не изисква допълнително време за изпълнение.

При МСГП на всяка итерация се извършва едно умножение на матрица по вектор, едно решаване на система с преобуславящата матрица, две скаларни произведения и три векторни операции от тип умножение по скалар и прибавяне на друг вектор. Тоест имаме

$$\mathcal{N}_{it}^{\text{МСГП}} = \mathcal{N}(A\mathbf{v}) + \mathcal{N}(C^{-1}\mathbf{v}) + 2\mathcal{N}(\mathbf{u}^T\mathbf{v}) + 3\mathcal{N}(\alpha\mathbf{u} + \mathbf{v}). \tag{2.6.41}$$

Тъй като локалните елементни матрици A_e са с размери 6×6 и всеки възел от Ω не лежащ на границата е общ за точно 2 елемента, то редовете в матрицата A съответстващи на вътрешните възли имат точно 11 ненулеви елемента. Останалите редове имат по 6 ненулеви елемента. Броят на възлите по границата на Ω е

$$N_{\partial\Omega} = 2(n_1n_2 + n_2n_3 + n_3n_1).$$

Така броят на аритметичните операции при умножението на матрицата A по вектор е равен на броя на ненулевите елементи на матрицата A $Z(A)$, тоест

$$\begin{aligned} Z(A) = \mathcal{N}(A\mathbf{v}) &= 11(N - N_{\partial\Omega}) + 6N_{\partial\Omega} && \text{(2.6.42а)} \\ &= 33n_1n_2n_3 + n_1n_2 + n_2n_3 + n_3n_1 && \text{(2.6.42б)} \\ &\approx 11N. && \text{(2.6.42в)} \end{aligned}$$

Нека обърнем внимание на една оптимизация, която използваме при решаването на системата с MIC(0) преобусловителя:

$$C = (X - L)X^{-1}(X - L^T). \tag{2.6.43}$$

При решаването на долнотриъгълната система (вижте (2.3.8)) имаме деление с диагоналния елемент $x_{i,i}$. Но следващата стъпка е решаването на диагоналната система от преобусловителя, което по същество е умножение по същия този елемент $x_{i,i}$. По този начин, ако запазим междинния резултат (преди делението с $x_{i,i}$) получаваме резултата след решаване на долно триъгълната и диагоналната системи. Делението с $x_{i,i}$ трябва да се извърши за да продължи решаването на триъгълната система. Нещо повече, ако пазим реципрочните стойности на $x_{i,i}$, избягваме напълно по-бавната за изпълнение операция деление.

За да преброим операциите при решаването на системата с преобусловителя, първо ще преброим ненулевите елементи в спомагателните матрици B^{B1} за Вариант 1 и B^{B2} за Вариант 2.

При Вариант 1 отпадат връзките между срещуположните възли във всеки елемент. Всяка връзка съответства на 2 ненулеви елемента. Следователно за броя на ненулевите елементи $Z(B^{B1})$ получаваме

$$Z(B^{B1}) = Z(A) - 6n_1n_2n_3 \approx 9N. \tag{2.6.44}$$

При Вариант 2 на преобусловителя освен връзките отпадащи при Вариант 1 отпадат още 4 връзки. Така за $Z(B^{B2})$ получаваме

$$Z(B^{B2}) = Z(A) - 14n_1n_2n_3 \approx 6N. \tag{2.6.45}$$

Броят на ненулевите елементи в строго долно триъгълните части L^{B1} и L^{B2} на B^{B1} и B^{B2} е

$$Z(L^{B1}) = \frac{Z(B^{B1}) - N}{2} \approx 4N, \tag{2.6.46}$$

$$Z(L^{B2}) = \frac{Z(B^{B2}) - N}{2} \approx 2.5N. \tag{2.6.47}$$

Така за решаване на системата във Вариант 1 на преобусловителя C^{B1} са необходими

$$\mathcal{N}((C^{B1})^{-1}\mathbf{v}) = 2Z(L^{B1}) + 2N \approx 10N \tag{2.6.48}$$

операции. Аналогично, за Вариант 2 получаваме

$$\mathcal{N}((C^{B2})^{-1}\mathbf{v}) = 2Z(L^{B2}) + 2N \approx 7N. \tag{2.6.49}$$

За общия брой операции на една итерации от МСГП за Вариант 1 получаваме

$$\mathcal{N}_{it}^{\text{МСГП}} \approx 26N, \tag{2.6.50}$$

а за Вариант 2

$$\mathcal{N}_{it}^{\text{МСГП}} \approx 23N. \tag{2.6.51}$$

2.6.2 Времена

Оценките са изведени при следните допускания:

а) За изпълнението на M аритметични операции на един процесор е необходимо време

$$T = Mt_a \quad [\text{s}], \tag{2.6.52}$$

където t_a е средното време необходимо за изпълнение на една операция.

б) Времето за прехвърляне на M елемента между два съседни процесора може да бъде приближено с

$$T^{comm} = t_s + Mt_c \quad [\text{s}]. \tag{2.6.53}$$

Тук параметърът t_s е независещото от размера на съобщението време за започване на комуникацията (start-up time), а t_c е времето с което се увеличава общото време за прехвърлянето на всеки един от M-те елемента. Константата t_c характеризира скоростта (bandwidth) на комуникационния канал, докато t_a зависи от дължината на комуникационния канал, броя на междините ретранслатори между източника и приемника на съобщението.

в) Изпращането и получаването на данни между два съседни процесора може да се извършва паралелно. Тоест съществуват независими канали за комуникация в различните посоки.

Времето t_c е обратнопропорционално на скоростта на мрежата. Времето t_s включва операции, като проверка на параметрите на използваните функции, определяне на маршрут, инструктиране на хардуера. В съвременните високоскоростни компютърни мрежи е в сила следната релация:

$$t_s \gg t_c. \tag{2.6.54}$$

Това означава, че времето за пращане на N елемента в едно съобщение е много по-малко (от порядъка на N пъти) от времето за изпращане на 1 елемент N пъти. Времето t_s може, да бъде разглеждано, като времето необходимо за прехвърляне на първия елемент, а времето t_c – това за всеки следващ.

Разглежданият модел наподобява качването на хора с ескалатор. Ако допуснем, че на едно стъпало в ескалатора може да стои само един човек (и няма хора, които бързат, катерейки се по стъпалата), то t_s е времето, за което едно стъпало се качва от най-долния край на ескалатора до най-горния, а $t_c = t_s/s$, където s е броят на стъпалата в ескалатора (тези стъпала, които се виждат в даден момент).

Трябва да отбележим, че t_s е ограничено отдолу, от фундаменталните закони на физиката, като се има пред вид, че скоростта на светлината $c = 299\ 792\ 458\text{m/s}$, а това е и скоростта, с която се разпространяват електрическите сигнали. Тогава, ако разгледаме един съвременен процесор, работещ на 3GHz, за един такт на процесора, едно ново съобщение може да достигне максимум 10cm от източника. Ако разстоянието между двата комуникиращи процесора е 1m (мерено по пътя на сигналите), то на съобщението му трябват поне 10 процесорни такта.

Друга съставка на времето t_s е времето за подготвяне на съобщението за изпращане: Това включва проверка на валидността на съобщението, определяне на начин, алгоритъм и маршрут за изпращане. В нашата аналогия с ескалаторите, това са допълнителните времена за размисъл, дали трябва да се качим с ескалатор или не (може интересуващият ни обект да е на същия етаж), да изберем конкретен ескалатор, да си харесаме стъпало на което да стъпим и да изберем с кой крак да престъпим.

Вземайки пред вид броя на прехвърляните елементи и горните допускания, получаваме следните оценки за времето за комуникация при отделните операции:

$$T^{comm}(A\mathbf{v}) \approx t_s + 2n_1n_2t_c, \tag{2.6.55}$$

$$T^{comm}(C_{B1}^{-1}\mathbf{v}) \approx 6n_1n_2t_s + 8n_1n_2t_c, \tag{2.6.56}$$

$$T^{comm}(C_{B2}^{-1}\mathbf{v}) \approx 2n_1t_s + 8n_1n_2t_c. \tag{2.6.57}$$

В предложения MIC(0) преобусловител, тези комуникации са локални и не зависят от броя на процесорите. За скаларното произведение са необходими една глобална комуникация от всички към един и една комуникация един към всички. При задачи с достатъчно голяма размерност те не влияят на водещите членове в оценката на общото паралелно време. Паралелната ефективност на алгоритъма не зависи от броя на итерациите (както и броя на итерациите не зависи от броя на процесорите), така че е достатъчно да изследваме времената

за една итерация. По този начин ще получим оценки за паралелните ускорения и ефективност.

Нека предположим, че областта Ω е разделена на еднакви подобласти (слоеве) Ω_i, $i = 1, \ldots, p$, така че изчисленията се разпределят почти по равно между процесорите. Ако допуснем също така, че не използваме възможността за припокриване между комуникациите и изчисленията, то за паралелното време за изпълнение на една итерация са в сила следните оценки:
Вариант 1:

$$\begin{aligned} T_p^{it} &\approx \frac{26Nt_a}{p} + T_{B1}^{comm}(C^{-1}\mathbf{v}) + T^{comm}(A\mathbf{v}) \\ &= \frac{26Nt_a}{p} + (6n_1n_2 + 1)t_s + 10n_1n_2t_c \end{aligned}$$

Вариант 2:

$$\begin{aligned} T_p^{it} &\approx \frac{23Nt_a}{p} + T_{B2}^{comm}(C^{-1}\mathbf{v}) + T^{comm}(A\mathbf{v}) \\ &= \frac{23Nt_a}{p} + (2n_1 + 1)t_s + 10n_1n_2t_c \end{aligned}$$

Ускоренията $S_p = T_1/T_p$ са съответно

Вариант 1:

$$\begin{aligned} S_p &\approx \frac{26Nt_a}{\dfrac{26Nt_a}{p} + (6n_1n_2 + 1)t_s + 10n_1n_2t_c} \\ &\approx \frac{p}{1 + \dfrac{2t_sp}{26n_3t_a} + \dfrac{10t_cp}{78n_3t_a}}, \end{aligned} \tag{2.6.58}$$

и Вариант 2:

$$\begin{aligned} S_p \approx & \frac{23Nt_a}{\dfrac{23Nt_a}{p} + (2n_1 + 1)t_s + 10n_1n_2t_c} \\ \approx & \frac{p}{1 + \dfrac{2t_sp}{69n_2n_3t_a} + \dfrac{10t_cp}{69n_3t_a}}. \end{aligned} \tag{2.6.59}$$

Тук е използвано приближението $N \approx 3n_1n_2n_3$. Ускоренията и следователно ефективностите $E_p = S_p/p$, ще растат заедно с n_3 и в двата варианта до теоретическите им максимуми $S_p = p$ и $E_p = 1$. Обаче, тъй като при съществуващите компютри $t_s \gg t_c$ и $t_s \gg t_a$ ще можем да очакваме добри ефективности само когато $n_3 \gg p\, t_s/t_a$. Ускоренията при Вариант 2 очакваме

да са много по-добри от тези при Варианта 1, защото се изпращат около $3n_2$ по-малко съобщения.

Трябва да отбележим, че когато комуникациите се припокриват с изчисленията паралелните времена намаляват. Припокриването им може значително да намали влиянието на бавните комуникационни мрежи с голяма латентност. Разбира се, това изисква голям брой изчисления припокриващи се с всяка една комуникация. При Вариант 2 около $3n_2$ пъти по-малко съобщения се изпращат от колкото при Вариант 1. Същевременно при решаването на системата на преобусловителя n_2 пъти повече аритметични операции могат да бъдат извършени по време на комуникацията. Това би трябвало да даде допълнително предимство на Вариант 2.

2.7 Числени експерименти

В този раздел са включени резултати от два вида числени експерименти. Едните показват сходимостта на предложените алгоритми. Вторият вид са резултати от паралелни тестове, при които са показани времената за изпълнение, както и паралелните ускорения и ефективности.

2.7.1 Сходимост

В настоящия раздел представяме резултати от числени експерименти демонстриращи сходимостта на предложените паралелни алгоритми. Решаваме следната задача:

$$-\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{2.7.60а}$$

$$u(\mathbf{x}) = 0, \qquad \mathbf{x} \in \partial\Omega, \tag{2.7.60б}$$

където $\Omega = [0,1]^3$, а матрицата $a(\mathbf{x})$ има следния вид:

$$a(\mathbf{x}) = \begin{bmatrix} 1+\epsilon e^{x_1+x_2+x_3} & & \\ & 1+\frac{\epsilon}{2}sin(2\pi(x_1+x_2+x_3)) & \\ & & 1+\frac{\epsilon}{2}sin(2\pi(x_1+x_2+x_3)) \end{bmatrix}.$$

Тук ϵ е параметър, а e е неперовото число. Дясната част $f(\mathbf{x})$ е подбрана така, че задачата (2.7.60) да има за решение функцията

$$u(\mathbf{x}) = sin(2\pi x_1)sin(2\pi x_2)sin(2\pi x_3). \tag{2.7.61}$$

Експериментите са направени на компютър с процесор "Intel(R) Core(TM) i7-3960X CPU @ 3.30GHz". Използвано е само едното ядро на процесора.

Областта Ω е разделена на $n \times n \times n$ елемента със страна $h = n^{-1}$. В Таблици 2.4, 2.5 и 2.6 са дадени параметърът n, броят на неизвестните N, броят на итерациите N_{it} и времената за решаване t в секунди, за различните методи за решаване, както и грешката $err(h)$ за различни стойности на параметъра ϵ. Грешката $err(h)$ е дефинирана по следния начин:

$$err(h) = ||u - u_h||_\infty, \tag{2.7.62}$$

където с u е означено точното решение, а с u_h – полученото приближено решение.

Сравнени са методът на спрегнатия градиент без преобусловител (МСГ), МСГП с MIC(0) преобусловител построен от матриците $A(MIC(0)_A)$, B – Вариант 1 ($MIC(0)_{B1}$) и B – Вариант 2($MIC(0)_{B2}$) за варианта на дискретизация MP.

Таблица 2.4: Сходимост при дискретизация MP, $\epsilon = 0$.

		МСГ		$MIC(0)_A$		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	40	0.01	15	0.01	15	0.02	20	0.01	1.25×10^{-2}
32	101 376	79	0.15	20	0.10	21	0.14	25	0.11	3.19×10^{-3}
64	798 720	157	3.08	27	1.28	28	1.54	32	1.36	8.05×10^{-4}
128	6 348 608	284	43.06	36	12.96	38	14.86	44	14.07	2.03×10^{-4}

Таблица 2.5: Сходимост при дискретизация MP, $\epsilon = 0.1$.

		МСГ		$MIC(0)_A$		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	72	0.02	17	0.01	15	0.02	21	0.01	2.13×10^{-2}
32	101 376	152	0.28	27	0.13	24	0.15	31	0.13	5.34×10^{-3}
64	798 720	318	6.18	45	2.04	40	2.12	50	2.06	1.34×10^{-3}
128	6 348 608	656	98.89	80	27.56	70	26.54	80	24.75	3.34×10^{-4}

Таблица 2.6: Сходимост при дискретизация MP, $\epsilon = 1$.

		МСГ		$MIC(0)_A$		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	152	0.03	27	0.01	25	0.03	21	0.01	4.68×10^{-2}
32	101 376	331	0.58	59	0.27	50	0.30	34	0.15	1.15×10^{-2}
64	798 720	701	13.56	139	5.97	115	5.83	60	2.44	2.86×10^{-3}
128	6 348 608	1470	221.83	339	114.31	272	101.10	102	31.32	7.15×10^{-4}

В Таблици 2.7, 2.8 и 2.9 са показани същите данни, но за варианта на дискретизация MV. При тази дискретизация, е изпуснат варианта $MIC(0)_A$, защото $MIC(0)$ факторизацията не може да бъде построена, поради наличието на положителни извъндиагонални елементи в матрицата A.

От Таблици 2.4–2.9 се вижда, че при всички направени експерименти грешката намалява както h^2, тоест

$$err(x) = O(h^2). \tag{2.7.63}$$

Таблица 2.7: Сходимост при дискретизация MV, $\epsilon = 0$.

		МСГ		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	59	0.02	39	0.04	56	0.03	1.25×10^{-2}
32	101 376	115	0.21	48	0.29	84	0.33	3.21×10^{-3}
64	798 720	228	4.45	61	3.15	103	4.09	8.62×10^{-4}
128	6 348 608	434	65.81	73	27.40	160	48.37	2.33×10^{-4}

Таблица 2.8: Сходимост при дискретизация MV, $\epsilon = 0.1$.

		МСГ		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	103	0.03	55	0.06	73	0.04	1.64×10^{-2}
32	101 376	220	0.39	85	0.50	102	0.42	4.13×10^{-3}
64	798 720	466	9.84	132	6.70	142	5.59	1.03×10^{-3}
128	6 348 608	974	147.09	212	77.80	240	72.32	2.58×10^{-4}

Таблица 2.9: Сходимост при дискретизация MV, $\epsilon = 1$.

		МСГ		$MIC(0)_{B1}$		$MIC(0)_{B2}$		
n	N	N_{it}	t	N_{it}	t	N_{it}	t	$err(h)$
16	13 056	198	0.05	72	0.07	83	0.05	2.92×10^{-2}
32	101 376	441	0.78	111	0.64	124	0.48	7.22×10^{-3}
64	798 720	955	18.41	179	8.97	172	6.73	1.80×10^{-3}
128	6 348 608	2037	306.19	321	117.97	275	82.60	4.49×10^{-4}

От извършените експерименти се вижда, че броят на итерациите при МСГ расте както n, тоест

$$N_{it}^{\text{МСГ}} = O(n) = O(N^{1/3}) \tag{2.7.64}$$

В случаите без скок на коефициентите ($\epsilon = 0$) за броя на итерациите в МСГ с MIC(0) преобусловител, и в двата варианта имаме

$$N_{it}^{\text{MIC(0)}} = O(n^{1/2}) = O(N^{1/6}) \tag{2.7.65}$$

При наличието на коефициентна анизотропия, сходимостта е малко по-лоша от тази в (2.7.65), но по-добра от (2.7.64).

При дискретизация тип МР се вижда, че използването на модифицирани матрици на коравина за построяването на факторизацията дава не по-лоша сходимост от използването на оригиналната матрица на коравина.

2.7.2 Паралелни времена

В този раздел са представени резултати от паралелни експерименти проведени на три различни паралелни компютърни системи. Показани са паралелните времена, ускорения и ефективности.

Разглеждаме моделна задача на Поасон в единичния куб с хомогенни гранични условия на Дирихле върху едната стена (тази с върхове с координати {(1,0,0), (1,0,1), (1, 1, 1), (1,1, 0)}). Използвано е равномерно разделяне на областта и по трите направления, тоест $n_1 = n_2 = n_3 = n$. Тогава за размера на дискретната задача имаме $N = 3(n^3 + n^2)$. Използван е следният относителен критерий за край в МСГ:

$$\frac{(C^{-1}\mathbf{r}^i, \mathbf{r}^i)}{(C^{-1}\mathbf{r}^0, \mathbf{r}^0)} < 10^{-9},$$

където с $\mathbf{r}^i$ е означен резидула на i-тата стъпка. Тъй като паралелните свойства на алгоритъма не зависят от използваната дискретизация (МР или МV), показани са резултати само с дискретизация от тип МР.

Експериментите са изпълнени на следните три компютърни системи, които ще наричаме за краткост С1, С2 и С3:

С1 Тази система е “IBM SP Cluster 1600”. Съставена е от 64 компютърни възела тип p5-575, свързани с по две връзки към “Federation HPS” (High Performance Switch). Всеки един от възлите p5-575 съдържа 8 процесора “IBM Power5” в SMP режим и има 16GB RAM. Процесорите са с тактова честота от 1.9GHz. Скоростта на мрежата е 16Gb/s.

С2 Тази система е “IBM LinuxCluster 1350”. Тя се състои от 512 двупроцесорни възела тип “IBM X335”. Всеки възел съдържа по два процесора “Intel Xeon Pentium IV” на 3.06GHz и по 2GB RAM. Възлите са свързани чрез гигабитова мрежа “Myrinet”.

С3 Последната система е “Cray XD1”. Състои се от 72 възела. Всеки възел има по два процесора “AMD Opteron” и по 4GB RAM. Процесорите работят на 2.4GHz. Процесорите са свързани с мрежа тип “Cray RaidArray”

със скорост 5.6Gb/s. Отличителна черта на тази система е изключително малкото време за започване на комуникацията (start-up time) – $1.6\mu s$.

Процесорите на системата C1 могат да изпълняват по 4 операции с плаваща точка (с двойна точност) на такт, докато тези на C2 и C3 - само по две.

В Таблица 2.10 са показани размерът на мрежата n, броя на итерациите N_{it}, времето за изпълнение T_p в секунди, паралелното ускорение S_p и паралелната ефективност E_p за Вариант 1 на системите C1, C2 и C3. Тук с p е означен броя на процесорите. Същите данни, но за изпълнението на Вариант 2 са показани в Таблица 2.11.

Таблица 2.10: Вариант 1

		C1			C2			C3		
p	$\frac{n}{N_{it}}$	T_p	S_p	E_p	T_p	S_p	E_p	T_p	S_p	E_p
1		1.08			1.28			0.70		
2	$\frac{31}{}$	0.94	1.15	0.57	0.93	1.38	0.69	0.65	1.07	0.54
4	22	0.91	1.19	0.30	0.83	1.55	0.39	0.45	1.57	0.39
8		1.14	0.95	0.12	1.12	1.15	0.14	0.37	1.90	0.24
16		1.28	0.84	0.05	1.11	1.15	0.07	0.37	1.89	0.11
1		9.87			12.90			7.07		
2	$\frac{63}{}$	6.98	1.41	0.71	8.56	1.51	0.75	4.44	1.59	0.79
4	31	5.91	1.67	0.42	6.39	2.02	0.50	3.54	2.00	0.50
8		5.46	1.81	0.23	7.12	1.81	0.23	2.77	2.55	0.32
16		5.83	1.69	0.11	11.9	1.08	0.07	2.40	2.94	0.18
1		95.3			134.2			78.1		
2	$\frac{127}{}$	63.2	1.51	0.75	84.2	1.59	0.80	43.4	1.80	0.90
4	44	48.5	1.96	0.49	54.2	2.47	0.62	29.2	2.67	0.66
8		36.9	2.59	0.32	45.0	2.98	0.37	20.3	3.84	0.48
16		34.2	2.79	0.17	74.8	1.79	0.11	16.2	4.83	0.30
1		1147			1474[1]			859[1]		
2	$\frac{255}{}$	659	1.74	0.87						
4	64	361	3.18	0.79	551	2.67[1]	0.67[1]	277	3.10[1]	0.78[1]
8		287	4.00	0.50	432	3.41[1]	0.42[1]	173	4.97[1]	0.62[1]
16		264	4.34	0.27	397	3.71[1]	0.23[1]	123	6.98[1]	0.43[1]

[1]Прогнозирани стойности

Тестовете с $n = 255$ не можеше да се проведат на един и два процесора, поради недостиг на памет на системите C2 и C3. Поради същата причина, експериментите на C2 при $n = 255$ и $p = 4$ са проведени на 4 възела, използвайки само по един процесор на възел. Представените последователни времена са приложени само за сравнение. Те са изчислени, като се вземат пред вид броя на итерациите, както и времената за изпълнение при $n = 128$ на един процесор. Програмите реализиращи Вариант 1 и Вариант 2 имат различен шаблон за достъп до паметта. На това се дължи различното поведение на последователните времена на отделните платформи, а по-конкретно по-голямото време за

Таблица 2.11: Вариант 2

		C1			C2			C3		
p	$\frac{n}{N_{it}}$	T_p	S_p	E_p	T_p	S_p	E_p	T_p	S_p	E_p
1		0.60			1.00			0.65		
2	31	0.30	2.00	1.00	0.65	1.54	0.77	0.39	1.68	0.84
4	22	0.20	3.05	0.76	0.44	2.27	0.57	0.27	2.37	0.59
8		0.17	3.58	0.45	0.31	3.29	0.41	0.16	4.02	0.50
16		0.08	7.09	0.44	0.27	3.65	0.23	0.11	5.78	0.36
1		6.03			11.14			7.48		
2	63	3.09	1.95	0.97	6.67	1.67	0.83	3.88	1.92	0.96
4	31	1.68	3.58	0.89	3.97	2.80	0.70	2.35	3.18	0.79
8		0.94	6.42	0.80	2.58	4.32	0.54	1.49	5.00	0.63
16		0.57	10.57	0.66	2.55	4.36	0.27	1.09	6.87	0.42
1		74.0			127.0			95.9		
2	127	38.1	1.94	0.97	74.5	1.70	0.85	50.0	1.92	0.96
4	44	20.8	3.56	0.89	41.1	3.10	0.77	25.0	3.83	0.96
8		10.7	6.94	0.86	23.9	5.31	0.66	13.8	6.97	0.87
16		5.63	13.14	0.82	18.8	6.74	0.42	8.6	11.11	0.69
1		910			1397[1]			1055[1]		
2	255	458	1.99	0.99						
4	61	206	4.4	1.10	470	2.97[1]	0.74[1]	291	3.62[1]	0.90[1]
8		111	8.21	1.02	253	5.52[1]	0.69[1]	154	6.85[1]	0.85[1]
16		61	14.88	0.92	149	9.38[1]	0.58[1]	85	12.41[1]	0.77[1]

[1]Прогнозирани стойности

изпълнение за Вариант 2 на платформата C3, въпреки, по-малкия брой аритметични операции. Повече по този въпрос може да намерите в Раздел 2.5.2.

Мрежата на системата C3 има изключително малка латентност, и виждаме, че Вариант 1 е по-бърз на системата C3 от колкото на система C1, въпреки че C1 има по 8 процесора на възел и комуникациите се извършват чрез общата памет при $p \leq 8$.

Вижда се, че броят на итерациите е от порядък $O(n^{1/2}) = O(N^{1/6})$, а общото време за изчисление расте както $O(n^{7/2}) = O(N^{7/6})$. Като правило, за фиксиран брой процесори, ускоренията и ефективностите растат при увеличаване на размера на задачата. И обратно, ефективността пада, при фиксирано n и увеличение на броя на процесорите. Това важи и за трите платформи и потвърждава теоретичния анализ.

При Вариант 1 приемливи ефективности се получават само когато отношението n/p е достатъчно голямо. Както очаквахме, за фиксирани p и n Вариант 2 работи много по-добре даже и при по-малки отношения n/p. Ясно се вижда как намаляването на броя на стъпките за комуникация подобрява паралелната производителност. Също така припокриването на комуникациите с изчисленията има по-голям положителен ефект при Вариант 2.

Ефективностите по-големи от 1 се дължат на нелинейната организация на оперативната памет. При изпълнение на повече от един процесор, програмите имат достъп до повече по-бърза кеш-памет, което води до допълнителното ускорение.

Глава 3

Паралелен метод за преобуславяне на уравненията на Ламе

В тази глава е представен паралелен метод за преобуславяне на уравненията на Ламе, базиран на паралелния MIC(0) преобусловител представен в предишната глава.

Резултатите от тази глава са публикувани в [53, 54, 61, 85]:

- S. Margenov and Y. Vutov. Parallel PCG algorithms for voxel FEM elasticity systems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 517–526, 2007
- I. Lirkov and Y. Vutov. Comparative analysis of high performance solvers for 3D elliptic problems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 483–492, 2007
- Y. Vutov. Parallel DD-MIC(0) Preconditioning of Nonconforming Rotated Trilinear FEM Elasticity Systems. *Large-Scale Scientific Computing*, *LNCS* **4818**, 745–572. Springer-Verlag, 2008
- I. Lirkov, Y. Vutov, M. Ganzha, and M. Paprzycki. Comparative Analysis of High Performance Solvers for 3D Elasticity Problems. *Numerical Methods and Applications*, *LNCS* **5434**, 392–399. Springer-Verlag, 2009

3.1 Постановка на задачата

Нека разгледаме следната линейно еластична задача, представена във вариационна формулировка: Да се намери функция $\mathbf{u} \in [H^1_E(\Omega)]^3 = \{\mathbf{v} \in [H^1(\Omega)]^3 : \mathbf{v}_{\Gamma_D} = \mathbf{u}_S\}$ такава, че

$$\int_\Omega [2\mu\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) + \lambda \operatorname{div}\mathbf{u} \operatorname{div}\mathbf{v}]d\Omega = \int_\Omega \mathbf{f}^t\mathbf{v}d\Omega + \int_{\Gamma_N} \mathbf{g}^t\mathbf{v}d\Gamma, \qquad (3.1.1)$$

$\forall \mathbf{v} \in [H^1_0(\Omega)]^3 = \{\mathbf{v} = [H^1(\Omega)]^3 : \mathbf{v}|\Gamma_D = 0\}$, където $\lambda > 0$ и $\mu > 0$ са константите на Ламе,

$$\varepsilon(\mathbf{u}) := 0.5(\nabla\mathbf{u} + (\nabla\mathbf{u})^t)$$

е симетричният тензор на деформация, $\mathbf{f}$ са обемните сили и $\mathbf{g}$ са зададените напрежения на границата $\Gamma_N \cup \Gamma_D = \partial\Omega$. За дискретизация на задачата 3.1.1 използваме неконформните елементи на Ранахер-Турек.

За получаване на стабилна апроксимация, обикновено се използва смесен МКЕ, където неизвестните са $\mathbf{u}$ и $\operatorname{div}\mathbf{u}$. Като се използват неконформни крайни елементи, дуалната променлива може да бъде елиминирана на макроелементно ниво [21]. По този начин се получава симетрична и положително определена система, само в термините на преместванията [4]. Този подход е известен в литературата като *reduced and selective integration* (RSI), вижте [57].

Нека $\Omega^H = n_1^H \times n_2^H \times n_3^H$ е мрежа, получена чрез равномерно разделяне на областта $\Omega \subset \mathbb{R}^3$ на шестостени, и нека $\Omega^h = n_1^h \times n_2^h \times n_3^h$ е мрежата, получена чрез разделянето на всеки макроелемент $E \in \Omega^H$ на 8 шестостена. В сила са равенствата $n_i^h = 2n_i^H$ за $i = 1, 2, 3$.

Разглеждаме следния вариант на RSI дискретизация [58]: Да се намери функция $\mathbf{u}^h \in V_E^h$, която да изпълнява равенството

$$\sum_{e\in\Omega^h} \int_e \left[2\mu\varepsilon^*(\mathbf{u}^h) : \varepsilon^*(\mathbf{v}^h) + \lambda \operatorname{div}\mathbf{u}^h \operatorname{div}\mathbf{v}^h\right] de = \int_\Omega \mathbf{f}^t\mathbf{v}^h d\Omega + \int_{\Gamma_N} \mathbf{g}^t\mathbf{v}^h d\Gamma, \quad (3.1.2)$$

$\forall \mathbf{v}^h \in V_0^h$, където

$$\varepsilon^*(\mathbf{u}) := \nabla\mathbf{u} - 0.5 I_L^{Q^H}[\nabla\mathbf{u} - (\nabla\mathbf{u})^t],$$

V_0^h е пространството, което изпълнява хомогенни гранични условия върху Γ_D. Означили сме с $I_L^{Q^H}$ оператора на ортогонална L^2 проекция върху пространството от на части константни функции Q^H в (макро)елементите от разделянето Ω^H на Ω.

След стандартна процедура за асемблиране на крайноелементна матрица получаваме следната линейна система алгебрични уравнения:

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} \mathbf{u}_h^1 \\ \mathbf{u}_h^2 \\ \mathbf{u}_h^3 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_h^1 \\ \mathbf{f}_h^2 \\ \mathbf{f}_h^3 \end{bmatrix}. \quad (3.1.3)$$

Тук матрицата на коравина K е записана в блочна форма, като блоковете $\mathbf{u}_h^i$ съответстват на преместванията по i-тото направление. Тъй като матрицата K е разредена, симетрична и положително определена, използваме метода на спрегнатия градиент с преобуславяне за решаването на системата (3.1.3).

3.2 Преобуславяне чрез разделяне по преместванията

Като основа за паралелния алгоритъм за решаване на системата на Ламе използваме т.н. изотропен вариант на разделяне по преместванията (DD, от англ. displacement decomposition) [14].

Разглеждаме DD преобусловител в следния вид:

$$C_{DD}(K) = \begin{bmatrix} A & & \\ & A & \\ & & A \end{bmatrix}. \quad (3.2.4)$$

Тук с A е означена матрица съответстваща на следната билинейна форма:

$$a(u^h, v^h) = \sum_{e \in \Omega^h} \int_e E_e \left(\sum_{i=1}^{3} \frac{\partial u^h}{\partial x_i} \frac{\partial v^h}{\partial x_i} \right) de. \tag{3.2.5}$$

С E_e е означен модулът на еластична деформация, известен също като модул на Юнг, за материала в съответния елемент. Такова преобуславяне базирано на разделяне по преместванията е теоретически мотивирано от второто неравенство на Корн, което е изпълнено за разглежданата RSI дискретизация [8]. За широк кръг задачи е в сила и зависимостта [14]

$$\kappa \left(C_{DD}^{-1} K \right) = O((1 - 2\nu)^{-1}), \tag{3.2.6}$$

където с ν е означен коефициента на Поасон за съответния материал.

3.3 Паралелен MIC(0) преобусловител

За паралелно решаване на системата (3.1.3) ще използваме разделяне по преместванията (3.2.4), както и разработения в Глава 2 паралелен алгоритъм за MIC(0) преобуславяне на задачата (3.2.5).

Построяваме

$$C_{DDMIC(0)}(K) = \begin{bmatrix} C_{MIC(0)}(B) & & \\ & C_{MIC(0)}(B) & \\ & & C_{MIC(0)}(B) \end{bmatrix}. \tag{3.3.7}$$

Помощната матрица B се конструира на елементно ниво по следния начин. Следвайки стандартната процедура в МКЕ за асемблиране на матрица на коравина, записваме A във вида

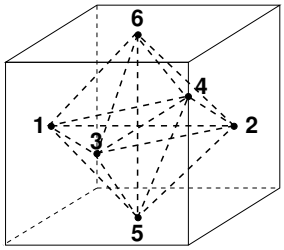
$$A = \sum_{e \in \Omega^h} L_e^T A_e L_e,$$

където с L_e е означено изображението от глобалния вектор с неизвестни към вектора с неизвестни за текущия елемент e, а

$$A_e = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}.$$

е елементната матрица на коравина.

Локалната номерация и шаблона на свързаност са изобразени на Фигура 3.1. Ще използваме двата варианта на локална апроксимация, от Глава 2.



Фигура 3.1: Локална номерация на възлите и шаблон на връзките

Вариант 1:

$$B_e^1 = \begin{bmatrix} b_{11} & & a_{13} & a_{14} & a_{15} & a_{16} \\ & b_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & b_{33} & & a_{35} & a_{36} \\ a_{41} & a_{42} & & b_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & b_{55} & \\ a_{61} & a_{62} & a_{63} & a_{64} & & b_{66} \end{bmatrix}$$

и **Вариант 2**:

$$B_e^2 = \begin{bmatrix} b_{11} & & a_{13} & a_{14} & a_{15} & a_{16} \\ & b_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & b_{33} & & & \\ a_{41} & a_{42} & & b_{44} & & \\ a_{51} & a_{52} & & & b_{55} & \\ a_{61} & a_{62} & & & & b_{66} \end{bmatrix}$$

Матриците B_e^1 и тези B_e^2 са симетрични и положително полуопределени, с неположителни извъндиагонални елементи. Сега можем да конструираме глобалните матрици

$$B^1 = \sum_{e \in \Omega_h} \lambda_e^1 L_e^T B_e^1 L_e$$

и

$$B^2 = \sum_{e \in \Omega_h} \lambda_e^2 L_e^T B_e^2 L_e,$$

където с λ_e^i, $i = 1, 2$ е означена най-малката нетривиална собствена стойност на обобщената задача за собствени стойности $(B_e^i)^{-1} A_e$. Както видяхме в предишната глава, така конструираните матрици B^1 и B^2 позволяват разпаралеляване на решаването на СЛАУ с тях.

Така получаваме два варианта за преобусловители на K:

$$C^1_{DDMIC(0)}(K) = \begin{bmatrix} C_{MIC(0)}(B^1) & & \\ & C_{MIC(0)}(B^1) & \\ & & C_{MIC(0)}(B^1) \end{bmatrix} \tag{3.3.8}$$

и

$$C^2_{DDMIC(0)}(K) = \begin{bmatrix} C_{MIC(0)}(B^2) & & \\ & C_{MIC(0)}(B^2) & \\ & & C_{MIC(0)}(B^2) \end{bmatrix}. \tag{3.3.9}$$

3.4 Паралелен мултигрид преобусловител

Друг възможен вариант на преобуславяне е да решим системите с матрица A в (3.2.4) с вътрешна итерация с МСГП, като за преобусловител използваме алгебричния мултигрид BoomerAMG от библиотеката Hypre [41, 87]. Така получаваме следния преобусловител на K:

$$C_{DD\ BAMG}(K) \approx \begin{bmatrix} A & & \\ & A & \\ & & A \end{bmatrix}, \tag{3.4.10}$$

където при решаването на системи с преобусловителя $C_{DD\ BAMG}$ прилагаме вътрешни итерации на МСГ с BoomerAMG преобусловител:

$$C^{-1}_{DD\ BAMG} \begin{bmatrix} \mathbf{b^1} \\ \mathbf{b^2} \\ \mathbf{b^3} \end{bmatrix} = \begin{bmatrix} \text{МСГП}(A, \mathbf{b^1}) \\ \text{МСГП}(A, \mathbf{b^2}) \\ \text{МСГП}(A, \mathbf{b^3}) \end{bmatrix} \tag{3.4.11}$$

Тук с МСГП$(A, \mathbf{b})$ е означено решаването на система с матрица A и дясна част $\mathbf{b}$ по МСГ с BoomerAMG преобусловител.

Възможни са редица варианти за реализирането на тези вътрешни итерации. Например да се извършват фиксиран брой вътрешни итерации (например една или две). Друг вариант е да се зададе подходящ относителен критерий за грешка на вътрешния цикъл на МСГП. Първият вариант води до ниска цена на вътрешните итерации, но съответно по-голям брой външни итерации. При втория – вътрешните итерации стават по-скъпи но броят на външните намалява. Не сме си поставяли за цел да намерим оптимален вариант – още повече, самият мултигрид преобусловител има много параметри, и изчерпването на всички варианти е непрактично. Опитът ни показва че при различни задачи различни комбинации са по-добри.

3.5 Реализация

Да припомним означенията от Глава 3.1. $\Omega^H = n_1^H \times n_2^H \times n_3^H$ е мрежа, получена чрез равномерно разделяне на областта $\Omega \subset \mathbb{R}^3$ на шестостени, и нека $\Omega^h = n_1^h \times n_2^h \times n_3^h$ е мрежата, получена чрез разделянето на всеки макроелемент $E \in \Omega^H$ на 8 шестостена. Общият брой на неизвестните след RSI дискретизация върху мрежата Ω^h е

$$N = 3(3n_1^h n_2^h n_3^h + n_1^h n_2^h + n_1^h n_3^h + n_2^h n_3^h). \tag{3.5.12}$$

При $n_1^h = n_2^h = n_3^h = n$, изразът (3.5.12) се опростява:

$$N = 9n^2(n+1). \tag{3.5.13}$$

Използваната L^2 проекция $I_L^{Q^H}$ върху на части константните функции върху макроелементите увеличава силно свързаността на матрицата, а от там и броя на ненулевите елементи в нея. Когато макроелементите са с еднаква форма, то макроелементните матрици на коравина за един и същи материал са еднакви.

Избрали сме да не пазим глобалната матрица на коравина K, която има приблизително $16N$ ненулеви елемента, а при решаването на задачи, операцията умножение на матрицата K с вектор извършваме макроелемент по макроелемент. Това ограничава областите в които можем (ефективно) да решаваме задачи, но пести много памет. Както ще видим в Глава 4 това не намалява приложимостта на разглежданите алгоритми за важни задачи от практиката. При малък брой материали (до няколко стотин такива) макроелементните матрици K^E могат (е практично) да бъдат предварително пресметнати и да се взимат от таблица. В противен случай може да се използва зависимостта

$$K^E(\lambda, \mu) = \lambda K_l^E + \mu K_m^E, \tag{3.5.14}$$

където λ и μ са параметрите на Ламе, а K_l^E и K_m^E са независещи от материала матрици.

Съхранението на векторите с преместванията в паметта, както и разпределянето им по процесорите става по същия начин, както тези при решаването на скаларната елиптична задача (виж. Глава 2.5.2). При скаларния вариант имаме вектора

$$\mathbf{v} = \{v_{i,j,k}^l\},$$

където индексите i, j и k определят елемента, индекса l стената на елемента. Така при решаване на задачата на Ламе векторите се съхраняват в 5 мерни масиви (4 мерни масиви от наредени тройки)

$$\mathbf{d} = \{d_{i,j,k}^{l,q}\},$$

където индекса $q = \{0,1,2\}$ дава преместванията съответно по $\{x_1, x_2, x_3\}$.

Както споменахме по-горе извършването на операцията умножение на матрицата K с вектор, става на макроелементно ниво. Изпълнява се Алгоритъм 3.1. Нека $p \leq n_3 + 1$ е броят на процесорите. Когато $p > n_3/2$ при умножението на глобалната матрица на коравина по вектор, част от процесорите не работят (Те само чакат да получат тяхната част от резултата от умножението (стъпка (9)).

Алгоритъм 3.1. *Умножение на матрицата на коравина по вектор* $\mathbf{r} = K\mathbf{v}$*:*

(1) $\mathbf{r} = \mathbf{0}$
(2) Започни изпращането и получаването на елементи от $\mathbf{v}$ *за и от съседните процесори*
(3) За всички макроелементи E*, за които изчисленията не зависят от елементите на* $\mathbf{v}$ *в съседните процесори:*
(4) Извличане на елементите съответстващи на текущия макроелемент E *от* $\mathbf{v}$ *в* $\bar{\mathbf{v}}$
(5) Изпълнение $\bar{\mathbf{r}} = K^E\bar{\mathbf{v}}$
(6) Добавяне на елементите съответстващи на текущия макроелемент от $\bar{\mathbf{r}}$ *към* $\mathbf{r}$*;*
(7) Изчакване на приключването на комуникацията от стъпка (2).
(8) За останалите макроелементи изпълни стъпките (4)(5) и (6).

(9) Извърши комуникация за изпращане на частите от вектора с резултата **r** *към съседните процесори.*

Решаването на системите с преобусловителите (3.3.8) и (3.3.9) става чрез използването на *същите процедури и функции*, които реализират решаването на системите (2.5.39) и (2.5.40).

Тези процедури са разработени като C++ шаблони – тоест са параметризирани по тип. В скаларния вариант те използват базов тип скалар (float или double), а при преобуславянето на системите на Ламе, базовият тип е наредена тройка от преместванията във всяка точка (тримерен вектор). По същия начин (със същия код) се реализират и комуникациите, само че в този случай дефинираните от нас типове на MPI се конструират от базов тип тримерен вектор, а не от скалар.

Трите системи (3.3.8) или (3.3.9) се решават едновременно. Броят на комуникациите за решаването на трите системи е същият, както и за една. Разликата е, че броят на прехвърляните елементи е три пъти по-голям.

3.6 Оценки за паралелните времена

В този раздел са изведени оценки на паралелните времена. за решаване на системата на Ламе с предложените паралелни преобусловители.

3.6.1 Изчислителна сложност

Да припомним операциите в МСГП. Имаме умножение на матрица с вектор, решаване на система с преобусловителя, две скаларни произведения на вектори и три операции от тип умножение на вектор с число и прибавянето му към друг. За общия брой аритметични операции имаме

$$\mathcal{N}_{it}^{\text{МСГП}} = \mathcal{N}(K\mathbf{v}) + \mathcal{N}(C^{-1}\mathbf{v}) + 2\mathcal{N}(\mathbf{u}^T\mathbf{v}) + 3\mathcal{N}(\alpha\mathbf{u} + \mathbf{v}). \tag{3.6.15}$$

За операцията умножение на матрица по вектор имаме следното: Извършваме умножението макроелемент по макроелемент. Броят на ненулевите елементи в макроелементната матрица е 1740, а тя е с размери 108x108. Броят на макроелементите е $n_1^H n_2^H n_3^H$ или $\frac{1}{8}n_1^h n_2^h n_3^h$. От сега нататък ще изпускаме за краткост горния индекс h, тоест ще пишем n_i вместо n_i^h. За всеки ненулев елемент имаме по една операция умножение и събиране, и накрая имаме по една операция събиране за всеки елемент от вектора с резултата (при прибавянето му към глобалния такъв). Така получаваме

$$\mathcal{N}(K\mathbf{v}) = (1740 + 108)\frac{1}{8}n_1^h n_2^h n_3^h = 231 n_1^h n_2^h n_3^h \approx 25\frac{2}{3}N \tag{3.6.16}$$

За решаван на системи с преобусловителя се решават три скаларни системи и броят на операциите е три пъти този от (2.6.48) за Вариант 1 и от (2.6.49) за Вариант 2 на преобусловителя. Получаваме за Вариант 1

$$\mathcal{N}((C^1_{DDMIC(0)})^{-1}\mathbf{v}) \approx 10N \tag{3.6.17}$$

и

$$\mathcal{N}((C^2_{DDMIC(0)})^{-1}\mathbf{v}) \approx 7N \tag{3.6.18}$$

за Вариант 2 на преобусловителя.

Забележка 3.6.1. Забележете, че във формули (2.6.48) и (2.6.49) N е броят на неизвестните в скаларната задача. Въпреки че формулите (3.6.17) и (3.6.18) са същите като (2.6.48) и (2.6.49), тук броят на променливите N е три пъти по-голям.

Вече можем да пресметнем и общия брой операции за една итерация. За Вариант 1 получаваме:

$$\mathcal{N}_{it}^{\text{МСГП}} \approx 25\frac{2}{3}N + 10N + 5N = 40\frac{2}{3}N, \tag{3.6.19}$$

а за Вариант 2

$$\mathcal{N}_{it}^{\text{МСГП}} \approx 25\frac{2}{3}N + 7N + 5N = 37\frac{2}{3}N. \tag{3.6.20}$$

Вижда се, че преобуславянето е евтина операция в сравнение с умножаването на матрица по вектор.

3.6.2 Времена

Тук, както в Глава 2.6.2 ще изведем оценки за паралелните времена в термините на константите t_a - времето за извършване на една операция, t_s времето за прехвърляне на първия елемент в комуникацията и t_c – времето за прехвърляне на всеки следващ елемент.

Ето какви са времената за комуникации на отделните операции, за всяка стъпка от МСГП:

$$T^{comm}(K\mathbf{v}) \approx \quad 2t_s + 6n_1n_2t_c \approx \quad 2t_s + \frac{4}{3}N^{2/3}t_c, \tag{3.6.21}$$

$$T^{comm}(C_{B1}^{-1}\mathbf{v}) \approx \quad 6n_1n_2t_s + 24n_1n_2t_c \approx \quad \frac{2}{3}N^{2/3}t_s + \frac{8}{3}N^{2/3}t_c, \tag{3.6.22}$$

$$T^{comm}(C_{B2}^{-1}\mathbf{v}) \approx \quad 2n_1t_s + 24n_1n_2t_c \approx \quad \frac{2}{9}N^{1/3}t_s + \frac{8}{3}N^{2/3}t_c. \tag{3.6.23}$$

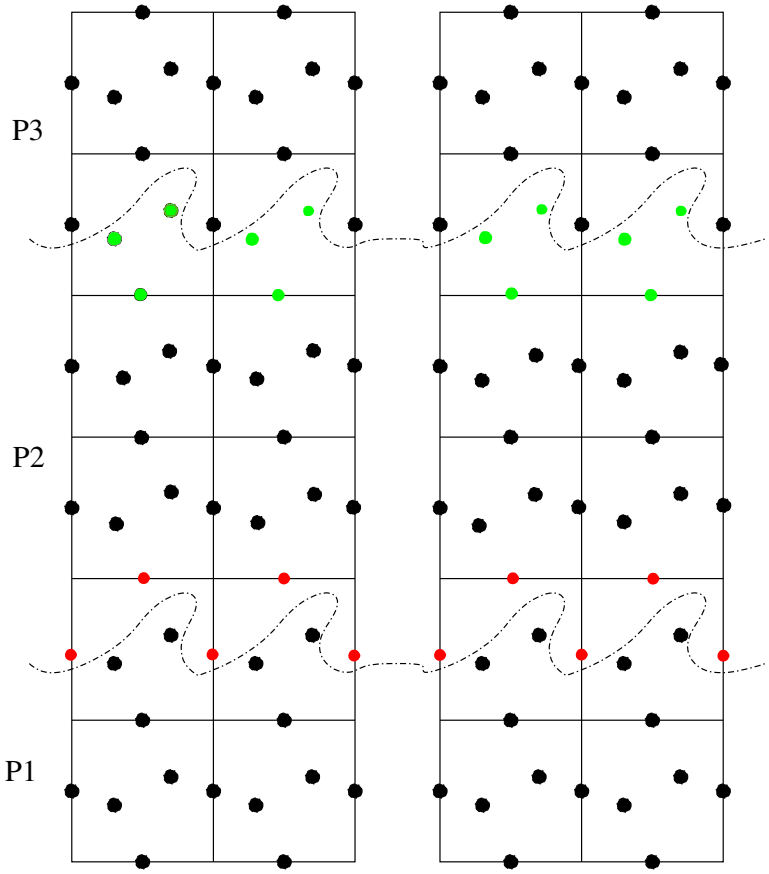
Две комуникационни стъпки са необходими за умножението на матрица по вектор. Това се прави за да не се дублират изчисления и да се избегне изпълнението на допълнителни проверки. На всяка стъпка се прехвърлят стойностите от $n_1n_2+n_1(n_2+1)$ или $n_1n_2+(n_1+1)n_2$ възли в едната посока (Виж. Фиг. 3.2). Това прави приблизително $3 \times 2n_1n_2$ на брой елемента или, $\frac{4}{3}N^{2/3}t_c$ на брой в случая $n_1 = n_2 = n_3$.

При решаване на системи с преобусловителите броят на комуникационните стъпки е същият, както при скаларния случай – (2.6.56) и (2.6.57), но броят на прехвърляните елементи се увеличава три пъти.

Така от (3.6.19)—(3.6.23) получаваме следния израз за общото време на итерация (при $p \leq n_3/2$):

$$T_{B1}^{it}(p) \approx \frac{41Nt_a}{p} + \frac{2}{3}N^{2/3}t_s + 4N^{2/3}t_c, \tag{3.6.24}$$

$$T_{B2}^{it}(p) \approx \frac{38Nt_a}{p} + \frac{2}{9}N^{1/3}t_s + 4N^{2/3}t_c \tag{3.6.25}$$



Фигура 3.2: Схема на комуникациите за умножението на матрица по вектор. 1) Елементите от възлите с червен цвят се прехвърлят от P2 към P1. Тези от възлите със зелен цвят се прехвърлят от P2 към P3. 2)След извършване на умножението за всеки макроелемент, елементите от решенията съответстващи на тези възли се прехвърлят от P1 към P2 и от P3 към P2.

Нека пресметнем относителните ускорения $S(p) = T(1)/T(p)$ и ефективности $E(p) = S(p)/p$.

$$\begin{aligned} S^{it}_{B1}(p) &\approx \frac{41Nt_a}{\frac{41Nt_a}{p} + \frac{2}{3}N^{2/3}t_s + 4N^{2/3}t_c} \\ &\approx \frac{p}{1 + \frac{p}{N^{1/3}}\left(\frac{2t_s}{123t_a} + \frac{4t_c}{41t_a}\right)} \end{aligned} \tag{3.6.26}$$

$$\begin{aligned} S^{it}_{B2}(p) &\approx \frac{38Nt_a}{\frac{38Nt_a}{p} + \frac{2}{9}N^{1/3}t_s + 4N^{2/3}t_c} \\ &\approx \frac{p}{1 + \frac{p}{N^{2/3}}\frac{t_s}{171t_a} + \frac{p}{N^{1/3}}\frac{2t_c}{19t_a}} \end{aligned} \tag{3.6.27}$$

$$E_{B1}^{it}(p) \approx \left(1 + \frac{p}{N^{1/3}}\left(\frac{2t_s}{123t_a} + \frac{4t_c}{41t_a}\right)\right)^{-1} \tag{3.6.28}$$

$$E_{B2}^{it}(p) \approx \left(1 + \frac{p}{N^{2/3}}\frac{t_s}{171t_a} + \frac{p}{N^{1/3}}\frac{2t_c}{19t_a}\right)^{-1} \tag{3.6.29}$$

Вижда се, че с нарастването на N (n_3), при фиксиран брой на процесорите p ускоренията и ефективностите клонят към оптималните си граници $S(p) = p$ и $E(p) = 1$. Разбира се, при съвременните компютри, където са в сила релациите $t_s \gg t_c$ и $t_s \gg t_a$, можем да очакваме добра ефективност, когато $n_3 \gg p\, t_s/t_a$.

От (3.6.28) и (3.6.29) се вижда, че можем да очакваме много по-добра ефективност за Вариант 2, от колкото за Вариант 1, поради по-малкия брой изпратени съобщения.

Ако се сравнят оценките за ефективностите с тези за решаване на скаларната задача, можем да очакваме по-голяма ефективност за еластичната задача, поради по-големия брой изчисления, при запазване на същия брой комуникационни стъпки.

3.7 Числени експерименти

3.7.1 Сходимост

Сравнение с аналитично решение

За проверка на сходимостта на метода решихме системата от уравнения на Ламе

$$(\lambda + \mu)\sum_{j=1}^{3}\frac{\partial^2 u_j}{\partial x_i \partial x_j} + \mu\sum_{j=1}^{3}\frac{\partial^2 u_i}{\partial x_j^2} = F_i \quad i = 1, 2, 3 \tag{3.7.30}$$

със следните гранични условия

$$u_i(\mathbf{x}) = g_i(\mathbf{x}) \qquad \mathbf{x} \in \Gamma_D, \tag{3.7.31}$$

$$\sum_{j=1}^{3}\sigma_{ij}(\mathbf{x})n_j = h_i(\mathbf{x}) \qquad \mathbf{x} \in \Gamma_N, \tag{3.7.32}$$

в единичният куб.

Зададени са подходящи гранични условия, както са и пресметнати десните части в системата на Ламе(3.7.30) F_i, така че следната функция

$$u(\mathbf{x}) = \begin{bmatrix} x_1^3 x_2 x_3 + \sin(x_1 + x_2 + x_3) \\ x_1 x_2^3 x_3 + \cos(x_1 + x_2 + x_3) \\ x_1 x_2 x_3^3 + \sin(x_1 + x_2 - x_3) \end{bmatrix} \tag{3.7.33}$$

да е решение на граничната задача. Наложено е гранично условие на Дирихле на едната стена от областта, а на останалите – такива на фон Нойман. Свойствата на материала са коефициент на Поасон $\nu = 0.2$ и модул на Юнг $E = 3.6$Pa.

Използван е следният критерий за край при МСГП:

$$(C^{-1}\mathbf{r}^i, \mathbf{r}^i)/(C^{-1}\mathbf{r}^0, \mathbf{r}^0) < \varepsilon^2, \tag{3.7.34}$$

където с $\mathbf{r}^i$ е означен резидулът на i-тата итерация, с C е означен използваният преобусловител и относителна грешка $\varepsilon = 10^{-9}$.

Таблица 3.1: Сходимост при дискретизация MP

n	N	МСГ	$DD_{MIC(0)A}$	$DD_{MIC(0)B1}$	$DD_{MIC(0)B2}$	DD_{BAMG}	err(h)
16	39 168	429	137	137	169	40	3.11×10^{-2}
32	304 128	824	193	192	245	40	9.57×10^{-3}
64	2 396 160	1 609	269	270	350	39	2.84×10^{-3}
128	19 021 824	3 177	374	378	496	38	8.20×10^{-4}
256	151 584 768	–	520	530	694	–	2.33×10^{-4}

В Таблица 3.2 са показани същите данни, но за варианта на дискретизация MV. При тази дискретизация, е изпуснат варианта $MIC(0)_A$, защото $MIC(0)$ факторизацията не може да бъде построена, поради наличието на положителни извъндиагонални елементи в матрицата A.

Таблица 3.2: Сходимост при дискретизация MV

n	N	МСГ	$DD_{MIC(0)B1}$	$DD_{MIC(0)B2}$	DD_{BAMG}	err(h)
16	39 168	581	422	553	39	3.01×10^{-2}
32	304 128	1 101	577	792	40	9.33×10^{-3}
64	2 396 160	2 123	806	1 088	39	2.78×10^{-3}
128	19 021 824	4 210	1 116	1 864	38	8.05×10^{-4}
256	151 584 768	–	1 401	3 500	–	2.29×10^{-4}

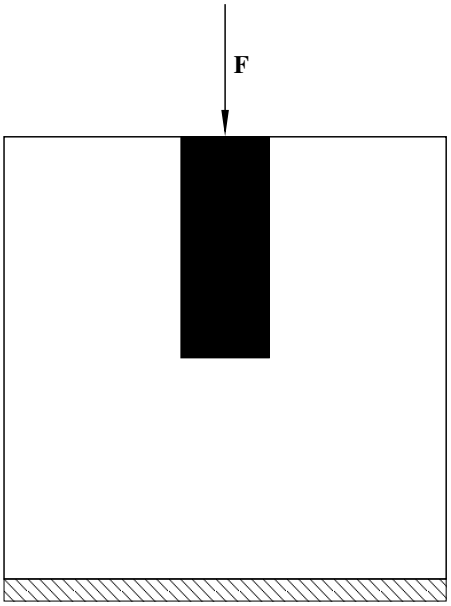
От таблици 3.1 и 3.2 ясно се вижда, че броят на итерациите расте, както $O(n^{1/2}) = O(N^{1/6})$ за двата вида дискретизация, както и за двата варианта на преобусловителя на база непълна факторизация. Броят на итерациите е забележимо по-голям за дискретизация от тип MV в сравнение с дискретизацията MP. При Вариант 2 на преобусловителя броят на итераците е по-голям в сравнение с Вариант 1. Вижда се, също така, че итерациите при преобусловителя базиран на алгебричен мултигрид, практически не зависят от размера на задачата.

Сходимост за задача със скок в коефициентите

В този раздел ще илюстрираме сходимостта на предложените методи за задача с променливи скокове в коефициентите.

Изчислителната област е куб със страна $10m$. Хомогенни гранични условия на Дирихле са приложени на долната страна на областта. Използвана е равномерна мрежа, като по всяко едно направление областта е разделена на n части.

Изследваме взаимодействието между пилотен фундамент и почвата. областта на пилота е $\Omega_f = [3.75, 6.75]m \times [3.75, 6.75]m \times [5, 10]m$. Механичните характеристики на почвата са $E_s = 1MPa$ и $\nu_s = 0.2$, а тези на пилотния фундамент – $E_f = J \times 1$MPa и $\nu_f = 0.2$. Изпълнили сме числени експерименти със скок $J \in \{1, 10, 100, 1000\}$. Приложена е сила от $10MN$ върху горната част от пилотния фундамент. На Фигура 3.3 е показано сечение на изчислителната област.



Фигура 3.3: Опитна постановка на симулацията на натоварване на пилотен фундамент.

Отново е използван относителен критерий за край на итерациите на МСГП: $(C^{-1}\mathbf{r}^i, \mathbf{r}^i)/(C^{-1}\mathbf{r}^0, \mathbf{r}^0) < \varepsilon^2$, където с $\mathbf{r}^i$ е означен резидулът на i-тата итерация, а $\varepsilon = 10^{-6}$.

В Таблица 3.3 и Таблица 3.4 са представени резултати за броя на итерациите и за двата варианта на преобуславяне съответно за дискретизация MP и за дискретизация MV в зависимост от скока J. В Таблица 3.3 е добавена и колона „B0" съответстваща на прилагането на MIC(0) факторизация без модификация на матрицата A. Забележете, че това е възможно само при дискретизация MP, поради появяването на положителни извъндиагонални елементи на A при дискретизация MV и само на един процесор!

От горните таблици ясно се вижда стабилността на предложените преобусловители. Броят на итерациите е от порядък $O(n^{1/2}) = O(N^{1/6})$. Наблюдава се увеличение на броя на итерациите заедно с увеличения на скока J и при двата вида дискретизация. Това увеличение е съществено по-малко от $J^{1/2}$. Освен това се вижда, че броят на итерациите при дискретизация MV са повече от тези при дискретизация MP.

Забележително е, че при дискретизация MP, броят на итерациите за Вариант 2 е по-малък от този за Вариант 1, който на свой ред е по-малък от

Таблица 3.3: Брой итерации при дискретизация MP

		$J=1$			$J=10$			$J=100$			$J=1000$		
n	N	B0	B1	B2	B0	B1	B2	B0	B1	B2	B0	B1	B2
32	304 128	161	147	113	186	173	130	227	253	189	361	343	247
64	2 396 160	264	223	162	284	262	186	428	391	271	565	523	357
128	19 021 824	367	331	230	424	389	264	638	581	385	843	780	509
256	151 584 768		486	327		570	377		852	542		1 148	725

Таблица 3.4: Брой итерации при дискретизация MV

		$J=1$		$J=10$		$J=100$		$J=1000$	
n	N	B1	B2	B1	B2	B1	B2	B1	B2
32	304 128	173	255	197	280	313	348	405	411
64	2 396 160	295	648	310	744	486	904	630	1069
128	19 021 824	471	916	536	1 053	778	1 281	1 013	1 517
256	151 584 768	730	1 282	857	1 486	1 198	1 813	1 600	2 154

получените итерации без модификация на матрицата A.

3.7.2 Паралелни времена

Тук ще представим паралелни времена, ускорения и ефективности на предложените преобусловители на три паралелни компютъра с разпределена памет за решаване на задачата за пилотния фундамент от предишния раздел, в случая на най-голям скок на коефициента $J = 1000$.

Експериментите са проведени на същите три компютърни системи C1, C2 и C3, както и тестовете от раздел 2.7.2. Нека ги припомним:

C1 "IBM SP Cluster 1600". Съставена е от 64 компютърни възела тип p5-575, свързани с по две връзки към "Federation HPS" (High Performance Switch). Всеки един от възлите p5-575 съдържа 8 процесора "IBM Power5" в SMP режим и има 16GB RAM. Процесорите са с тактова честота от 1.9GHz. Скоростта на мрежата е 16Gb/s.

C2 "IBM LinuxCluster 1350". Тя се състои от 512 двупроцесорни възела тип "IBM X335". Всеки възел съдържа по два процесора "Intel Xeon Pentium IV" на 3.06GHz и по 2GB RAM. Възлите са свързани чрез гигабитова мрежа "Myrinet".

C3 "Cray XD1". Състои се от 72 възела. Всеки възел има по два процесора "AMD Opteron" и по 4GB RAM. Процесорите работят на 2.4GHz. Процесорите са свързани с мрежа тип "Cray RapidArray" със скорост 5.6Gb/s.

Тъй като паралелните свойства на алгоритмите не зависят от типа на дискретизацията, представяме експерименти само за дискретизация MP. В Таблица 3.5 представяме времето за изпълнение на един процесор за двата варианта на паралелния преобусловител.

В Таблица 3.6 са показани относителните ускорения $S(p)$ и ефективности $E(p)$ за различни размерни на задачата n и различен брой процесори p. Включени са резултати и за двата варианта на паралелния MIC(0) преобусловител.

Таблица 3.5: Време за изпълнение на 1 процесор

	Вариант 1			Вариант 2		
n	C1	C2	C3	C1	C2	C3
32	52.18	30.87	29.47	28.16	18.61	21.18
64	578.4	336.8	347.6	336.1	228.4	224.2
128	6596	3793	3556	3887	2556	2610

Таблица 3.6: Паралелни ускорения и ефективности

		Вариант 1						Вариант 2					
		C1		C2		C3		C1		C2		C3	
n	p	$S(p)$	$E(p)$	$S(p)$	$E(p)$	$S(p)$	$E(p)$	$S(p)$	$E(p)$	$S(p)$	$E(p)$	$S(p)$	$E(p)$
32	2	1.49	0.74	1.31	0.66	1.77	0.88	1.93	0.96	1.33	0.66	1.97	0.99
	4	1.83	0.45	1.49	0.37	2.40	0.60	3.53	0.88	2.08	0.51	3.25	0.81
	8	2.11	0.26	1.22	0.15	3.34	0.42	5.78	0.72	3.07	0.38	5.20	0.65
	16	1.61	0.10	0.92	0.06	3.22	0.20	9.45	0.59	3.93	0.25	7.63	0.48
64	2	1.68	0.84	1.38	0.69	2.02	1.01	2.02	1.01	1.35	0.68	1.77	0.88
	4	2.46	0.61	1.98	0.49	3.17	0.79	3.92	0.98	2.49	0.62	3.50	0.87
	8	3.27	0.41	1.93	0.24	4.26	0.53	7.38	0.92	4.21	0.52	5.91	0.73
	16	3.78	0.23	2.06	0.13	6.03	0.38	12.83	0.81	6.53	0.40	8.64	0.54
128	2	1.82	0.91	1.51	0.76	1.56	0.78	2.00	1.00	1.49	0.74	1.93	0.96
	4	2.96	0.74	2.40	0.60	2.73	0.68	3.90	0.98	2.54	0.63	3.72	0.93
	8	4.50	0.56	2.70	0.34	5.34	0.67	7.33	0.92	4.59	0.57	7.30	0.91
	16	5.83	0.36	3.64	0.23	7.64	0.48	12.73	0.80	7.51	0.47	12.21	0.76

От таблица 3.6 се вижда че за Вариант 1 на преобусловителя, ускоренията, а съответно и ефективностите навсякъде са по-малки от тези при Вариант 2. Също така, ефективноста пада при всички експерименти с увеличаване на броя на процесорите, и се вдига при увеличаване размера на задачата. Това е в съгласие с оценките получени в Секция 3.6. Добри ускорения се получавт на системите C1 и C3 за Вариант 2 на преобусловителя, където имаме ефективности над 48% за най-малкия размер на задачата и за тестовете на 16 процесора. При по-големите размери ефективностите достигат съответно до 80% и до 76% за системите C1 и C3. От проведените експерименти ясно се вижда влиянието на скоростта на комуникационните мрежи върху времената за изпълнение. Най-лоши ефективности се получават на системата C2, която е и с най-бавната мрежа (1Gb/s).

Както очаквахме от теоретичните оценки за ускоренията за скаларната задача (2.6.58)(2.6.59) и тези за еластичната (3.6.26)(3.6.26) и като сравним ефективностите в Таблици 2.10 и 2.11 с тези в Таблица 3.6, почти навсякъде имаме по-добра ефективност при решаването на еластичната задача при решаването на задачи с един и същ размер n, на еднакъв брой процесори p за

компютърните системи C1 и C3. Забележително е, че това не е вярно за системата C2. Според нас, това се дължи на архитектурата на процесора Pentium 4 Xeon. При използване на системата от инструкции SSE2 (Streaming Single instruction multiple data Extensions 2) зареждането от паметта става на части от по 16 байта (две числа плаваща точка с двойна точност). Когато адресът от който се чете не е кратен на 16, се получават няколко „наказателни" такта в повече [24]. Нашият приоритет е бил решаването на възможно най-големи задачи, и затова не сме си позволили лукса да подравняваме структурите (в частност преместванията в всяка точка) на адреси кратни на 16, което би коствало $4/3$ повече памет.

Глава 4

Приложения

В тази глава са разгледани приложения на представения в глава 3 паралелен алгоритъм за решаване на системата на Ламе. Те включват симулация на еластични структури, и числена хомогенизация.

Резултати от тази глава са публикувани в [32, 54, 56, 60–63]:

- I. Georgiev, E. Ivanov, S. Margenov, and Y. Vutov. Numerical homogenization of epoxy-clay composite materials. *Numerical Methods and Applications*, *LNCS* **8962**, 130–137. Springer Berlin Heidelberg, 2015

- S. Margenov, S. Stoykov, and Y. Vutov. Numerical homogenization of heterogeneous anisotropic linear elastic materials. *Large-Scale Scientific Computing*, LNCS, 347–354. Springer Berlin Heidelberg, 2014

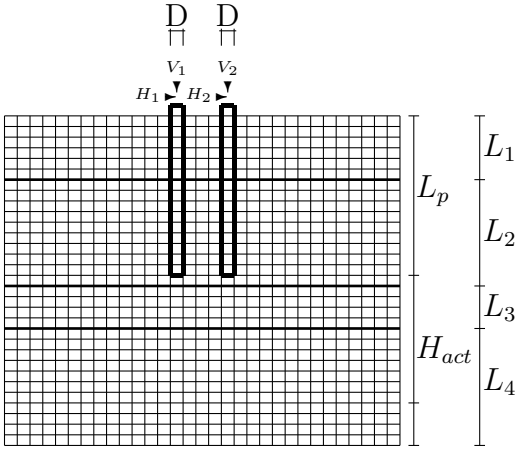
- I. Lirkov, Y. Vutov, M. Ganzha, and M. Paprzycki. Comparative Analysis of High Performance Solvers for 3D Elasticity Problems. *Numerical Methods and Applications*, *LNCS* **5434**, 392–399. Springer-Verlag, 2009

- S. Margenov and Y. Vutov. Parallel MIC(0) preconditioning for numerical upscaling of anisotropic linear elastic materials. *Large-Scale Scientific Computing*, *LNCS* **5910**, 805–812. Springer, 2010

- I. Lirkov, Y. Vutov, M. Paprzycki, and M. Ganzha. Parallel performance evaluation of MIC(0) preconditioning algorithm for voxel μFE simulation. *Parallel Processing and Applied Mathematics*, *LNCS* **6068**, 135–144. Springer, 2010

- S. Margenov and Y. Vutov. Parallel PCG algorithms for voxel FEM elasticity systems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 517–526, 2007

- S. Margenov and Y. Vutov. Preconditioning of voxel FEM elliptic systems. *TASK Quarterly*, 11(1-2):117–128, 2007



Фигура 4.1: Описание на задачата в сечение на изчислителната област Ω. $x_1^{max} = x_2^{max} = 37.2m$, $x_3^{max} = 31.0m$. $|H_1| = |H_2| = 150kN$, $|V_1| = 4000kN$, $|V_2| = 2000kN$, $E_{pile} = 31500MPa$, $\nu_{pile} = 0.2$, $E_{L_1} = 5.2MPa$, $\nu_{L_1} = 0.4$, $E_{L_2} = 9.4MPa$, $\nu_{L_2} = 0.35$, $E_{L_3} = 14.0MPa$, $\nu_{L_3} = 0.25$, $E_{L_4} = 21.4MPa$, $\nu_{L_4} = 0.2$.

4.1 Симулации на линейни еластични системи

4.1.1 Пилотни фундаменти

Типично приложение в практиката е симулацията на фундаментите на различни конструкции, които предават и разпределят общото натоварване върху почвата. Съвместната работа на конструкцията и почвата формират сложно напрегнато деформирано състояние, в областта на взаимодействие. При проектиране на съвременни конструкции се търси постигане на висока икономическа ефективност, при гарантирани надежност и функционалност. Точно в такива случаи, компютърното моделиране се оказва незаменим инструмент.

Тази инженерна задача се моделира чрез системата уравнения на Ламе (1.2.18).

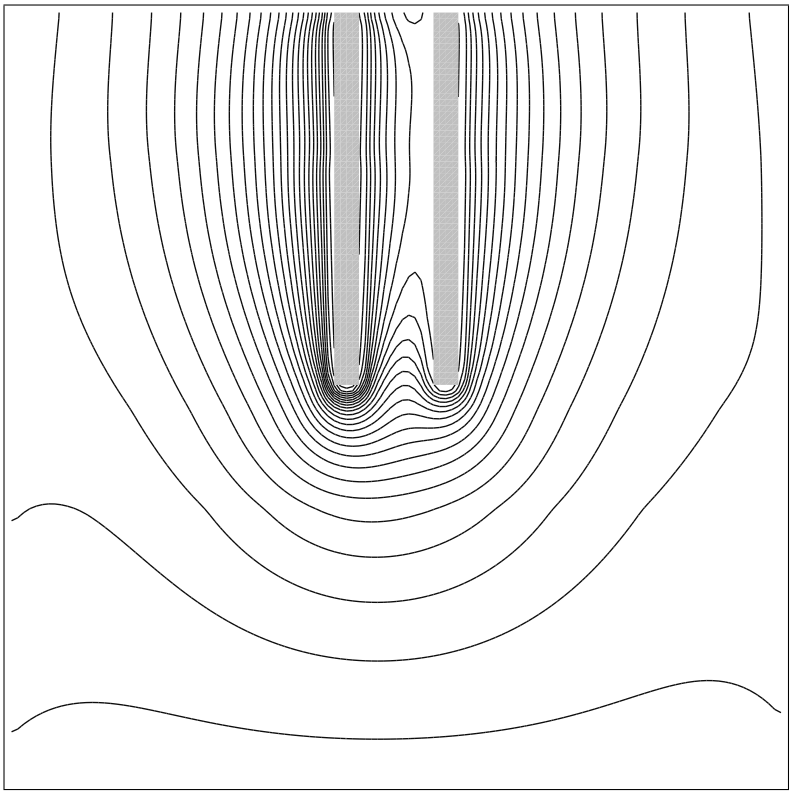
В разглеждания от нас модел се правят следните допускания: (1) преместванията са малки и (2) свойствата на материалите са изотропни. По-подробно описание на задачата може да бъде намерено в [51].

В разглежданата от нас задача, областта Ω е паралелепипед $\Omega = [0, x_1^{max}] \times [0, x_2^{max}] \times [0, x_3^{max}]$.

Тук за числените експерименти е използвана моделна задача от [31]. Задачата описва взаимодействието на два пилота в нехомогенна песъчливо-глинеста почва, вижте Фигура 4.1. С n е означен броят на елементите по всяко направление. Представени са експерименти за две мрежи – по-груба с елементи с размери $1.2 \times 1.2 \times 1$[m] и фина – с размери $0.6 \times 0.6 \times 0.5$[m].

Числени експерименти

За решение на задачата сме използвали, метода разработен в Глава 3. В статията [54] се сравнява използването на този метод с друг паралелен преобусловител на базата на блочна циркулантна факторизация [51, 52], за реша-



Фигура 4.2: Вертикалните премествания

Таблица 4.1: Експериментални резултати на Bassi

p	N	N_{it}	T_p	S_p	E_p	N	N_{it}	T_p	S_p	E_p
1	2 179 548	1 533	1 514.3			17 298 000	2 840	23 101.0		
2		1 533	795.3	1.90	0.952		2 840	11 584.3	1.99	0.997
4		1 533	414.9	3.65	0.912		2 840	5 973.1	3.87	0.967
8		1 533	217.7	6.95	0.869		2 840	3 219.0	7.18	0.897
16		1 533	125.9	12.02	0.752		2 840	1 684.5	13.71	0.857
32		1 533	74.1	20.43	0.638		2 840	913.2	25.30	0.791
64							2 840	544.4	42.43	0.663

ването на тази задача.

На Фигури 4.2, 4.3 и 4.4 са визуализирани вертикалните преместванията, деформациите и напреженията в едно сечение от изчислителната област. С изолинии са свързани точките с еднакви стойности.

Паралелните симулации са извършени на компютърни клъстери на NERSC (National Energy Research Scientific Computing Center - САЩ). За решаване на СЛАУ с N неизвестни на p процесора , представяме броя на итерациите N_{it}, необходимото време за решение T_p, ускорение $S_p = T_1/T_p$, и паралелната ефективност $E_p = S_p/p$. Поради ограничение върху използваните ресурси (процесорно време и използвана оперативна памет), за някои варианти нямаме последователни времена за голямата задача. Тогава сме сравнявали ефективността на базата на времето постигнато на 2 процесора. В Таблица 4.1 са показани резултатите на системата Bassi, представляваща клъстер от 111 осемпроцесорни IBM p575 POWER 5 възела. Пиковата производителност на всеки възел е

Фигура 4.3: Вертикалните деформации

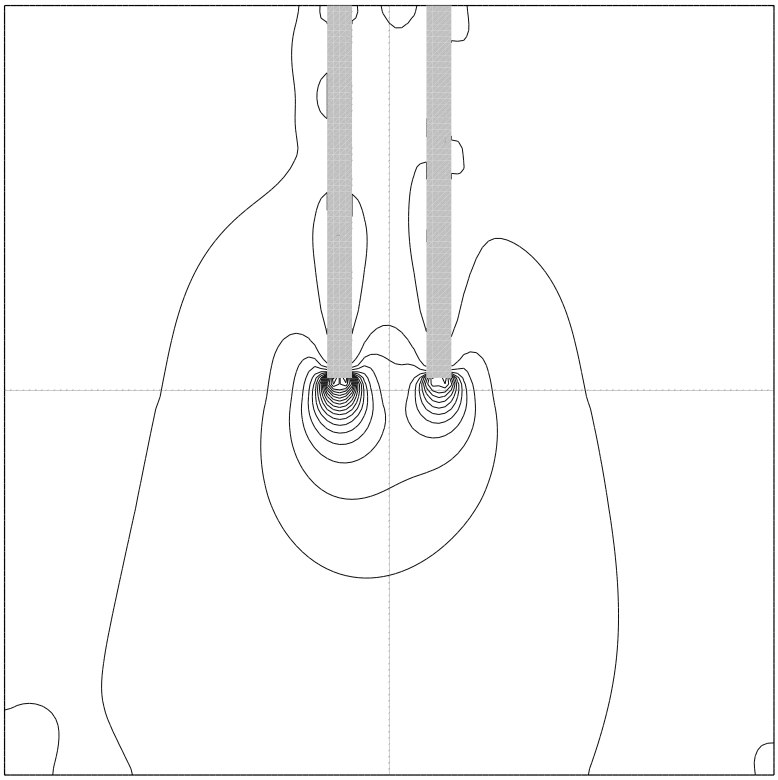
Таблица 4.2: Експериментални резултати на Franklin

p	N	N_{it}	T_p	S_p	E_p	N	N_{it}	T_p	S_p	E_p
1	2 179 548	1 959	3 787.7			17 298 000				
2		1 959	1 935.9	1.96	0.978		3 404	26 593.8		
4		1 959	1 001.6	3.78	0.945		3 404	13 712.8	0.970	
8		1 959	526.9	7.19	0.899		3 404	6 921.7	0.961	
16		1 959	286.9	13.20	0.825		3 404	3 573.8	0.930	
32		1 959	167.2	22.65	0.708		3 404	1 937.2	0.858	
64							3 404	1 115.7	0.745	

7.6 GFlop/s. На всеки възел има 32 GB оперативна памет. Възлите са свързани посредством IBM "Federation" HPS суичове, с по две връзки на възел. Използван е C++ компилатора на IBM.

В Таблица 4.2 са показани резултати получени на Cray XT4 системата Franklin. Тя е съставена от 9660 двупроцесорни възли, като на всеки възел има двуядрен AMD Opteron процесор, работещ на 2.6GHz. Пиковата му производителност е 5.2GFlop/s. Възлите са с по 4GB оперативна памет и са свързани чрез специализиран рутер „SeaStar2" директно с процесорната шина. Топологията на мрежата е тримерно-тороидална. Използван е C++ компилатора на PGI.

В Таблица 4.3 са показани резултати от изпълнението на системата Jacquard. Тя е клъстер, съставен от 356 двупроцесорни възела. Процесорите са двуядрени AMD Opteron на 2.2 GHz. Всеки възел е с по 6GB оперативна памет и



Фигура 4.4: Вертикалните напрежения

Таблица 4.3: Експериментални резултати на Jacquard

p	N	N_{it}	T_p	S_p	E_p
1	2 179 548	1 959	1 083.6		
2		1 959	535.7	2.023	1.011
4		1 959	301.4	3.594	0.899
8		1 959	180.7	5.997	0.750
16		1 959	117.0	9.264	0.579
32		1 959	81.7	13.260	0.414

са свързани чрез висикоскоростна InfiniBand мрежа. Използван е C++ компилаторът на PathScale. Ограничения във времето са причина за липсващи експерименти за по-фината мрежа.

Интересна особеност е разликата в получения брой итерации на различните компютърни системи. На системата Bassi те са чувствително по-малко. Например за малката задача, броят на итерациите на Bassi е 1533, а на другите две системи – 1959. За по-голямата система броят на итерациите е съответно 2840 и 3404. Това най-вероятно се дължи на различната микропроцесорна архитектура (Power 5 и . x86_64) и/или използвания компилатор.

Забележка 4.1.1. При пресмятане на скаларните произведения участващи в МСГП, неизменно се натрупват грешки от закръгления [34]. Сумирането на голям брой числа *силно* зависи от реда на извършване на операциите, както и от самите числа. Ние сме използвали алгоритъма на Кахан за сумиране [45]:

Алгоритъм 4.1. *Алгоритъм за сумиране на Кахан*

```
Sum:=0
Err:=0
for i:=1 to N do
{
  tmp1:= A[i] - Err;
  tmp2 := Sum + tmp1;
  Err := (tmp2-Sum) - tmp1;
  Sum := tmp2;
}
```

Този алгоритъм по същество натрупва грешката при сумиране в отделна променлива, увеличавайки на практика размера на мантисата двойно. Въпреки това грешки се натрупват. Възможно е агресивни оптимизации на компилатора, да неутрализират ефекта на Алгоритъм 4.1. Най-добрият начин за сумиране на много елементи е, положителните и отрицателните елементи да се отделят, да се сортират по отделно и след това да се съберат в ред на нарастване на абсолютната им стойност в отделни суматори. Това за съжаление е твърде времеотнемащо, затова и не е реализирано.

4.1.2 Костни микроструктури

Човешките кости, както и тези на много други гръбначни животни се състоят от два типа тъкан. Единият, наречен *компактна костна тъкан* формира външната част на костите, а другият - наричан *трабекуларна костна тъкан* – вътрешната. Трабекуларната кост се състои от множество свързани гредички (трабекули), които заграждат пространства в които се разполага костния мозък.

Трабекуларната кост има ниска плътност, и голяма повърхнина. Това я прави по-еластична, макар и по-слаба от компактната. Добре известно е, че трабекулите се ориентират по направление на натоварванията, (факт наричан закон на Волф [86]). Трабекуларната кост става по-здрава по направление на усилията, които изпитва. Вярно е и обратното, когато усилията намалеят, костната структура се разгражда и костта отслабва.

Остеопорозата е болест на костната система. Тя е свързана със загуба на костна маса, като загубата е най-осезаема в трабекуларната част. В следствие на намаляването на плътноста на твърдата фаза, костите стават по-податливи на счупвания. От болестта са силно застрашени жените в менопауза, както и двата пола над 75 годишна възраст.

От съществено значение е диагностиката и превенцията на остеопорозата. Основно средство за целта е рентгенографията, а напоследък и рентгеновата компютърна томография.

Определение 4.1.2. Рентгеновата компютърната томография *(от английски Computed tomography) е реконструкция на тримерен образ на базата на поредица от двумерни рентгенови снимки, направени около една и съща ос*

на въртене. Тримерния образ е съставен от (може да бъде разглеждан като) поредица двумерни изображения, но те вече отразяват не проекция на обекта върху екрана, а самата му структура.

Поради технологични особености на рентгенографията, трудно може да се прецени обективно костната плътност. Това по-лесно става при компютърната томография. Още повече, съществена роля играе и самата микро структура, така че, костната плътност сама по себе си не е еднозначно определящ фактор.

По-добър индикатор за наченки на остеопороза са напреженията във всяка една точка от костта. Получаваме ги с помощта на компютърната томография и компютърното моделиране. Прилагаме така наречения μМКЕ, при който всеки отделен воксел се представя с отделен краен елемент [3, 10].

Определение 4.1.3. Воксел *(от английски volume element, voxel) е съставна част от тримерен образ (тухличка). Тримерен аналог на пиксела.*

Тук прилагаме разработеният в Глава 3 паралелен алгоритъм за решаване на еластични задачи, за симулация на натоварване на трабекуларната част от костта. Геометрията е получена от тримерна компютърна томография на човешки гръбначен прешлен L3 [29], виж. Фигура 4.5. Всеки воксел е с форма на куб със страна 37μm.

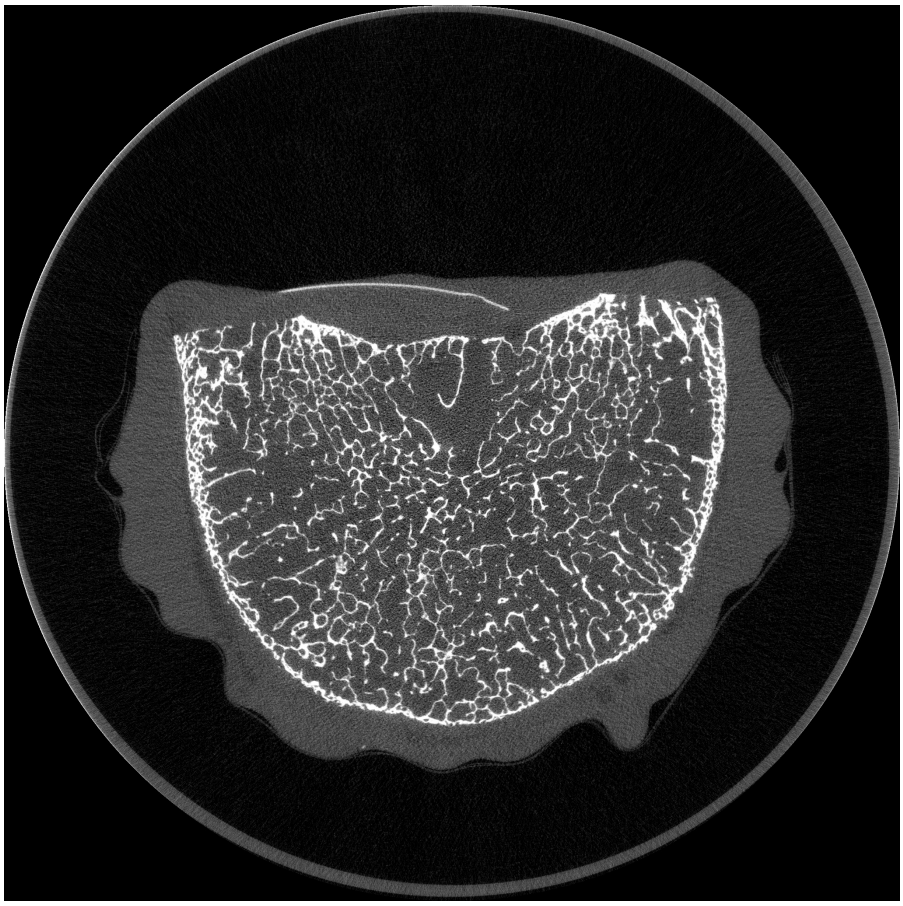
Като първа стъпка тримерното изображение се сегментира. Това означава за всеки воксел да се определи, към каква тъкан принадлежи. За нашите експерименти ние сме използвали бинарна сегментация – всички воксели с интензитет по-голям от фиксирана стойност маркираме като „кост“, а останалите – „не кост“.

След това сме „изрязали“ три образеца с размери $32 \times 32 \times 32$, $64 \times 64 \times 64$ и $128 \times 128 \times 128$ воксела. Оставащите „висящи“ парчета са премахнати. В горната и долната част и на трите изображения са поставени пластинки с дебелина един воксел, които служат за по-добра интерпретация на натоварването. Получените костни фрагменти са изобразени на Фигура 4.6

Числени експерименти

Важно е да отбележим, че микроструктурата на трабекуларната кост е типичен пример за силно хетерогенна среда. В нашата симулация разглеждаме трабекуларната кост, състояща се от твърда и флуидна част. Както споменахме по-горе, костния образец е поставен между две пластини. На долната пластина са наложени нулеви гранични условия на Дирихле – тоест тя е твърдо застопорена. Към горната пластина са наложени гранични условия на фон Нойман, съответстващи на приложена вертикално натоварване с интензитет $20[N/m^2]$

За симулацията са използвани следните коефициенти: $E_p = 200[GPa]$, $E_s = 20[GPa]$, $E_f = \zeta E_s$, $\zeta \in \{0.1, 0.01, 0.001\}$, а коефициента на Поасон за трите вида материали е взет $\nu = 0.3$. Тук с E_p е означен модула на Юнг за двете пластини, с E_s – този за твърдата част на костта, а коефициента на флуидната фаза E_f е изразен чрез този на твърдата посредством вариращ скок ζ на модула на Юнг .



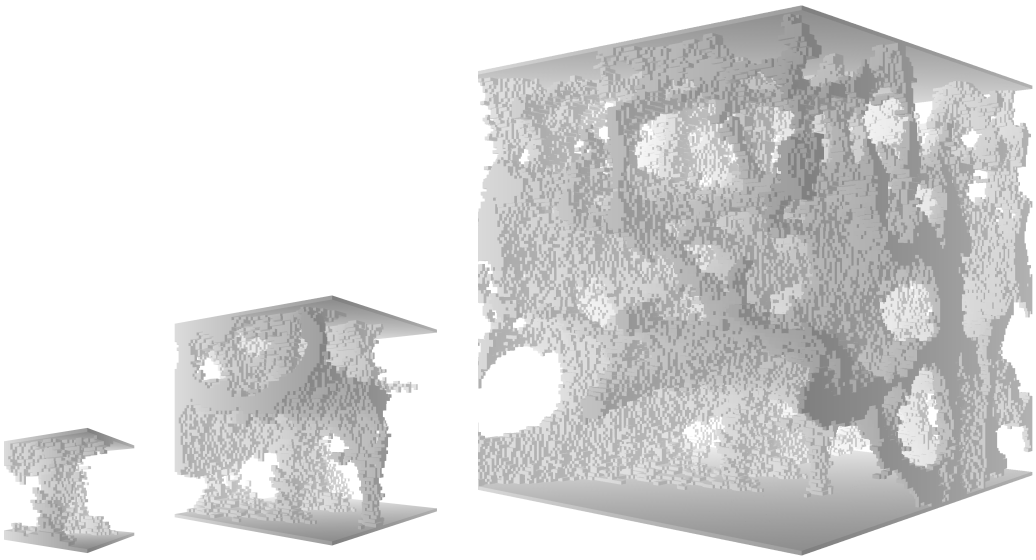
Фигура 4.5: Сечение от сканирания образ на прешлен

Резултати с вариращ параметър на скока ζ са представени в Таблица 4.5. Задачата е решена както с преобусловителя с паралелна непълна факторизация Вариант 2 (3.3.7), така и с този с алгебричен мултигрид (3.4.11). Трябва да отбележим, че заради особености при дискретизацията, можем да си позволим скокове на коефициентите само между различните макроелементи. Тоест за образеца с размери $m \times m \times m$ воксела, решаваме задачи с параметър $n = 2m$, като броят на неизвестните става $9n^2(n+1)$

При всички експерименти с алгебричния мултигрид е използвано загрубяване на Фалгот, V(1,1) цикъл с метод за релаксация хибриден Гаус-Зайдел. Параметъра за близост (AMG strength threshold) е избран 0.5, което е и препоръчваната стойност, от авторите на използваната библиотеката за тримерни задачи.

Като първи опит, бе използвано загрубяването по подразбиране с една вътрешна итерация на МСГП, но това се оказа твърде скъпо от към използвана оперативна памет. След това включихме опцията за агресивно загрубяване на първите две нива. Това доведе до забележимо намаляване на потребностите от оперативна памет.

За да получим сходимост на външната итерация по МСГП, трябваше да увеличим броя на вътрешните итерации. Таблица 4.4 обоснова нашия избор



Фигура 4.6: Структура на твърдата фаза на трабекуларна кост, различни екземпляри: $32 \times 32 \times 32$ – вляво, $64 \times 64 \times 64$ – по средата, и $128 \times 128 \times 128$ – вдясно

на параметри. В нея са показани следните данни: параметър за размера на задачата n; брой на процесорите p; брой на нивата с агресивно загрубяване NAC (0 означава без агресивно загрубяване), брой на външните итерации It_{out}, брой на вътрешните итерации It_{inn}, времето за изпълнение в секунди T; и необходимата оперативна памет M в мегабайти. Тестовете са за задача без скок на коефициентите. От Таблица 4.4 ясно се вижда, че въпреки увеличения брой

Таблица 4.4: Параметри на BoomerAMG

n	p	NAC	It_{out}	It_{inn}	$T[s]$	$M[MB]$
32	1	0	12	1	15.6	108
32	1	2	9	4	10.4	66
64	1	0	13	1	185.3	899
64	1	2	9	4	107.2	432
64	4	0	24	1	180.1	1225
64	4	2	10	4	58.5	664

вътрешни итерации, времената за изчисление намаляват, както и паралелната ефективност се подобрява, при използването на агресивно загрубяване.

В случая с най-голям скок на коефициентите (ζ=0.001), за най-голямата задача (N=151 584 768), външната итерация по МСГП с мултигрид преобусловителя не се сходи за определения времеви лимит от 7 200 секунди. Този тест бе повторен с увеличен брой на вътрешни итерации, така, резултатите в таблицата са за брой на вътрешните итерации $It_{in} = 6$.

От Таблица 4.5 се вижда, че броят на итерациите за MIC(0) преобусловителя нараства по-бързо при увеличаване на размера на задачата, когато има

Таблица 4.5: Паралелни тестове

		$\zeta = 1$				$\zeta = 0.1$				$\zeta = 0.01$				$\zeta = 0.001$			
		MIC(0)		AMG		MIC(0)		AMG		MIC(0)		AMG		MIC(0)		AMG	
n	p	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It		
64	1	91.2	122	156.2	13	239.3	330	374.9	27	348.3	505	757.9	57	588.6	823	1040.5	78
128	8	221.1	181	293.6	13	833.2	708	681.0	25	975.5	830	1501.3	60	2166.7	1850	2908.9	107
256	64	537.6	272	492.8	13	2393.8	1237	945.4	25	3495.7	1831	2114.4	57	6025.8	3150	5520.1	114

скокове, в сравнение със случая без скок (ζ=1). Броят на външните итерации при мултигрид преобусловителя се вижда, че почти не зависи от размера на задачата, освен в случая на най-голям скок ($\zeta = 0.001$), където се наблюдава увеличение. За най-малката задача (N=2 396 160) MIC(0) преобусловителят е по-бърз от мултигридския за всички стойности на ζ. За средната по размер задача (N=19 021 824) MIC(0) е по-бърз при всички тестове, освен при $\zeta = 0.1$. За най-голямата задача мултигридът е по-бърз, но превъзходството му намалява с увеличаване на скока. Причината за това е, че външните итерации при мултигрида са много по-скъпи от тези при непълната факторизация, което от своя страна води до влошаване на времената.

Експериментите са проведени по следния начин. Заедно с размера на задачата (броят на неизвестните) увеличаваме пропорционално и броя на процесорите на които я решаваме. Ако методът ни за решение е оптимален, то би трябвало времето за решение да остане едно и също. Виждаме, че въпреки оптималния брой външни итерации, дори преобусловителят с алгебричен мултигрид не успява да запази времената. Това се дължи не само на не идеалната паралелна ефективност, но и на различната структура на задачата. Същото важи и за преобусловителя с непълна факторизация, но там заради увеличаващият се брой итерации не можем да очакваме запазване на времето.

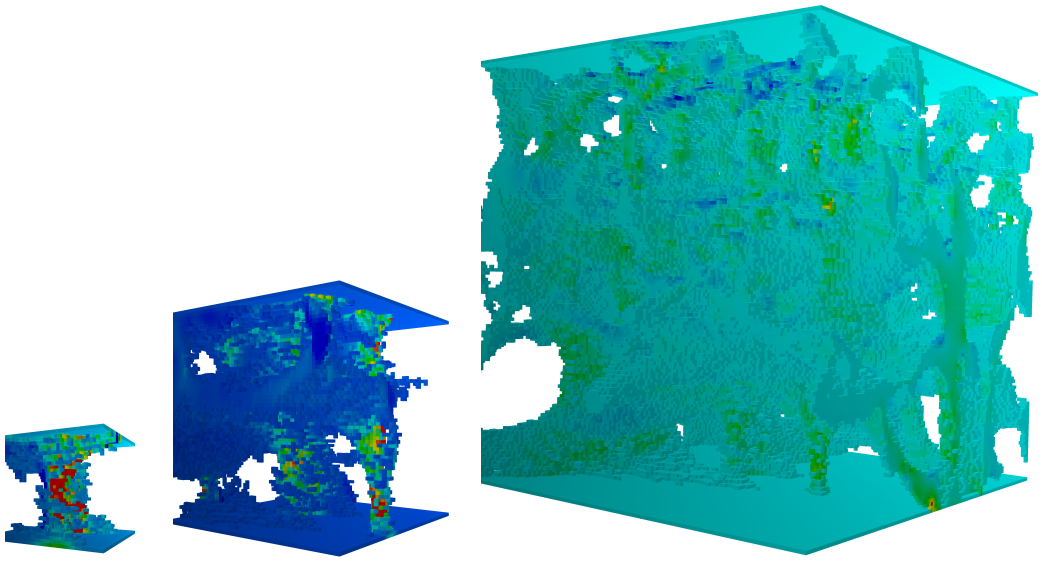
Можем да направим извода, че и двата преобусловителя предоставят стабилна платформа за компютърни симулации на костна структура. И двата преобусловителя имат плюсове и минуси, в зависимост от размера на задачата, както и от костната микроструктура.

След решаване на задачите получаваме преместванията. От тях можем да пресметнем деформациите, а от деформациите – напреженията. На Фигура 4.7 са изобразени напреженията за трите решени задачи, в случая на най-голям скок ζ.

4.2 Числена хомогенизация

В практиката често се налага да се решават задачи върху неоднородни структури, чиито характеристики силно се различават в зависимост от мащаба. Един такъв пример – трабекуларната кост – разгледахме в предишния раздел.

Предполагаме, че трабекуларната кост е съставена от хомогенни изотропни материали на микро ниво. При по-голямо увеличаване на мащаба тъканта няма видима структура. Свойствата на материала на микро ниво, както и микроструктурата, намират отражение в еквивалентните (може и анизотропни)



Фигура 4.7: Напрежения: $32 \times 32 \times 32$ – вляво, $64 \times 64 \times 64$ – по средата, и $128 \times 128 \times 128$ – вдясно

свойства на материала на макро ниво.

В днешно време въпреки, експоненциалното развитие на производителността на изчислителната техника, остава невъзможно директното симулиране на макро обекти с дискретизация на нивото на вокселното представяне на сложната микроструктура. Именно в такива случаи се прибягва до числена хомогенизация. Това е метод за извличане на макро характеристики, взимайки под внимание микроструктурата на разглеждания обект.

4.2.1 Описание на метода

В този раздел разглеждаме един метод за числена хомогенизация, който намира ефективният еластичния тензор, за даден композитен образец. С други думи намира свойствата на такъв хомогенен (еднороден) материал, който има същото поведение, като разглеждания композитен (съставен) образец.

Нека Ω е паралелепипедална област представляваща разглеждания образец и $\mathbf{u} = (u_1, u_2, u_3)$ е вектор от преместванията в Ω. Тогава компонентите на тензора на малките деформации са:

$$\varepsilon_{ij}\left(\mathbf{u}\left(\mathbf{x}\right)\right) = \frac{1}{2}\left(\frac{\partial u_i(\mathbf{x})}{\partial x_j} + \frac{\partial u_j(\mathbf{x})}{\partial x_i}\right) \tag{4.2.1}$$

Ние приемаме че е в сила закона на Хук, тоест че тензорът на напреженията може да бъде изразен чрез тензора на деформациите по следния начин: $\sigma_{ij} = s_{ijkl}\varepsilon_{kl}$, Тук, както и по нататък в този раздел се използва сумиране по повтарящите се индекси. Тензорът от четвърти ред s се нарича тензор на коравина. В сила са следните симетрии:

$$s_{ijkl} = s_{jikl} = s_{ijlk} = s_{klij}. \tag{4.2.2}$$

Закона на Хук, може да се запише и в следния матричен вид:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \begin{bmatrix} s_{1111} & s_{1122} & s_{1133} & s_{1123} & s_{1113} & s_{1112} \\ s_{2211} & s_{2222} & s_{2233} & s_{2223} & s_{2213} & s_{2212} \\ s_{3311} & s_{3322} & s_{3333} & s_{3323} & s_{3313} & s_{3312} \\ s_{2311} & s_{2322} & s_{2333} & s_{2323} & s_{2313} & s_{2312} \\ s_{1311} & s_{1322} & s_{1333} & s_{1323} & s_{1313} & s_{1312} \\ s_{1211} & s_{1222} & s_{1233} & s_{1223} & s_{1213} & s_{1212} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{bmatrix}. \tag{4.2.3}$$

Тук симетричната матрица S с размери 6×6 (4.2.3) се нарича матрица на коравина.

Определение 4.2.1. *Ортотропен материал наричаме материал притежаващ три взаимно перпендикулярни равнини на симетрия.*

За изотропен материал, матрицата S, а съответно и тензорът s имат само две независими степени на свобода. За ортотропни материали матрицата S има девет независими степени на свобода. В общия анизотропен случай S има 21 независими степени на свобода.

Разработеният в дисертацията алгоритъм следва подхода от [42], вижте също така и [13, 60]. Хомогенизационната схема изисква намирането на Ω-периодични функции $\boldsymbol{\xi}^{kl} = (\xi_1^{kl}, \xi_2^{kl}, \xi_3^{kl})$, k, $l = 1, 2, 3$, удовлетворяващи следната задача в слаба формулировка:

$$\int_\Omega \left(s_{ijpq}(x) \frac{\partial \xi_p^{kl}}{\partial x_q} \right) \frac{\partial \phi_i}{\partial x_j} d\Omega = \int_\Omega s_{ijkl}(x) \frac{\partial \phi_i}{\partial x_j} d\Omega, \tag{4.2.4}$$

за всяка Ω-периодична функция $\boldsymbol{\phi} = (\phi_1, \phi_2, \phi_3)$, $\phi_i \in H^1(\Omega)$. След пресмятането на характеристичните премествания $\boldsymbol{\xi}^{kl}$, от (4.2.4) можем директно да пресметнем хомогенизирания еластичен тензор s_{ijkl}^Hчрез следната формула:

$$s_{ijkl}^H = \frac{1}{|\Omega|} \int_\Omega \left(s_{ijkl}(x) - s_{ijpq}(x) \frac{\partial \xi_p^{kl}}{\partial x_q} \right) d\Omega. \tag{4.2.5}$$

От (4.2.4) и благодарение на симетриите в тензора на коравина (4.2.2), имаме $\boldsymbol{\xi}^{kl} = \boldsymbol{\xi}^{lk}$. Следователно се налага решаването само на 6 задачи (4.2.4) за намирането на хомогенизирания тензор на коравина.

За решаване на задачите (4.2.4) тук използваме RSI дискретизацията (3.1.2), както и паралелния алгоритъм разработен в Глава 3.

Периодичността на решението налага използване на периодични гранични условия за задачите (4.2.4). В резултат на това матрицата на получената линейна система става положително полуопределена и се налага промяна в алгоритъма за реализация на МСГП. В раздел 4.2.3 от настоящата глава е описана модификацията на МСГП за неопределени матрици, а в раздел 4.2.4 са скирицрани съответните реализации на преобуславянето и умножението на матрица по вектор.

4.2.2 Главни направления на анизотропия

Тъй като тензорът на коравина за анизотропии материали зависи от координатната система, важно за сравнения и класификации на материалите е трансформацията на тензора по главните направления на анизотропия. Това води до еднозначно записване на всеки тензор на коравина.

Казваме че осите на дадена координатна система съвпадат с главните направления на анизотропия, когато равномерно разтягане във всички посоки, формира напрегнато състояние без тангенциални напрежения.

За намиране на главните направления на анизотропия (ГНА) ще следваме техниката описана в [70]. Нека представим тензора на обемно напрежение:

$$K = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix}. \tag{4.2.6}$$

Елементите на K се дефинират по следния начин:

$$K_{ij} = \sum_{k=1}^{3} s_{ijkk}. \tag{4.2.7}$$

Записваме равномерното разтягане във всички посоки по следния начин:

$$\varepsilon_{ij} = \tilde{\varepsilon}\delta_{ij}, \tag{4.2.8}$$

където $\tilde{\varepsilon}$ е фиксирана константна деформация, а с δ_{ij} е означен символът на Кронекер. Тогава компонентите на тензора на напреженията са

$$\sigma_{ij} = K_{ij}\tilde{\varepsilon}. \tag{4.2.9}$$

Следователно главните направления на тензора K съвпадат с главните направления на тензора на напреженията σ_{ij}. Напреженията по главните направления са:

$$\sigma_{ij} = \lambda_i\tilde{\varepsilon}\delta_{ij}, \tag{4.2.10}$$

където с λ_i са означени собствените стойности на тензора K. За да осигурим еднозначност на трансформацията, подреждаме собствените стойности по големина $\lambda_1 \leq \lambda_2 \leq \lambda_3$, тоест най-голямата собствена стойност е последната, а първата е най-малката. По този начин, най-здравото направление на материала съвпда с оста z, а най-слабото – с оста x. Когато имаме една или повече равни собствени стойности, това съответства на материали с еднакви свойства по различните направления. Трансформационната матрица T, която завърта текущата координатната система в координатната система на ГНА се определя от ортонормираните собствени вектори v^i на K:

$$T = \begin{bmatrix} v_1^1 & v_2^1 & v_3^1 \\ v_1^2 & v_2^2 & v_3^2 \\ v_1^3 & v_2^3 & v_3^3 \end{bmatrix}. \tag{4.2.11}$$

Вече можем да извършим и самата трансформация по следната формула:

$$\bar{s}_{klst} = s_{mnpr}T_{km}T_{ln}T_{sp}T_{tr}. \tag{4.2.12}$$

4.2.3 Сходимост на МСГП за полуопределени матрици

Както споменахме по-горе при налагане на периодични гранични условия матрицата на коравина става положително полуопределена. Ако векторът $\mathbf{u}$ е решение на задачата

$$A\mathbf{u} = f, \tag{4.2.13}$$

то решения ще бъдат и всички вектори $\mathbf{u}^*$:

$$\mathbf{u}^* = \mathbf{u} + \mathbf{v},\ \mathbf{v} \in KerA. \tag{4.2.14}$$

Ако знаем ядрото на оператора A, то съществува модификация на МСГП, която се схожда и за съответната положително полуопределена матрици [15].

Нека стълбовете на матрицата V образуват базис на ядрото на матрицата A. Тогава линейния оператор $P : \mathbb{R}^N \to ImA$ дефиниран по следния начин

$$P = I - V(V^TV)^{-1}V^T \tag{4.2.15}$$

проектира векторите от $\mathbb{R}^N$ върху образа на A. За системата уравнения на Ламе, ядрото на оператора се определя от движенията на твърдото тялое (без промяна на взаимното положение между точките от тялото) – транслации и ротации. Така че в нашият случай, матрицата V в уравнението (4.2.15) се състои от 6 вектора три съответстващи на транслации по всяко едно от направленията, и три на ротации около една от трите оси.

При тези предположения, модифицирания МСГП има вида:

Алгоритъм 4.2. *Метод на спрегнатия градиент с преобуславяне за положително полуопределена матрица A (МСГПН),* $P : \mathbb{R}^N \to ImA$
$\mathbf{x} = МСГПН(A, \mathbf{b})$

$$\begin{aligned}
\mathbf{x}^0 &= 0,\ \mathbf{g}^0 = P(A\mathbf{x}^0 - \mathbf{b}),\ \mathbf{h}^0 = P(C^{-1}\mathbf{g}^0),\ \mathbf{d}^0 = -\mathbf{h}^0,\ \gamma_0 = \left(\mathbf{g}^0, \mathbf{h}^0\right)\\
k &= 0,\ 1,\ 2, \ldots\\
\mathbf{t}^k &= A\mathbf{d}^k\\
\tau_k &= \frac{\gamma_k}{(\mathbf{d}^k, \mathbf{t}^k)}\\
\mathbf{x}^{k+1} &= \mathbf{x}^k + \tau_k\mathbf{d}^k\\
\mathbf{g}^{k+1} &= P(\mathbf{g}^k + \tau_k\mathbf{t}^k)\\
\mathbf{h}^{k+1} &= P(C^{-1}\mathbf{g}^{k+1})\\
\gamma_{k+1} &= \left(\mathbf{g}^{k+1}, \mathbf{h}^{k+1}\right)\\
\beta_k &= \frac{\gamma_{k+1}}{\gamma_k}\\
\mathbf{d}^{k+1} &= \beta_k\mathbf{d}^k - \mathbf{h}^{k+1}
\end{aligned}$$

Всички представени по-нататък експерименти за числено хомогенизиране използват МСГПН.

Нека отбележим само, че обръщаната матрица в (4.2.15) е с размери 6×6. Нейна LU факторизация може да бъде намерена в началото, така че проекцията P приложена върху вектор $\mathbf{v}$ се свежда до 6 скаларни произведения (умножението на V^T с вектора $\mathbf{v}$, обратен ход на Гаус за LU факторизираната матрица и 6 векторни операции тип умножение на вектор с число и изваждане от друг вектор.

4.2.4 Алгоритмична реализация

Периодичните гранични условия променят и структурата на матрицата на коравина. Броят на неизвестните намалява и става равен на

$$N = 9n_1n_2n_3, \tag{4.2.16}$$

където с $n_i, i = 1, 2, 3$ сме означили броя на елементите по всяко едно направление. Ние, обаче сме запазили структурата както на матрицата на коравина, така и на векторите участващи при пресмятанията. Тоест дублираме неизвестните по границите, както и съответните редове на матрицата A. Тогава при извършване на умножението на матрица по вектор

$$\mathbf{r} = A\mathbf{v} \tag{4.2.17}$$

първо копираме елементите на $\mathbf{v}$ на съответните им места на срещуположните страни на областта. След това извършваме умножението по нормалния начин и накрая събираме стойностите съответстващи на едно и също неизвестно в резултата $\mathbf{r}$. И при двете операции се налага една допълнителна комуникация между първия и последния процесор, като тези комуникации се извършват заедно с останалите комуникации за умножение на матрица по вектор.

Колкото до непълната факторизация MIC(0) диагоналната пертурбация, която правим, гарантира успешно конструиране на преобусловителя. И тука се налага допълнителна комуникация между първия и последния процесор на всяка една стъпка при решаване на системата с преобусловителя, но и тук тези комуникации се извършват заедно с останалите.

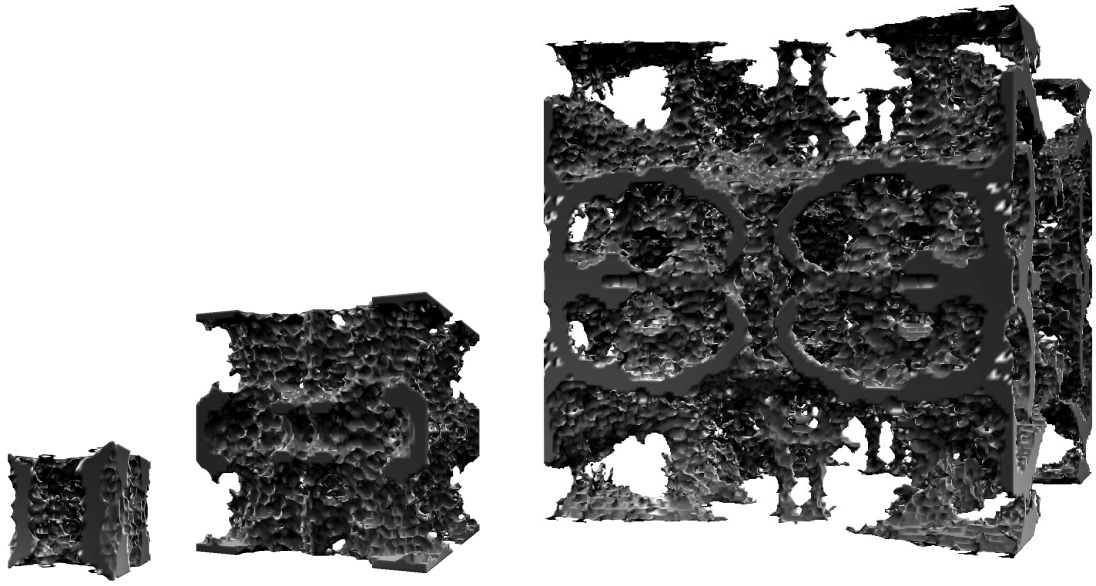
4.2.5 Числени експерименти

Хомогенизация на трабекуларни тъкани

В тази серия от експерименти ще намерим хомогенизирания тензор на коравина за трабекуларна кост. Ще вземем под внимание само твърдата фаза на костта. За целта ще използваме така наречения метод на фиктивната област.

Нека отбележим, че за ортотропни материали, тензорът на коравина има 9 степени на свобода. Три модула на Юнг E_1, E_2, E_3, три коефициента на Поасон $\nu_{12}, \nu_{23}, \nu_{31}$ и три модула на срязване $\mu_{12}, \mu_{23}, \mu_{31}$. Матрицата на коравина за ортотропни материали тогава може да бъде записана в следния вид

$$S = \delta \begin{bmatrix} \frac{1-\nu_{23}\nu_{32}}{E_1} & \frac{\nu_{12}+\nu_{31}\nu_{23}}{E_1} & \frac{\nu_{31}+\nu_{21}\nu_{32}}{E_1} & & & \\ \frac{\nu_{12}+\nu_{13}\nu_{32}}{E_2} & \frac{1-\nu_{31}\nu_{13}}{E_2} & \frac{\nu_{32}+\nu_{31}\nu_{12}}{E_2} & & & \\ \frac{\nu_{13}+\nu_{12}\nu_{23}}{E_3} & \frac{\nu_{23}+\nu_{13}\nu_{21}}{E_3} & \frac{1-\nu_{12}\nu_{21}}{E_3} & & & \\ & & & \frac{\mu_{23}}{\delta} & & \\ & & & & \frac{\mu_{31}}{\delta} & \\ & & & & & \frac{\mu_{12}}{\delta} \end{bmatrix}, \tag{4.2.18}$$



(a) (b) (c)

Фигура 4.8: Структура на твърдата част от костта (a) $n = 32$, (b) $n = 64$, (c) $n = 128$.

където

$$\delta = 1 - \nu_{12}\nu_{21} - \nu_{13}\nu_{31} - \nu_{23}\nu_{32} - 2\nu_{12}\nu_{23}\nu_{31},$$
$$\frac{\nu_{12}}{E_1} = \frac{\nu_{21}}{E_2}, \qquad \frac{\nu_{23}}{E_2} = \frac{\nu_{32}}{E_3}, \qquad \frac{\nu_{31}}{E_3} = \frac{\nu_{13}}{E_1}.$$

Анализираните костни проби са получени от компютърна томография [29]. Размерът на вокселите е 37 μm. За да получим периодична структура, сме използвали три пъти симетрия около три равнини, вижте Фигура 4.9.

Изследвали сме свойствата на три образеца с размери $n \times n \times n$, където $n \in \{32, 64, 128\}$, Виж. Фигура 4.9. Модулът на Юнг и коефициента на Поасон за твърдата част от костта са взети от [25] и са $E^s = 14.7\text{GPa}$ и $\nu^s = 0.325$. За да вземем под внимание само твърдата част, от костта, интерпретираме флуидната част като фиктивна област. За целта взимаме модула на Юнг за фиктивната област да е $E^f = \zeta E^S$, където намаляваме параметъра ζ експоненциално. Също така използваме и $\nu^f = \nu^s$, което не влияе на резултатите от хомогенизацията.

Критерия за край на итерацията при всички експерименти е

$$||\mathbf{r}^j||_{C^{-1}}/||\mathbf{r}^0||_{C^{-1}} < 10^{-6},$$

където $\mathbf{r}^j$ е резидула на j-тата итерация, а с C е означен използвания преобусловител.

Числените експерименти са проведени на IBM Blue Gene/P компютър намиращ се в Българския център по суперкомпютърни приложения. Той се състои от PowerPC процесори работещи на 850 MHz. Всеки възел има 2GB оперативна памет. Възлите са свързани чрез няколко специализирани високоскоростни мрежи – тримерна тороидална мрежа за локалните комуникации,

дървовидна мрежа за глобалните операции (разпръскване, редукция), гигабитов етернет за управление и входно-изходни операции, мрежа за бариери и прекъсвания.

Използвали сме 2 процесора за решаване на най-малката задача ($n = 32$), и сме увеличавали броя на процесорите пропорционално с броя на неизвестните за по-големите задачи. В Таблица 4.6 са събрани времената T и броя на итерациите It използвайки MIC(0) преобусловителя, Вариант 2 за една от шестте задачи (4.2.4). Показани са резултати за трите образеца и за $\zeta \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. Броят на неизвестните е означен с N, а броят на процесорите – с p. Същата информация, за мултигрид преобусловителя е представена в Таблица 4.7.

Таблица 4.6: MIC(0)

			$\zeta = 10^{-1}$		$\zeta = 10^{-2}$		$\zeta = 10^{-3}$		$\zeta = 10^{-4}$		$\zeta = 10^{-5}$	
n	N	p	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It
32	2 359 296	2	267	219	455	378	689	576	816	684	836	701
64	18 874 368	16	442	320	804	588	1 235	907	1 683	1 239	1 937	1 426
128	150 994 944	128	937	462	1 715	851	2 939	1 462	3 953	1 969	4 634	2 309

Таблица 4.7: BoomerAMG

			$\zeta = 10^{-1}$		$\zeta = 10^{-2}$		$\zeta = 10^{-3}$		$\zeta = 10^{-4}$		$\zeta = 10^{-5}$	
n	N	p	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It	$T[s]$	It
32	2 359 296	2	596	30	989	51	1 385	73	1 644	86	1 682	88
64	18 874 368	16	669	30	1 212	56	1 736	81	2 238	105	2 447	115
128	150 994 944	128	1 114	39	1 706	61	2 566	93	3 292	120	3 856	141

Както виждаме, броят на итерациите и за двата преобусловителя расте с намаляването на ζ. Броят на итерациите за моделни задачи при MIC(0)преобусловител расте както $n_{it} = O(n^{1/2}) = O(N^{1/6})$. Тука итерациите се държат по подобен начин за $\zeta \in \{10^-1, 10^-2\}$. Въпреки това, даже за големи скокове на коефициентите, сходимостта става само малко по-лоша.

Наблюдава се и малко увеличаване на броя на итерациите за мултигрид преобусловителя. Виждаме сравними паралелни времена и за двата преобусловителя. Мултигридът има преимущество за най-голямата задача и при големи скокове на коефициентите.

Структурата на получените тензори на коравина S^H във всичките случаи съответстват на ортотропен материал. Това се дължи на огледалната структура на образците. От (4.2.18) могат да бъдат намерени модулите на Юнг по трите направления E_i и съответните коефициенти на Поасон ν_{ij} по формулите

$$E_i = 1/c_{ii} \qquad \nu_{ij} = -E_i c_{ji},$$

където c_{ij} са елементите на матрицата $C = (S^H)^{-1}$ [75].

Таблици 4.8, 4.9 и 4.10 съдържат пресметнатите хомогенизирани модули на Юнг E, коефициенти на Поасон ν и модули на срязване μ за трите образеца, в зависимост от параметъра ζ.

Таблица 4.8: Хомогенизирани свойства – $n = 32$

ζ	E_1	E_2	E_3	ν_{12}	ν_{23}	ν_{31}	μ_{23}	μ_{31}	μ_{12}
10^{-1}	4.52×10^{9}	6.23×10^{9}	6.24×10^{9}	0.208	0.300	0.286	2.29×10^{9}	1.39×10^{9}	1.35×10^{9}
10^{-2}	2.03×10^{9}	4.72×10^{9}	4.67×10^{9}	0.095	0.271	0.229	1.73×10^{9}	4.81×10^{8}	3.80×10^{8}
10^{-3}	1.67×10^{9}	4.48×10^{9}	4.45×10^{9}	0.074	0.264	0.212	1.66×10^{9}	3.56×10^{8}	2.42×10^{8}
10^{-4}	1.63×10^{9}	4.46×10^{9}	4.42×10^{9}	0.072	0.263	0.210	1.65×10^{9}	3.42×10^{8}	2.26×10^{8}
10^{-5}	1.62×10^{9}	4.45×10^{9}	4.42×10^{9}	0.071	0.262	0.210	1.65×10^{9}	3.40×10^{8}	2.24×10^{8}

Таблица 4.9: Хомогенизирани свойства – $n = 64$

ζ	E_1	E_2	E_3	ν_{12}	ν_{23}	ν_{31}	μ_{23}	μ_{31}	μ_{12}
10^{-1}	2.86×10^{9}	3.11×10^{9}	3.55×10^{9}	0.288	0.270	0.281	1.19×10^{9}	9.07×10^{8}	9.50×10^{8}
10^{-2}	8.73×10^{8}	1.12×10^{9}	1.94×10^{9}	0.191	0.164	0.185	4.94×10^{8}	1.62×10^{8}	1.71×10^{8}
10^{-3}	5.69×10^{8}	8.02×10^{8}	1.73×10^{9}	0.127	0.124	0.117	3.90×10^{8}	5.22×10^{7}	5.22×10^{7}
10^{-4}	5.33×10^{8}	7.62×10^{8}	1.71×10^{9}	0.117	0.119	0.102	3.77×10^{8}	3.88×10^{7}	3.77×10^{7}
10^{-5}	5.29×10^{8}	7.58×10^{8}	1.71×10^{9}	0.116	0.118	0.101	3.76×10^{8}	3.74×10^{7}	3.62×10^{7}

Таблица 4.10: Хомогенизирани свойства – $n = 128$

ζ	E_1	E_2	E_3	ν_{12}	ν_{23}	ν_{31}	μ_{23}	μ_{31}	μ_{12}
10^{-1}	2.66×10^{9}	2.47×10^{9}	2.67×10^{9}	0.315	0.284	0.278	8.76×10^{8}	8.78×10^{8}	8.87×10^{8}
10^{-2}	7.90×10^{8}	5.97×10^{8}	9.51×10^{8}	0.282	0.180	0.171	1.93×10^{8}	1.68×10^{8}	1.64×10^{8}
10^{-3}	4.65×10^{8}	2.81×10^{8}	7.10×10^{8}	0.228	0.114	0.094	8.46×10^{7}	6.22×10^{7}	3.37×10^{7}
10^{-4}	4.20×10^{8}	2.24×10^{8}	6.78×10^{8}	0.222	0.100	0.076	6.66×10^{7}	4.89×10^{7}	1.40×10^{7}
10^{-5}	4.15×10^{8}	2.16×10^{8}	6.75×10^{8}	0.222	0.098	0.073	6.44×10^{7}	4.75×10^{7}	1.18×10^{7}

Постига се добра точност за хомогенизираните коефициенти, когато се използва за фиктивната област коефициент $E^f = \zeta E^s$ за $\zeta \in \{10^{-4}, 10^{-5}\}$. И при трите образеца се наблюдава ясно изразена ортотропия. Това опровергава хипотезата, че трабекуларната кост може да бъде разглеждана като изотропен материал.

Хомогенизация на полимерни нанокомпозити

Полимерните нанокомпозити представляват комбинация от полимерна матрица и пълнители, размерът на поне някой от които е в нанометровата област. Силикатните нанокомпозити имат многобройни приложения, като олекотени структурирани композити за авио- и авто-индустриите Те се отличават с много добри механични свойства – модул на Юнг, увеличена здравина, бариерни свойства, негоримост, електропроводимост и т.н. с цената на добавянето на малко количество нано частици[67, 68]. Това се дължи на голямата повърхност на взаимодействие между полимерната матрица и нано пълнителите.

Епоксидните смоли са широко използвани заради отличните химическа и топлинна устойчивост, висока сила на адхезия, висока коравина, както и електрическа изолация.

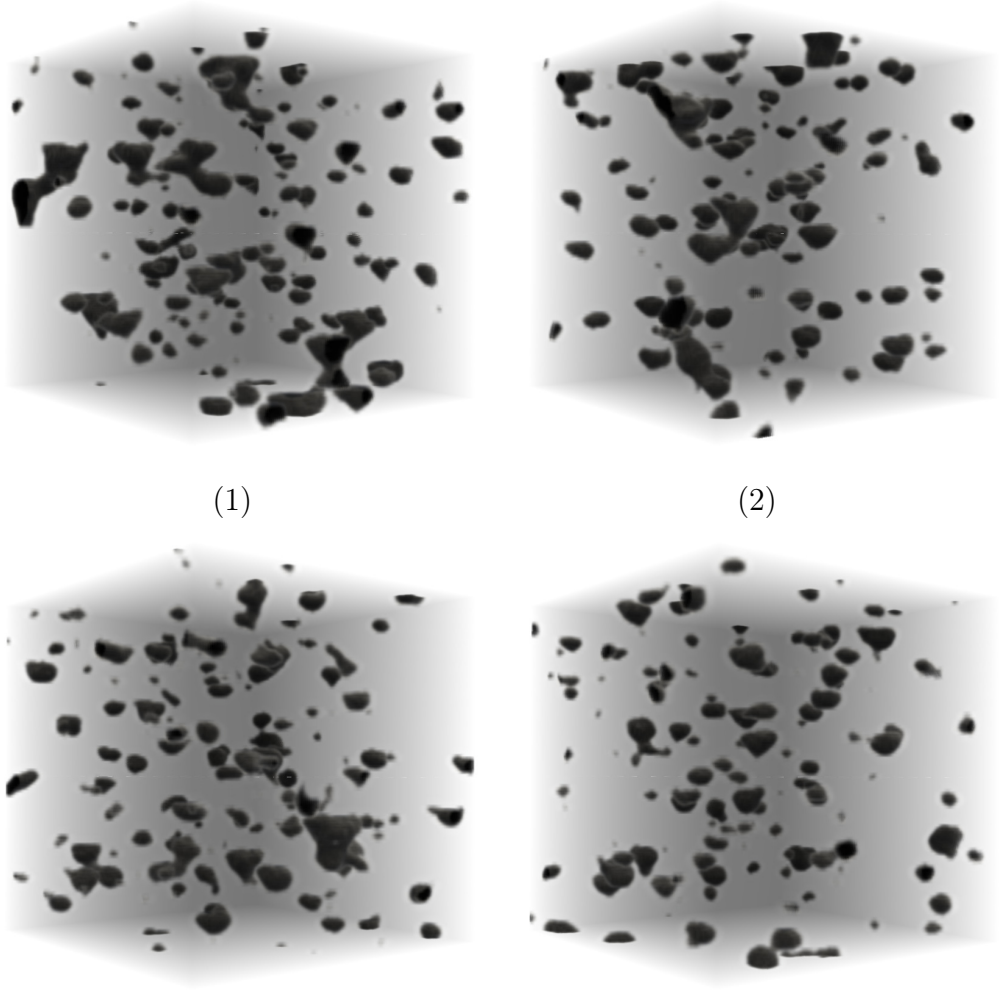
Трикомпонентните хибридни нанокомпозити, като органично модифицирана глина и златни наночастици в полимерна матрица са изследвани с цел получаване на нови материали с изключителни свойства на базата на синергичния ефект на златото и глината в композитите.

Нанопълнителят е разпръснат в матрица от епоксидна смола, създавайки по този начин полимер съдържащ три вещества. Анализираните тестови образци са част от такъв епоксидно-глинен композит. Структурата му е получена посредством компютърна томография с намиращия се в ИИКТ индустриален рентгенов томограф. Размерът на воксела е 2.56μm. Процентното съдържание на глината е приблизително 1.44% от обема.

Използваните материали в настоящото изследване са: епоксидна смола (Epilox T 19-38/500, Leuna-Harze GmbH (Germany); втвърдител (Epilox H 10-30, Leuna-Harze GmbH (Germany)); органично модифицирана глина Cloisite 30B (Southern Clay Products, Inc.); Tetrachloroaurate trihydrate (HAuCl4·3H2O), Sigma-Aldrich – прекурсор за синтеза на златните наночастици.

Изследвали сме четири образеца с размери 128 × 128 × 128 воксела, Виж. Фигура 4.9. Свойствата на материалите са за епоксидната смола $E_e = 2.55GPa$, $\nu_e = 0.35$, а за глината $E_c = 3MPa$, $\nu_c = 0.2$.

Като приложим процедурата за хомогенизация от раздел 4.2.1, получаваме хомогенизираните тензори на коравина за всеки един образец. Означаваме ги със S_i^H, където с индекс i е означен номера на образеца. След това трансформираме получените тензори в координатните системи по направление на главните им направления на анизотропия получавайки тензорите на коравина



(1) (2)

(3) (4)

Фигура 4.9: Разглеждани образци

$\bar{S}_i^H$. Следват получените матрици $\bar{S}_i^H$. Всички стойности са в паскали.

$$\bar{S}_1^H = \begin{bmatrix} 3.86{\times}10^9 & 2.03{\times}10^9 & 2.02{\times}10^9 & -1.43{\times}10^5 & 1.59{\times}10^5 & 1.25{\times}10^6 \\ 2.03{\times}10^9 & 3.85{\times}10^9 & 2.02{\times}10^9 & 4.08{\times}10^5 & -1.20{\times}10^5 & -1.19{\times}10^6 \\ 2.02{\times}10^9 & 2.02{\times}10^9 & 3.82{\times}10^9 & -2.64{\times}10^5 & -3.89{\times}10^4 & -6.19{\times}10^4 \\ -1.43{\times}10^5 & 4.08{\times}10^5 & -2.64{\times}10^5 & 9.06{\times}10^8 & -1.36{\times}10^5 & -1.19{\times}10^5 \\ 1.59{\times}10^5 & -1.20{\times}10^5 & -3.89{\times}10^4 & -1.36{\times}10^5 & 9.07{\times}10^8 & -1.91{\times}10^5 \\ 1.25{\times}10^6 & -1.19{\times}10^6 & -6.19{\times}10^4 & -1.19{\times}10^5 & -1.91{\times}10^5 & 9.11{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_2^H = \begin{bmatrix} 3.88{\times}10^9 & 2.05{\times}10^9 & 2.04{\times}10^9 & 8.00{\times}10^4 & -2.50{\times}10^5 & 2.19{\times}10^4 \\ 2.05{\times}10^9 & 3.87{\times}10^9 & 2.03{\times}10^9 & -3.73{\times}10^5 & -2.61{\times}10^5 & 4.34{\times}10^4 \\ 2.04{\times}10^9 & 2.03{\times}10^9 & 3.84{\times}10^9 & 2.93{\times}10^5 & 5.11{\times}10^5 & -6.54{\times}10^4 \\ 8.00{\times}10^4 & -3.73{\times}10^5 & 2.93{\times}10^5 & 9.09{\times}10^8 & -1.80{\times}10^5 & -2.70{\times}10^5 \\ -2.50{\times}10^5 & -2.61{\times}10^5 & 5.11{\times}10^5 & -1.80{\times}10^5 & 9.10{\times}10^8 & 2.61{\times}10^4 \\ 2.19{\times}10^4 & 4.34{\times}10^4 & -6.54{\times}10^4 & -2.70{\times}10^5 & 2.61{\times}10^4 & 9.14{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_3^H = \begin{bmatrix} 3.89{\times}10^9 & 2.05{\times}10^9 & 2.04{\times}10^9 & -1.38{\times}10^5 & 1.05{\times}10^5 & -3.49{\times}10^5 \\ 2.05{\times}10^9 & 3.88{\times}10^9 & 2.04{\times}10^9 & -5.77{\times}10^5 & 3.07{\times}10^4 & 2.95{\times}10^5 \\ 2.04{\times}10^9 & 2.04{\times}10^9 & 3.85{\times}10^9 & 7.16{\times}10^5 & -1.35{\times}10^5 & 5.36{\times}10^4 \\ -1.38{\times}10^5 & -5.77{\times}10^5 & 7.16{\times}10^5 & 9.11{\times}10^8 & -7.24{\times}10^4 & 8.31{\times}10^4 \\ 1.05{\times}10^5 & 3.07{\times}10^4 & -1.35{\times}10^5 & -7.24{\times}10^4 & 9.11{\times}10^8 & -1.26{\times}10^5 \\ -3.49{\times}10^5 & 2.95{\times}10^5 & 5.36{\times}10^4 & 8.31{\times}10^4 & -1.26{\times}10^5 & 9.16{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_4^H = \begin{bmatrix} 3.94{\times}10^9 & 2.09{\times}10^9 & 2.08{\times}10^9 & 6.66{\times}10^3 & -1.52{\times}10^5 & -5.79{\times}10^5 \\ 2.09{\times}10^9 & 3.93{\times}10^9 & 2.08{\times}10^9 & 2.12{\times}10^5 & 3.17{\times}10^4 & 6.16{\times}10^5 \\ 2.08{\times}10^9 & 2.08{\times}10^9 & 3.91{\times}10^9 & -2.18{\times}10^5 & 1.20{\times}10^5 & -3.67{\times}10^4 \\ 6.66{\times}10^3 & 2.12{\times}10^5 & -2.18{\times}10^5 & 9.18{\times}10^8 & -4.12{\times}10^4 & 5.37{\times}10^4 \\ -1.52{\times}10^5 & 3.17{\times}10^4 & 1.20{\times}10^5 & -4.12{\times}10^4 & 9.19{\times}10^8 & 9.60{\times}10^3 \\ -5.79{\times}10^5 & 6.16{\times}10^5 & -3.67{\times}10^4 & 5.37{\times}10^4 & 9.60{\times}10^3 & 9.22{\times}10^8 \end{bmatrix}.$$

Интересен факт е че и за четирите образеца, трансформационната матрица представляваше ротация около оста z. При това, при всичките образци реакцията на материала по оста z е малко по-слаба. На този етап не можем да бъдем сигурни дали това се дължи на някакъв артефакт от сканирането или са такива свойствата на материала.

Сравнение с пакета GeoDict

Използвахме за сравнение и специализираният комерсиален софтуер за числена хомогенизация GeoDict [30], в частност модула му ElastoDict, за сравнение с нашите резултати. Получените матрици на коравина $\bar{S}_i^{H_{GD}},\ i = 1, \ldots, 4$ следват. Те също са завъртени по главните направление на анизотропия, но направленията не са сортирани по големина на съответните собствени стойности!

$$\bar{S}_1^{H_{GD}} = \begin{bmatrix} 3.86{\times}10^9 & 2.04{\times}10^9 & 2.03{\times}10^9 & -1.35{\times}10^5 & 3.71{\times}10^5 & -5.17{\times}10^5 \\ 2.04{\times}10^9 & 3.87{\times}10^9 & 2.03{\times}10^9 & 1.56{\times}10^5 & -1.93{\times}10^5 & 6.15{\times}10^5 \\ 2.03{\times}10^9 & 2.03{\times}10^9 & 3.83{\times}10^9 & -2.15{\times}10^4 & -1.53{\times}10^5 & -5.58{\times}10^4 \\ -1.35{\times}10^5 & 1.56{\times}10^5 & -2.15{\times}10^4 & 9.08{\times}10^8 & -1.33{\times}10^5 & -2.21{\times}10^5 \\ 3.71{\times}10^5 & -1.93{\times}10^5 & -1.53{\times}10^5 & -1.33{\times}10^5 & 9.07{\times}10^8 & -1.10{\times}10^5 \\ -5.17{\times}10^5 & 6.15{\times}10^5 & -5.58{\times}10^4 & -2.21{\times}10^5 & -1.10{\times}10^5 & 9.11{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_2^{H_{GD}} = \begin{bmatrix} 3.88{\times}10^9 & 2.06{\times}10^9 & 2.04{\times}10^9 & -2.41{\times}10^4 & 2.03{\times}10^5 & -2.53{\times}10^4 \\ 2.06{\times}10^9 & 3.89{\times}10^9 & 2.05{\times}10^9 & 8.95{\times}10^3 & -7.79{\times}10^4 & -9.29{\times}10^4 \\ 2.04{\times}10^9 & 2.05{\times}10^9 & 3.85{\times}10^9 & 2.72{\times}10^5 & -1.41{\times}10^5 & 5.85{\times}10^4 \\ -2.41{\times}10^4 & 8.95{\times}10^3 & 2.72{\times}10^5 & 9.11{\times}10^8 & 1.71{\times}10^5 & -3.00{\times}10^4 \\ 2.03{\times}10^5 & -7.79{\times}10^4 & -1.41{\times}10^5 & 1.71{\times}10^5 & 9.10{\times}10^8 & -2.70{\times}10^4 \\ -2.53{\times}10^4 & -9.29{\times}10^4 & 5.85{\times}10^4 & -3.00{\times}10^4 & -2.70{\times}10^4 & 9.15{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_3^{H_{GD}} = \begin{bmatrix} 3.88{\times}10^9 & 2.06{\times}10^9 & 2.05{\times}10^9 & -5.83{\times}10^4 & -3.61{\times}10^5 & -9.47{\times}10^4 \\ 2.06{\times}10^9 & 3.89{\times}10^9 & 2.05{\times}10^9 & -4.30{\times}10^4 & 4.63{\times}10^4 & 1.23{\times}10^5 \\ 2.05{\times}10^9 & 2.05{\times}10^9 & 3.86{\times}10^9 & 1.27{\times}10^5 & 3.59{\times}10^5 & -9.63{\times}10^4 \\ -5.83{\times}10^4 & -4.30{\times}10^4 & 1.27{\times}10^5 & 9.12{\times}10^8 & 3.97{\times}10^4 & 3.93{\times}10^4 \\ -3.61{\times}10^5 & 4.63{\times}10^4 & 3.59{\times}10^5 & 3.97{\times}10^4 & 9.12{\times}10^8 & -1.11{\times}10^5 \\ -9.47{\times}10^4 & 1.23{\times}10^5 & -9.63{\times}10^4 & 3.93{\times}10^4 & -1.11{\times}10^5 & 9.16{\times}10^8 \end{bmatrix}.$$

$$\bar{S}_4^{H_{GD}} = \begin{bmatrix} 3.94{\times}10^9 & 2.09{\times}10^9 & 2.08{\times}10^9 & -3.71{\times}10^4 & -1.91{\times}10^5 & 1.01{\times}10^5 \\ 2.09{\times}10^9 & 3.94{\times}10^9 & 2.08{\times}10^9 & 3.94{\times}10^5 & 3.40{\times}10^4 & -7.29{\times}10^4 \\ 2.08{\times}10^9 & 2.08{\times}10^9 & 3.91{\times}10^9 & 8.99{\times}10^3 & 1.63{\times}10^5 & -2.72{\times}10^4 \\ -3.71{\times}10^4 & 3.94{\times}10^5 & 8.99{\times}10^3 & 9.20{\times}10^8 & -3.45{\times}10^4 & 1.92{\times}10^4 \\ -1.91{\times}10^5 & 3.40{\times}10^4 & 1.63{\times}10^5 & -3.45{\times}10^4 & 9.19{\times}10^8 & -5.32{\times}10^4 \\ 1.01{\times}10^5 & -7.29{\times}10^4 & -2.72{\times}10^4 & 1.92{\times}10^4 & -5.32{\times}10^4 & 9.23{\times}10^8 \end{bmatrix}.$$

Наблюдава се голямо сходство – разликите са под 1% – между резултатите получени от нашия алгоритъм, и тези – от GeoDict. И при резултатите от GeoDict се наблюдава същата слабост на материала по направление z. Друго важно наблюдение е че на макро ниво свойствата на материалите са почти изотропни. Обикновено такава е и целта, освен в случаи, когато се нуждаем от специална структура. Следователно можем да пресметнем изотропно приближение на модула на еластичност за материала: $E_H \approx 2.46$ GPa. В сравнение с модула на Юнг на чистата смола $E = 2.55$ GPa, наблюдаваме сравнително малко влияние на глинените клъстери.

С настоящата рентгенова техника можем да откриваме глинени клъстери само на микро ниво. А характеристиките на епоксидната смола се влияят и от частиците по-малки от 3 μm.

В друг експеримент [32] бяха проведени тестове посредством наномеханичен тестер [23]. С него бяха извършени 48 повърхностни теста (4 реда по 12 опита на отстояние 80 μm). Всеки тест се прави като се натиска с нано глава и сила от 100 mN. Постоянно се следят приложената сила и преместванията на главата. Модулът на Юнг се пресмята от данните свалени по време на отпускане на главата автоматично от софтуера на апарата, използвайки метода на Оливер-Фар [66]. След усредняване се получава следния модул на Юнг $E_N = 3.37$ GPa.

Експериментите бяха повторени и с изходен модул на Юнг, получен от наномеханичния тестер $E_N = 3.37$GPa. По този начин получихме за хомогенизирания модул на Юнг стойност $E_{H_N} \approx 3.25$. Тези експерименти ясно показват малкото влияние на „големите“ клъстери.

4.2.6 Сравнение с аналитични оценки

Най-накрая искаме да сравним получаваните от нас хомогенизирани тензори на коравина с някои оценки получени по аналитичен път.

Нека композитът е съставен от матрица и пълнител. Матрицата е с параметри модул на Юнг E_m и коефициент на Поасон ν_m, а тези на пълнителя са – E_f и ν_f. Нека също така обемната част съдържаща материала на матрицата към целия обем е означена с V_m, а тази на пълнежа – с V_f, като е в сила равенството $V_m + V_f = 1$.

От литературата [44] са взети следните аналитично изведени оценки, при допускане че $\nu_m = \nu_f$. Като горна граница за получения модул на Юнг се дава така нареченото правило на сместването (на английски – rule of mixtures):

$$E^{max} = V_m E_m + V_f E_f, \tag{4.2.19}$$

а като долна – обратното правило на смесване:

$$E^{min} = \frac{E_m E_f}{V_m E_f + V_f E_m}. \tag{4.2.20}$$

Когато пълнителят е съставен от нишки, ориентирани в една и съща посока за модула на Юнг E_1 по направлението надлъжно на нишките имаме достигане на максимума $E_1 = E^{max}$. За еластичния модул E_2 по другите направления нямаме точна оценка. Съществуват и други по-точни оценки, като тази на Цай [83]:

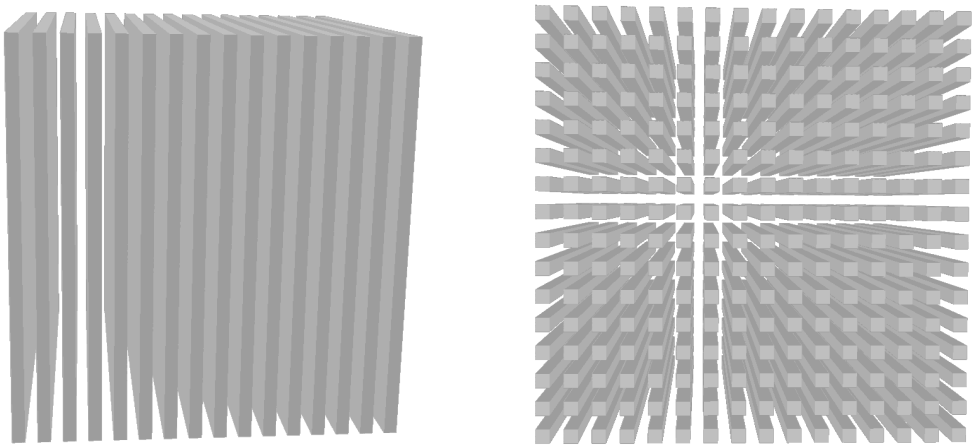
$$\begin{aligned} E^{Tsai} =& 2\left(1 - \nu_f + (\nu_f - \nu_m)\, V_m\right) \left((1 - C) \frac{K_f(2K_m + G_m) - G_m(K_f - K_m)V_m}{(2K_m + G_m) + 2(K_f - K_m)V_m} \right. \\ &\left. + C \frac{K_f(2K_m + G_f) + G_f(K_m - K_f)V_m}{(2K_m + G_f) - 2(K_m - K_f)V_m} \right), \end{aligned} \tag{4.2.21}$$

където

$$\begin{aligned} K_f =& \frac{E_f}{2(1 - \nu_f)} \\ G_f =& \frac{E_f}{2(1 + \nu_f)} \\ K_m =& \frac{E_m}{2(1 - \nu_m)} \\ G_m =& \frac{E_m}{2(1 + \nu_m)}, \end{aligned} \tag{4.2.22}$$

където параметъра C зависи от взаимното разположение на нишките ($C = 0$ означава че всички нишки са изолирани, а $C = 1$ когато всички нишки са в контакт). В общия случай параметърът C не е известен и той трябва да се определя експериментално.

Приложили сме числената хомогенизация за два материала:а) първият е ламинат от 32 пласта редуващи се матрица и пълнител вижте Фигура 4.10,



(а) (б)

Фигура 4.10: (а) Ламинат съставен от 32 пласта, V_f=0.5; (б) Композит съставен от нишки, $V_f = 0.25$

Таблица 4.11: Сравнение с аналитични оценки

Геометрия	V_f	ζ	E^{max}	E_1	E_2	E^{min}	E^{Tsai}
(а)	0.5	10^{-2}	5.05×10^{8}	5.05×10^{8}	2.63×10^{7}	1.98×10^{7}	1.05×10^{8}
(а)	0.5	10^{-1}	5.50×10^{8}	5.50×10^{8}	2.20×10^{8}	1.82×10^{8}	2.51×10^{8}
(а)	0.5	10^{1}	5.50×10^{9}	5.50×10^{9}	2.20×10^{9}	1.82×10^{9}	2.51×10^{9}
(а)	0.5	10^{2}	5.05×10^{10}	5.05×10^{10}	2.63×10^{9}	1.98×10^{9}	1.05×10^{10}
(б)	0.25	10^{-2}	7.53×10^{8}	7.53×10^{8}	5.58×10^{8}	3.88×10^{7}	2.25×10^{8}
(б)	0.25	10^{-1}	7.75×10^{8}	7.75×10^{8}	6.37×10^{8}	3.08×10^{8}	4.31×10^{8}
(б)	0.25	10^{1}	3.25×10^{9}	3.25×10^{9}	1.62×10^{9}	1.29×10^{9}	1.58×10^{9}
(б)	0.25	10^{2}	2.58×10^{10}	2.58×10^{10}	1.85×10^{9}	1.33×10^{9}	4.59×10^{9}

б)във втория пълнителят е разположен в 256 равномерно разположени греди вижте Фигура 4.10(б).

Проведохме числени експерименти с параметри $\nu_f = \nu_m = 0.3$, $E_m = 1\text{GPa}$, $E_f = \zeta E_m$, $\zeta = \{10^{-2}, 10^{-1}, 10, 10^2\}$. В Таблица 4.11 сме събрали резултати от числената хомогенизация, като в нея са показани получените модули на Юнг по направленията надлъжно и напречно на нишките E_1 и E_2, заедно с аналитичните оценки E^{min}, E^{max}, и E^{Tsai} за $C = 0.5$, в зависимост от геометрията (тази от Фигура 4.10(а) или Фигура 4.10(б)) и ζ.

Както виждаме от Таблица 4.11, числено получените оценки за надлъжния модул на Юнг съвпадат с аналитичните такива. Колкото за напречните модули на Юнг E_2 вярване, че получените от нас стойности са по-близки до реалността от грубите оценки E^{min} и E^{Tsai}. Оценките E^{min} никога не се достигат на практика, а за оценките E^{Tsai} параметърът C се установява емпирично в зависимост за различните материали[44].

В заключение ще отбележим, че разгледания алгоритъм за хомогенизация

и софтуер позволяват хомогенизирането на композити съставени от произволен брой материали, а не само от два, както в показаните примери.

Библиография

[1] A. Alexandrescu. *Modern C++ design: generic programming and design patterns applied.* Addison-Wesley Professional, 2001.

[2] P. Arbenz, S. Margenov, and Y. Vutov. Parallel MIC(0) preconditioning of 3D elliptic problems discretized by Rannacher–Turek finite elements. *Computers and Mathematics with Applications*, 55(10):2197–2211, 2008.

[3] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala. A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures. Technical Report 543, Institute of Computational Science, ETH Zurich, 2006.

[4] T. Arbogast and Z. Chen. On the implementation of mixed methods as nonconforming methods for second-order elliptic problems. *Mathematics of Computation*, 64:943–972, 1995.

[5] H. Avron, E. Ng, and S. Toledo. A generalized Courant-Fischer minimax theorem. Technical Report LBNL-6393E, Lawrance Berkley National Laboratory, 2008.

[6] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1996.

[7] O. Axelsson and V. Barker. *Finite Element Solution of Boundary Value Problems: Theory and Computations.* Academic Press, 1983.

[8] O. Axelsson and I. Gustafsson. Iterative methods for the solution of the Naviers equations of elasticity. *Comp. Meth. Appl. Mech. Engin.*, 15:241–258, 1978.

[9] N. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *Zh. Vychisl. Mat. mat. Fiz*, 6:861–885, 1966.

[10] H. Bayraktar. Nonlinear micro finite element analysis of human trabecular bone: a study. ABAQUS Inc.

[11] G. Bencheva and S. Margenov. Parallel incomplete factorization preconditioning of rotated linear FEM systems. *J. Comput. Appl. Mech*, 4(2):105–117, 2003.

[12] G. Bencheva and S. Margenov. Performance analysis of a parallel MIC(0) preconditioning of rotated bilinear nonconforming FEM systems. *Mathematica Balkanica*, 17:319–335, 2003.

[13] A. Bensoussan, J.-L. Lions, and G. Papanicolaou. *Asymptotic analysis for periodic structures*. American Mathematical Soc., 2011.

[14] R. Blaheta. Displacement decomposition – incomplete factorization preconditioning techniques for linear elasticity problems. *Numer. Lin. Alg. Appl*, 1:107–126, 1994.

[15] R. Blaheta, O. Jakl, and J. Starỳ. Iterative solution of singular systems with applications. *Parallel Processing and Applied Mathematics*, 114–123. Springer, 2014.

[16] D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 1997.

[17] A. Brandt. Multi-level adaptive technique/MLAT/ for fast numerical solution to boundary value problems. *Third International Conference on Numerical Methods in Fluid Mechanics, Paris, France*, 82–89, 1973.

[18] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 333–390, 1977.

[19] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, 1986.

[20] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical report, Institute for Computational Studies, POB, 1983.

[21] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer Series In Computational Mathematics. Springer-Verlag, 1991.

[22] W. Briggs and S. McCormick. *A multigrid tutorial*. Society for Industrial Mathematics, 2000.

[23] Bruker Corporation. Nanomechanical testing system. `http://www.bruker.com/products/surface-analysis/tribology-and-mechanical-testing/nanoforce/overview.html`, 2015.

[24] Intel Corporation. Intel® 64 and IA-32 Architectures Optimization Reference Manual. `http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf`, 2015.

[25] S. C. Cowin. Bone poroelasticity. *Journal of biomechanics*, 32(3):217–238, 1999.

[26] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Comput. Math. and Math. Phys*, 1(5):1092–1096, 1961.

[27] Message Passing Interface Forum. MPI: A message-passing interface standard. `http://mpi-forum.org/`, 2003.

[28] Message Passing Interface Forum. MPI-2: Extensions to the message-passing interface.
`http://mpi-forum.org/`, 2012.

[29] G. Beller, M. Burkhart, D. Felsenberg, W. Gowin, H.-C. Hege, B. Koller, S. Prohaska, P. I. Saparin and J. S. Thomsen. Vertebral body data set ESA29-99-L3.
`http://bone3d.zib.de/data/2005/ESA29-99-L3/`, 2009.

[30] Geodict. GeoDict – the Virtual Material Laboratory.
`http://geodict.com`, 2014.

[31] A. Georgiev, A. Baltov, and S. Margenov. Hipergeos benchmark problems related to bridge engineering applications. *REPORT HG CP*, 94–0820, 1994.

[32] I. Georgiev, E. Ivanov, S. Margenov, and Y. Vutov. Numerical homogenization of epoxy-clay composite materials. *Numerical Methods and Applications*, *LNCS* **8962**, 130–137. Springer Berlin Heidelberg, 2015.

[33] C. Geuzaine and J. F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.
`http://www.geuz.org/gmsh/`, 2014.

[34] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

[35] Khronos Group. OpenCL - The open standard for parallel programming of heterogeneous systems.
`http://www.khronos.org/opencl/`, 2014.

[36] I. Gustafsson. *Stability and rate of convergence of modified incomplete Cholesky factorization methods*. Informationsbehandling, särskilt numerisk analys, 1979.

[37] I. Gustafsson. An incomplete factorization preconditioning method based on modification of element matrices. *BIT Numerical Mathematics*, 36(1):86–100, 1996.

[38] I. Gustafsson and G. Lindskog. On parallel solution of linear elasticity problems: Part I: theory. *Numerical linear algebra with applications*, 5(2):123–139, 1998.

[39] W. Hackbusch. *Multigrid methods and Applications*. Springer Verlag, 1985.

[40] W. Hackbusch and U. Trottenberg. *Multigrid methods*. Springer Verlag, 1982.

[41] V. Henson and U. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 155–177, 2000.

[42] R. Hoppe and S. Petrova. Optimal shape design in biomimetics based on homogenization and adaptivity. *Mathematics and Computers in Simulation*, 65(3):257–272, 2004.

[43] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press New York, 1987.

[44] R. Jones. *Mechanics of composite materials*. McGraw-Hill New York, 1975.

[45] W. Kahan. Pracniques: Further remarks on reducing truncation errors. *Commun. ACM*, 8(1):40, January 1965.

[46] D. Kincaid and W. Cheney. *Numerical analysis: mathematics of scientific computing*. Brooks/Cole, 1991.

[47] N. Kosturski. MIC(0) DD Preconditioning of FEM Elasticity Systems on Unstructured Tetrahedral Grids. *Large Scale Scientific Computations 2007*, *LNCS* **4818**, 688–695. Springer, 2008.

[48] N. Kosturski and S. Margenov. Comparative Analysis of Mesh Generators and MIC(0) Preconditioning of FEM Elasticity Systems. *Numerical Methods and Applications 2006*, *LNCS* **4310**, 74–81. Springer, 2007.

[49] R. Lazarov and S. Margenov. On a Two-Level Parallel MIC(0) Preconditioning of Crouzeix-Raviart Non-conforming FEM Systems. *Numerical Methods and Applications*, *LNCS* **2542**, 192–201. Springer Berlin Heidelberg, 2003.

[50] R. Lipton, D. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal on Numerical Analysis*, 16(2):346–358, 1979.

[51] I. Lirkov. MPI solver for 3D elasticity problems. *Mathematics and computers in simulation*, 61(3):509–516, 2003.

[52] I. Lirkov and S. Margenov. MPI parallel implementation of CBF preconditioning for 3D elasticity problems. *Mathematics and computers in simulation*, 50(1):247–254, 1999.

[53] I. Lirkov and Y. Vutov. Comparative analysis of high performance solvers for 3D elliptic problems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 483–492, 2007.

[54] I. Lirkov, Y. Vutov, M. Ganzha, and M. Paprzycki. Comparative Analysis of High Performance Solvers for 3D Elasticity Problems. *Numerical Methods and Applications*, *LNCS* **5434**, 392–399. Springer-Verlag, 2009.

[55] I. Lirkov, Y. Vutov, M. Paprzycki, and M. Ganzha. Benchmarking Performance Analysis of Parallel Solver for 3D Elasticity Problems. *Large-Scale Scientific Computing*, *LNCS* **4818**, 705–7012. Springer-Verlag, 2008.

[56] I. Lirkov, Y. Vutov, M. Paprzycki, and M. Ganzha. Parallel performance evaluation of MIC(0) preconditioning algorithm for voxel μFE simulation. *Parallel Processing and Applied Mathematics*, *LNCS* **6068**, 135–144. Springer, 2010.

[57] D. S. Malkus and T. J. R. Hughes. Mixed finite element methods – reduced and selective integration techniques: a unification of concepts. *Computer Methods in Applied Mechanics and Engineering*, 63–81, 1990.

[58] S. Margenov. Displacement decomposition—MIC(0) preconditioning of linear elasticity nonconforming FEM problems. *16th IMACS World Congress 2000, Lausanne, Proceedings, 107-4*, 2000.

[59] S. Margenov and N. Kosturski. MIC(0) preconditioning of 3D FEM problems on unstructured grids: Conforming and non-conforming elements. *Journal of Computational and Applied Mathematics*, 226(2):288–297, 2009.

[60] S. Margenov, S. Stoykov, and Y. Vutov. Numerical homogenization of heterogeneous anisotropic linear elastic materials. *Large-Scale Scientific Computing*, LNCS, 347–354. Springer Berlin Heidelberg, 2014.

[61] S. Margenov and Y. Vutov. Parallel PCG algorithms for voxel FEM elasticity systems. *Proceedings of the International Multiconference on Computer Science and Information Technology*, 517–526, 2007.

[62] S. Margenov and Y. Vutov. Preconditioning of voxel FEM elliptic systems. *TASK Quarterly*, 11(1-2):117–128, 2007.

[63] S. Margenov and Y. Vutov. Parallel MIC(0) preconditioning for numerical upscaling of anisotropic linear elastic materials. *Large-Scale Scientific Computing*, *LNCS* **5910**, 805–812. Springer, 2010.

[64] J. E. Marsden and T. J. R. Hughes. *Mathematical foundations of elasticity.* Dover Publications, 1994.

[65] NVIDIA. CUDA Toolkit Documentation. `http://docs.nvidia.com/cuda/index.html`, 2014.

[66] W. Oliver and G. Pharr. Measurement of hardness and elastic modulus by instrumented indentation: Advances in understanding and refinements to methodology. *Journal of materials research*, 19(01):3–20, 2004.

[67] I. Petrova, E. Ivanov, R. Kotsilkova, Y. Tsekov, and V. Angelov. Applied study on mechanics of nanocomposites with carbon nanofillers. *Journal of Theoretical and Applied Mechanics*, 43(3):67–76, 2013.

[68] P. Pissis and R. Kotsilkova. *Thermoset nanocomposites for engineering applications.* iSmithers Rapra Publishing, 2007.

[69] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations.* Springer, 1994.

[70] O. Rand and V. Rovenski. *Analytical methods in anisotropic elasticity: with symbolic computational tools*. Springer Science & Business Media, 2007.

[71] R. Rannacher, S. Turek, S.S.M. Modelle, and D. Forschungsgemeinschaft. Simple nonconforming quadrilateral Stokes element. *Numerical Methods for Partial Differential Equations*, 8(2):97–112, 1992.

[72] D. Richfield. Stress vs. Strain curve for structural steel. `http://en.wikipedia.org/wiki/File:Stress_v_strain_A36_2.svg`, 2014.

[73] J. W. Ruge and K. Stuben. Algebraic multigrid. *Multigrid methods*, 3:73–130, 1987.

[74] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial Mathematics, 2003.

[75] M. Sadd. *Elasticity: theory, applications, and numerics*. Academic Press, 2009.

[76] P. Saint-Georges, G. Warzee, R. Beauwens, and Y. Notay. High-performance PCG solvers for FEM structural analysis. *International journal for numerical methods in engineering*, 39(8):1313–1340, 1996.

[77] J. Schöberl. NETGEN – automatic mesh generator. `http://www.hpfem.jku.at/netgen/`, 2014.

[78] J. Shewchuk. What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures. Technical report, In Proc. of the 11th International Meshing Roundtable, 2002.

[79] J. Shewchuk. Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. `http://www.cs.cmu.edu/~quake/triangle.html`, 2014.

[80] H. Si. TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. `http://wias-berlin.de/software/tetgen/`, 2015.

[81] W. S. Slaughter and J. Petrolito. Linearized theory of elasticity. *Applied Mechanics Reviews*, 55:90, 2002.

[82] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The Complete Reference*. The MIT Press, 1998.

[83] S. W. Tsai. Structural behavior of composite materials. Technical report, DTIC Document, 1964.

[84] Y. Vutov. Parallel incomplete factorization of 3D NC FEM elliptic systems. *Numerical Methods and Applications*, *LNCS* **4310**, 114–121. Springer-Verlag, 2007.

[85] Y. Vutov. Parallel DD-MIC(0) Preconditioning of Nonconforming Rotated Trilinear FEM Elasticity Systems. *Large-Scale Scientific Computing*, *LNCS* **4818**, 745–572. Springer-Verlag, 2008.

[86] J. Wolff. *The law of bone remodelling.* Springer, 1986.

[87] U. Yang. Parallel algebraic multigrid methods-high performance preconditioners. *Lecture Notes in Computational Science and Engineering*, 51:209, 2005.

[88] Б. Боянов. *Лекции по числени методи.* „Дарба“, София, 1995.

[89] Я. Вутов. Паралелни Итерационни Методи за Неконформни Крайни Елементи.
`http://parallel.bas.bg/~yavor/disert/`, 2015.

[90] С. Маргенов. *Числени методи за системи с разредени матрици.* Институт по Паралелна Обработка на Информацията – Българска Академия на Науките, София, 2007.

[91] К. Марков. *Математическо Моделиране.* Университетско издателство „Св. Климент Охридски“, София, 2002.

[92] М. Петков. *Числени методи на линейната алгебра.* „Наука и изкуство“, София, 1974.

[93] Бл. Сендов and В. Попов. *Числени методи – втора част.* Наука и изкуство, София, 1978.

[94] Бл. Сендов and В. Попов. *Числени методи – първа част.* Университетско издателство „Св. Кл. Охридски“, София, 1996.

[95] Г. Стренг and Дж. Фикс. *Теория метода конечных элементов.* „Мир“, Москва, 1977.