ИНСТИТУТ ПО ИНФОРМАЦИОННИ И
КОМУНИКАЦИОННИ ТЕХНОЛОГИИ
БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

БЪЛГАРСКА
АКАДЕМИЯ
на НАУКИТЕ
1869

Димитър Георгиев Славчев

СЪСТАВНИ ЧИСЛЕНИ МЕТОДИ И СКАЛИРУЕМИ БЛОЧНИ АЛГОРИТМИ

ДИСЕРТАЦИЯ

за придобиване на *образователната и научна степен „Доктор“*
по научна специалност: 01.01.09 „Изчислителна математика“
в професионално направление: 4.5 „Математика“

Научен ръководител: чл.-кор. Светозар Маргенов

20 януари 2022 г.

Съдържание

Списък на фигурите

Списък на таблиците

Често използвани означения и съкращения

$\begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$	При запис на матрици нулевите елементи и блокове са пропуснати.
δ	Делтата на Кронекер: $\delta_{ij} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$
$\emptyset$	Празно множество.
Γ	Гамма функцията: $\Gamma(z) = \int_0^\infty x^z e^{-x} dx$.
$\mathbf{u}, \mathbf{f}$	С удебелени малки латински букви са отбелязани вектори.
$\mathbb{R}$	Множеството на реалните числа.
$\mathrm{diag}(m_1, \ldots, m_N)$	Диагонална матрица с изброени елементи по главният диагонал.
A, D	С главни латински букви са отбелязани матрици или блокове от матрици.
gcc	GNU Compiler Collection съкращение и команда за извикване на компилаторите на GNU проекта.
HPC	High Performance Computing, високопроизводителни изчисления.
HSS	Йерархична полусепарабелна компресия, от Hierarchical Semi-Separable compression.

icc	Intel Compiler Collection съкращение и команда за извикване на компилаторите на Интел.
MIC	Many Integrated Core архитектура на Intel за високопроизводителни копроцесори.
P.V.	Главна стойност, от Principle Value.
STRUMPACK	STRUctured Matrix PACKage, паралелен пакет за използване на йерархични методи (в частност HSS) за решаване на системи линейни уравнения.
МГЕ	Метод на граничните елементи.
МКЕ	Метод на крайните елементи.
СЛАУ	Системи линейни алгебрични уравнения.

Увод

Актуалност на темата

Компютърното моделиране е в основата на много от значимите постижения в съвременните наука и технологии. Редица задачи от научните и приложните области на човешкото знание се нуждаят от ефективни методи за изучаване на природни процеси и феномени. Такива задачи често се описват с диференциални и интегрални уравнения, които не винаги могат да бъдат решени аналитично. Компютърните модели позволяват изучаване на характеристиките на нови материали и технологии, както и изследването на процеси, при които наблюденията и измерванията са трудно постижими. Това води и до намаляване на разходите за скъпо струващи лабораторни и реални експерименти.

Численото решаване на задачи с голяма размерност изисква използването на високопроизводителни изчислителни компютърни системи, както и специализиран хардуер и софтуер – графични карти, ускорители, висикоскоростна комуникация между сървърите на системата, софтуерни стандарти и пакети за комуникация между процесорни ядра и сървъри, софтуерни пакети имплементиращи ефективни числени методи и много други.

Съществуват различни методи за дискретизация на диференциални уравнения. Такива например са мрежовите методи като методът на крайните елементи, методът на граничните елементи и методът на крайните разлики. След тяхното прилагане, диференциалните уравнения се свеждат до системи от линейни алгебрични уравнения. Методът на Гаус е универсален подход за решаване на такива системи. В общия случай той има висока изчислителна сложност – $O(n^3)$, където n е броят на неизвестните [90]. При използване на метода на крайните елементи или метода на крайните разлики за задачи с локален характер например обикновена дифузия или топлопроводимост, получената система има разредена матрица. Това означава, че броят на ненулевите коефициентите във всеки ред или стълб е $O(1)$, т.е. е равномерно ограничен. При задачи с голяма размерност, когато $n >> 1$, това свойство е много важ-

но. Разработени са редица ефективни специализирани числени методи, които използват структурата на разредените матрици, виж например [90, 89].

При дискретизация на диференциалното уравнение с метода на граничните елементи, както и при прилагане на метода на крайните елементи за нелокални задачи (например изследваната в тази дисертация *аномална* (дробна) дифузия) [4], получената матрица на коравина е *плътна*. Един възможен подход за намаляване на изчислителната сложност на решението на системи с такива матрици е йерархичната компресия въведена от Хакбуш (Hackbusch) [39]. При нея се използва *структурата* на изходната матрица. Целта е както за да се редуцира заеманата памет, така и да се подобри изчислителната ефективност. Тук под *структура* на плътна матрица се разбира наличието на апроксимация на изходящата матрица с нисък ранг в извъндиагоналните блокове. Това свойство позволява представянето на извъндиагоналните блокове като произведение на по-малки матрици. Съществуват различни разновидности на йерархични матрици, в т.ч. $\mathcal{H}$, $\mathcal{H}^2$, или йерархични полусепарабелни (Hierarchically Semi-Separable) HSS матрици.

Изследванията в настоящата дисертация са посветени на използването на йерархична полусепарабелна компресия и нейното приложение за решаване на системи със *структурирани* плътни матрици получени при дискретизации на диференциални уравнения с метод на граничните елементи (за ламинарен поток около крилни профили) и метод на крайните елементи (за *дробен* лапласиан, моделиращ *аномална* дифузия). Йерархичните матрици са първоначално въведени за решаване на системи получени при дискретизация на класове задачи с метод на граничните елементи. Това е мотивация за изследванията в Глава 2, където са анализирани възможностите за прилагане на HSS компресия за задачата за обтичане на крилни профили. Разглежданият в дисертацията *дробен* лапласиан се дефинира в интегрален вид с помощта на потенциал на Риц. При дискретизация на съответните нелокални гранични задачи с метод на крайните елементи се получава матрица на коравина, за която могат да се търсят аналогии с матрици получавани при прилагане на метода на граничните елементи. В Глави 3 и 4 са анализирани методи и алгоритми за подобряване на изчислителната ефективност при прилагане на HSS компресия за задачи с *дробна* дифузия.

Обзор на основни резултати в областта

Законът на Мур [59] от далечната 1965-та година обобщава наблюдаваното експоненциално увеличаване на броя на транзисторите в чиповете на компютърните процесори и съответния експоненциален ръст на тяхната производителност. Тази тенденция продължава и до днес. Съществуват обаче фун-

даментални ограничения за максималната честота, на която един процесор може да работи. В тези условия все по-голямо значение има максималната реализация на идеята за паралелизъм на различни нива на архитектурата на изчислителните системи. Така например със средствата на паралелното програмиране работата се разделя между отделните процесорни ядра и сървъри (виж например [57]). Огромният напредък във възможностите на съвременните високопроизводителни изчислителни системи засилва още повече ролята на ефективните числени методи и паралелните алгоритми. Суперкомпютърните симулации са определящи за развитието в редица високотехнологични области. Такива например са *in silico* молекулярната биология и проектиране на лекарствени средства [60, 53], анализа на турбулентни течения [55], безразрушителният контрол [23], обработката на тримерни изображения [52], динамика на флуидите [29] и много други.

След подходяща дискретизация математическите модели обикновено се свеждат до задачи на линейната алгебра, измежду които определяща е ролята на решаването на системи от линейни алгебрични уравнения. За целта се разработват специализирани софтуерни средства. Linear Algebra PACKage [7] (LAPACK) се използва като утвърден стандарт за разработване на високопроизводителни софтуерни пакети за реализиране на основните задачи на линейната алгебра. Оптимизирани пакети базирани на LAPACK стандарта са реализирани от водещи производители на хардуер. Такива примери са Math Kernel Library [44] на Intel, ACML [5] на AMD, както и аналогични продукти на Apple, IBM, Cray и др. Ще отбележим също така софтуерния пакет със свободен достъп Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA) [26].

Заедно с производителността на компютърните системи все по-важен става и въпросът за тяхната енергийна ефективност. При тежки изчислителни задачи се използва специализиран хардуер като графични карти и ускорители. Архитектурата Many Integrated Core (MIC) на Intel е ускорител, който се конкурира с NVIDIA като съотношение на цена и производителност. MIC архитектурата е по-лесна за работа от графичните карти, но също се нуждае от внимателен избор на настройки и значителни промени в съществуващи програми за постигане на висока производителност. Такъв тип резултати са представени например в статиите [24, 45, 77, 48, 51]. Важно е също така, че някои от популярните софтуерни пакети за линейна алгебра също имат имплементация за MIC архитектурата. Такъв пример е Matrix Algebra on GPU and Multicore Architectures (MAGMA) [13].

При решаване на системи линейни уравнения с разредени матрици се прилагат специализирани методи. Те използват структурата на ненулевите елементи, както и специфични свойства свързани със задачите при дискрети-

зацията, на които са получени съответните системи. Така например целта на итерационните методи с преобуславяне е същественото намаляване на броя на итерациите и оттам на общата изчислителна сложност. За широк клас задачи със симетрични и положително определени разредени матрици са разработени многомрежови и многонивови методи, които имат оптимална изчислителна сложност $O(n)$. Обзор на числени методи за системи с разредени матрици може да се намери в [90].

В общия случай за решаване на системи линейни алгебрични уравнения с плътни матрици се прилагат варианти на метода на Гаус, които използват последователно изключване на неизвестните. По предположение плътната матрица е хомогенна, тъй като не се предполага, че тя има нулеви елементи. Методът на Гаус има изчислителна сложност $O(n^3)$. В настоящата дисертация е изследван алтернативен подход на базата на йерархична компресия. Целта е намаляване на изчислителната сложност. Тук под *структура* на плътна матрица се разбира наличието на извъндиагонални блокове с нисък ранг. Поточно предполага се че такова свойство е в сила за матрица, която апроксимира изходната. Наличието на подходяща *структура* е в основата на йерархичната компресия и съответните йерархични методи за решаване на системи с плътни матрици за класове от задачи на изчислителната математика. Йерархичната компресия е въведена от Хакбуш в [39], където са изследвани т.н $\mathcal{H}$-матрици. Други разновидности на йерархична компресия са $\mathcal{H}^2$-матриците [40] и йерархичните полусепарабелни матрици (HSS) [56]. Теоретична обосновка на методите използващи HSS компресия може да се намери в [87].

Софтуерният пакет STRUctured Matrices PACKage (STRUMPACK) предлага паралелна имплементация на солвър базиран на HSS компресия и ULV-подобна факторизация [66]. В същата статия са представени примери на класове матрици, за които алгоритъмът е ефективен. Оценката на изчислителната сложност е както следва: $O(n^2r)$ аритметични операции за HSS компресията; $O(nr^2)$ – за ULV-подобната факторизация; и $O(nr)$ – за решаването на системата с факторизираната матрица. Тук с r е означен най-високият ранг на извъндиагоналните блокове изчислен в процеса на HSS компресия. Показано е, че за определени класове задачи рангът r е много по малък от броя на неизвестни n. Така например за двумерни задачи на Поасон дискретизирани с метод на крайните елементи или метод на крайните разлики r е константа. В този случай матрицата е разредена и HSS компресията е точна. Показано е също така, че за тримерни задачи на Хелмхолц, дискретизирани с метод на граничните елементи, r расте бавно с нарастването на n [85].

Изчислителната сложност на йерархичните методи зависи от възможността изходната матрица да бъде апроксимирана с матрица, която има извъндиагонални блокове с нисък ранг. В този смисъл ефективността на HSS компреси-

ята се определя от това дали матрицата има подходяща *структура*. Известно е също така, че *структурата* на матрицата зависи съществено от подреждането на неизвестните. Има задачи и мрежови методи, за които добрата номерация на възлите (и от там на неизвестните) е интуитивно ясна. В общия случай обаче се налага да се използват специално разработени методи за пренареждане. Така например при компресиране на разредени матрици се прилага метода на вложените сечения [20]. В този контекст е и работа [64], където се анализира използването на k-средни (k-means) за плътни матрици, които са породени от хребетообразна регресия с ядро (Kernel Ridge Regression).

Съществена част от дисертацията е посветена на числено решаване на дробно дифузионни задачи. Дробната дифузия (наричана още аномална дифузия) описва нелокални процеси, които се наблюдават в различни физични и социални среди. За разлика от обикновената (локална) дифузия аномалната дифузия включва т.н. бързи преходи или тунелни ефекти. В литературата са публикувани разнообразни примери на математически модели на процеси и явления, които се описват с дробна дифузия. Такива например са: течения в силно нееднородни порести среди, суперпроводимост, дифузия на полимери в суперстудени среди [10]; електродифузия на йони в нервни клетки [49] и диагностика с помощта на фотонна дифузия [78]; обработка на изображения и машинно самообучение [65]; разпространение на вирусни заболявания, компютърни вируси, както и на престъпност [18]. *Дробният* оператор на Лаплас описва аномална дифузия по пространството. Съществуват различни дефиниции на *дробен* лапласиан. Важно е да отбележим, че те не са еквивалентни. Така например в [43] е анализирана разликата между интегралната и спектралната дефиниции (виж също статии [61] и [42] и литературата в тях).

Цели и задачи на дисертацията

Основни цели и задачи на дисертацията са:

- Сравнителен анализ на бързодействието на блочни методи и алгоритми за решаване на системи линейни алгебрични уравнения с плътни матрици.

- Сравнителен анализ на бързодействието и паралелната ефективност на софтуерни пакети прилагащи блочна LU факторизация за решаване на системи линейни алгебрични уравнения с плътни матрици.

- Анализ на бързодействието, паралелната ефективност и точност на метод за апроксимация на решението на системи линейни алгебрични уравнения базиран на йерархична полусепарабелна компресия (HSS) от соф-

туерния пакет STRUMPACK за системи с подходяща структура на матрицата.

- Разработване на алгоритми за пренареждане на неизвестните при задачи породени от дискретизация с метод на крайните елементи на дробна дифузия с цел подобряване на ефективността на йерархичната полусепарабелна компресия на така изчислената матрица на коравина.

- Числено решаване на елиптични и параболични задачи от областта на аномалната дифузия описана с интегралната формулировка на дробен лапласиан и дискретизирана по пространството с метод на крайните елементи.

Методология на изследването

В дисертацията се анализира ефективността, в смисъл на бързодействие, паралелно ускорение и точност (в случая на апроксимация с HSS компресия), на блочни методи за решаване на плътни системи линейни алгебрични уравнения. За целта се използват софтуерни пакети, в които са реализирани изследваните блочни методи.

При задачата разгледана в Глава 2 се използва паралелната програма разработена в [69] за дискретизация по метода на граничните елементи на интегралното уравнение и генериране на съответната система линейни алгебрични уравнения. При задачите за дробна дифузия, разгледана в Глави 3 и 4, се използва MatLab програмата публикувана в [4] от Акоста и др. за дискретизация по метода на крайните елементи и генериране на съответната система линейни алгебрични уравнения. Разработени са програми на MatLab, които реализират предложените алгоритми за пренареждане на неизвестните (Приложение А), както и за пресмятане на матрица на масата (Приложение Б). Софтуерните библиотеки със свободен достъп са компилирани върху използваните компютърни системи.

Провеждат се последователни и паралелни експерименти, като се варира броя на използваните нишки и процеси (вторите при системи с разпределена памет). Времето за решаване на задачите се измерва с помощта на системната функция `gettimeofday`. Анализират се получените времеви резултати и се сравнява бързодействието и паралелното ускорение на изследваните методи. При приблизителния йерархичен метод базиран на HSS компресия се анализира и относителната грешка на решението. В този случай за референтно се приема решението полученото с пряк гаусов солвър.

Спецификация на използваните високопроизводителни изчислителни системи

За числените експерименти описани в Глави 2 и 3 са използвани две високопроизводителни изчислителни системи на Института по информационни и комуникационни технологии към Българска академия на науките. Това са суперкомпютър AVITOHOL [1] и Сървърната системата на Центъра за върхови постижения по "Информатика и информационни и комуникационни технологии (ЦВП по Информатика и ИКТ) [2] (в текста се използва името nv001, от името на входния сървър). [1] По-голяма част от представените експерименти са проведени на AVITOHOL, докато сървърната система е използвана в Глава 2 за анализ на паралелната ефективност в случая на разпределена памет.

AVITOHOL е изграден от 150 HP Cluster Platform SL250S GEN8 сървъри с по 2 Intel Xeon E 2650 v2 CPUs и 2 Intel Xeon Phi 7120P копроцесора (Many Integrated Core (MIC) архитектура).

Сървърната система е част от Лабораториите за разработка на приложения и обработка на входно изходни данни – инфраструктура на ЦВП по Информатика и ИКТ. Те се състоят от 40 сървъра Fujitsu Primergy RX 2540 M4 с графична карта NVIDIA Tesla V100 32GB, 128 GB RAM и CPU 2x Intel Xeon Gold 5118 2.30GHz 24 core. Лабораторията в Института по информационни и комуникационни технологии включва 12 сървъра. За по-кратко ще наричаме тази сървърна система nv001 (от името на входния сървър).

Структура и съдържание на дисертацията

Увод

Уводът включва мотивацията за настоящата работа и встъпително описание на използваните методи и решаваните задачи. Направен е кратък литературен обзор на резултати в предметната област. Представена е методологията на изследването и накрая е дадено сбито описание на структурата и съдържанието на дисертацията.

[1] Авторът изказва благодарност за предоставения достъп до електронната инфраструктурата на Центъра за върхови постижения по Информатика и информационни и комуникационни технологии, с финансовата подкрепа на Договор BG05M2OP001-1.001-0003 по Оперативна програма „Наука и образование за интелигентен растеж“ (2014-2020), съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Глава 1: Методи за решаване на системи линейни уравнения с плътни матрици

Тази глава има въвеждащ характер. Дадено е описание на анализираните в дисертацията блочни методи за решаване на системи линейни алгебрични уравнения с плътни матрици, както и базови оценки на тяхната изчислителна сложност.

В Раздел 1.1 са описани преки методи за решаване на системи линейни алгебрични уравнения с плътни матрици. Накратко е разгледан универсалния пряк метод на гаусова елиминация и базираната на него LU факторизация. Методът на Гаус решава задачата на две стъпки – прав ход, при който задачата се преобразува в еквивалентна система с горна триъгълна матрица и обратен ход, в който се получава решението.

Раздел 1.2 е посветен на йерархични методи разработени за решаване на системи със *структурирани* плътни или разредени матрици. Този тип методи са първоначално въведени за работа с плътни матрици получени при прилагане на метода на граничните елементи. Специално внимание е отделено на йерархичната полусепарабелна (HSS) компресия на плътни матрици и нейната имплементация в софтуерния пакет със свободен достъп STRUMPACK. Описани са ULV-подобната факторизация на компресираната матрица и решаването на системата с така факторизираната матрица. Разгледани са предимствата на базирания на HSS компресия метод водещи до по-малка изчислителна сложност за задачи с подходяща *структура* на изходната матрица.

Глава 2: Метод на граничните елементи за числено решаване на двумерна задача за обтичане на крилни профили

В тази глава са представени числени резултати за обтичането на крилни профили на Жуковски. Получената система линейни алгебрични уравнения с *плътна* матрица се използва за бенчмарк при сравнителния анализ на разгледаните блочни алгоритми.

В Раздел 2.1 е описана постановката на математическия модел. Външната гранична задача за уравнението на Лаплас е представена в интегрален вид. За дискретизация на полученото интегрално уравнение е приложен метод на граничните елементи с колокация.

В Раздел 2.2 и Раздел 2.3 са представени числени експерименти получени съответно върху компютърни системи с обща и разпределена памет. Направен е сравнителен анализ на бързодействието и паралелната ефективност на разгледаните високопроизводителни софтуерни пакети за решаване на системи линейни алгебрични уравнения посредством блочна LU факторизация при използване на процесори Intel Xeon и ускорители Intel Xeon Phi. Изслед-

вана е също така ефективността върху компютърни системи с обща памет на метод базиран на йерархична полусепарабелна компресия. Резултатите са сравнени с тези на най-ефективния софтуерен пакет използващ блочна LU факторизация. Анализирана е и относителната грешка на решенията получени с *приблизителния* метод използващ HSS компресия.

Глава 3: Метод на крайните елементи за числено решаване на двумерна стационарна задача за дробна дифузия

Предмет на изследване в тази глава е двумерна гранична задача за *аномална* дифузия.

Постановката на задачата е представена в Раздел 3.1. *Аномалната* дифузия се описва с *дробен* оператор на Лаплас дефиниран в интегрален вид с помощта на потенциал на Риц. За дискретизация се прилага метод на крайните елементи. Дробният лапласиан е нелокален оператор, в резултат на което матрицата на коравина на системата от линейни алгебрични уравнения е плътна.

Бързодействието на йерархичния солвър зависи силно от свойствата на извъндиагоналните блокове на матрицата. В Раздел 3.2 са представени пет метода за пренареждане на неизвестните. Целта е подобряване на *структурата* на матрицата. Това означава, че в процеса на HSS компресията се получават извъндиагонални блокове с нисък ранг. Три от пренареждането имат покоординатна интерпретация: по „Y“ координати – „top“; по хоризонтални линии – „stripes“; и по спирала около центъра – „snake“. Другите два метода са рекурсивни. Те се основават съответно на метода на вложените сечения и на рекурсивна бисекция.

В Раздел 3.3 са разгледани числени експерименти върху компютърни системи с обща памет. Направен е сравнителен анализ на бързодействието и точността на *приблизителното* решение на задачата с йерархичния метод реализиран в пакета STRUMPACK при вариране на предложените пренареждания на неизвестните. Ефективността на йерархичния солвър е анализирана в сравнение с гаусовия солвър от пакета MKL.

Глава 4: Метод на крайните елементи за решаване на двумерна параболична задача за дробна дифузия

В Глава 4 разглеждаме параболична задача с двумерна *аномална* дифузия по пространството.

В Раздел 4.1 е представена постановката на задачата. Използват се понятия и формули от предходната глава. За дискретизация по времето се прилага

неявен метод на Ойлер с постоянна стъпка по времето и диагонална концентрация на матрицата на масата. Така задачата се свежда до решаване на поредица от системи линейни алгебрични уравнения с една и съща матрица на прехода и променящи се десни части на различните стъпки по времето. Това дава възможност факторизацията да се извърши еднократно. Показано е, че при тези предположения йерархичният метод базиран на HSS компресия има предимства в сравнение с блочната LU факторизация.

В Раздел 4.2 е направен сравнителен анализ на директния гаусов солвър реализиран в пакета MKL и метода използващ HSS компресия от пакета STRUMPACK. За втория метод са анализирани времената за изпълнение на отделните части от алгоритъма, както и относителната грешка. Получените резултати потвърждават преимуществата на йерархичния метод при числено решаване на разглежданото дробно дифузионно параболично уравнение.

Заключение

Заключителните бележки обобщават основните резултати, получени в предходните три глави.

Представени са обобщаващи бележки за основните резултати получени в дисертацията. Формулирани са научните и научно-приложните приноси. Даден е списък на публикуваните статии и на изнесените доклади на научни форуми върху които се базира тази работа.

Формулирани са основните научни и научно-приложни приноси.

Благодарности

Изказана е благодарност към хората, без които тази дисертация не би могла да бъде завършена или дори започната.

Приложения

В Приложение А са описани програмите на MatLab, които реализират алгоритмите за пренареждане на неизвестните предложени и използвани в Глави 3 и 4. В Приложение Б е описана програма на MatLab, с която се изчислява матрицата на масата с диагонална концентрация използвана в Глава 4.

Библиография

Библиографията включва 90 заглавия. Това са използвани в дисертацията източници – статии, книги и др., както и статиите, върху които тя е базирана.

Глава 1

Методи за решаване на системи линейни уравнения с плътни матрици

Много задачи от изчислителната практика се решават числено чрез свеждане до система от линейни алгебрични уравнения. Така например при прилагане на *метода на граничните елементи* се получава система с плътна матрица. При числено решаване на класически (локални) диференциални уравнения с помощта на методите на *крайните разлики* и *крайните елементи* непрекъсната задача се свежда до система с разредена матрица. Както ще видим по-късно при дискретизация с крайни елементи на уравнения с *дробна* степен на оператора на Лаплас (*дробна* дифузия) матрицата отново е плътна.

1.1 Преки методи

Съществуват два основни преки метода за решаване на системи от линейни алгебрични уравнения – *методът на Крамър* и *методът на Гаус*. При *метода на Крамър* (1.1) трябва да се пресметне детерминантата на системата (сложността на тази операция е $O(n^3)$), след което за всяко неизвестно да се пресмята детерминантата на адюнгирано количество (сложност $O(n^3)$). В сила е формулата

$$x_i = \frac{\det(A_i)}{\det(A)}, \quad i = [1, n]. \tag{1.1}$$

Така за общата сложност получаваме $O(n^4)$. По тази причина методът на Крамър се използва предимно за теоретични обосновки.

Методът на Гаус е универсален метод за решаване на системи от линейни алгебрични уравнения. Той е основа на повечето преки методи. Така например LU факторизацията (още наречена декомпозиция) се базира на последователно изключване на неизвестните по метода на Гаус. LU факторизацията е базов метод, реализиран в основните високопроизводителни софтуерни библиотеки на изчислителната линейна алгебра [26, 13]. Разработени са варианти на метода на Гаус, в които се отчитат специфични свойства на матрицата от коефициенти на системата. Така например LDL^t факторизация се прилага за решаване на системи със симетрични матрици, докато методът Холецки (Cholesky factorization) се използва за решаване на системи със симетрични положителни определени матрици [90].

1.1.1 Метод на Гаус

Методът на Гаус за решаване на системата от линейни алгебрични уравнения $Ax = b$ включва прав и обратен ход: (i) Чрез еквивалентни преобразувания матрицата на системата се привежда в горна триъгълна (прав ход, елиминация); (ii) Рекурсивно в обратна последователност се изключват извъндиагоналните елементи в i-тия ред на матрицата за $i = n-1, n-2, \ldots, 1$ (обратен ход, заместване).

Нека означим с $\tilde{A}$ разширената матрица $\tilde{A} = (A|b)$. Тогава на първата стъпка на правия ход се умножава първия ред на $\tilde{A}$ по $-\frac{a_{i1}}{a_{11}}$ и се прибавя към реда с номер i, за всяко $i = 2, \ldots, n$.

$$\tilde{A} = \left[\begin{array}{cccccc|c} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} & b_i \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nj} & \cdots & a_{nn} & b_n \end{array}\right] \quad \cdot\left(-\tfrac{a_{21}}{a_{11}}\right) \; \cdot\left(-\tfrac{a_{i1}}{a_{11}}\right) \; \cdot\left(-\tfrac{a_{n1}}{a_{11}}\right)$$

Към така получената еквивалентната матрица $\tilde{A}^{(1)}$ се прилага следващата стъпка от еквивалентни трансформации:

$$\tilde{A}^{(1)} = \left[\begin{array}{ccccccc|c} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} & & b_1 \\ & a_{22}^{(1)} & \cdots & a_{2j}^{(1)} & \cdots & a_{2n}^{(1)} & & b_2^{(1)} \\ & \vdots & \ddots & \vdots & \ddots & \vdots & & \vdots \\ & a_{i2}^{(1)} & \cdots & a_{ij}^{(1)} & \cdots & a_{in}^{(1)} & & b_i^{(1)} \\ & \vdots & \ddots & \vdots & \ddots & \vdots & & \vdots \\ & a_{n2}^{(1)} & \cdots & a_{nj}^{(1)} & \cdots & a_{nn}^{(1)} & & b_n^{(1)} \end{array}\right] \begin{array}{l} \\ \\ \cdot\left(-\frac{a_{i1}}{a_{21}^{(1)}}\right) \\ \\ \\ \cdot\left(-\frac{a_{n1}}{a_{21}^{(1)}}\right) \end{array}$$

Правият ход завършва след изпълняване на $n-1$ стъпки. В резултат получаваме горната триъгълна матрица $\tilde{A}^{(n-1)}$:

$$\tilde{A}^{(n-1)} = \left[\begin{array}{ccccccc|c} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} & & b_1 \\ & a_{22}^{(1)} & \cdots & a_{2j}^{(1)} & \cdots & a_{2n}^{(1)} & & b_2^{(1)} \\ & & \ddots & \vdots & \ddots & \vdots & & \vdots \\ & & & a_{ij}^{(i)} & \cdots & a_{in}^{(i-1)} & & b_i^{(i-1)} \\ & & & & \ddots & \vdots & & \vdots \\ & & & & & a_{nn}^{(n-1)} & & b_n^{(n-1)} \end{array}\right]$$

На обратния ход горната триъгълна матрица $A^{(n-1)}$ се преобразува в диагонална матрица, като за целта се изпълняват $n-1$ стъпки. На първата последния ред на разширената матрица се умножава по $\frac{a_{nn}^{(i-1)}}{a_{i-1,n}^{(n-2)}}$ и се събира с i-тия ред за $i = 1, \ldots, n-1$:

$$\tilde{A}^{(n-1)(1)} = \left[\begin{array}{ccccccc|c} a_{11} & a_{12} & \cdots & a_{1j} & & \cdots & a_{1n} & b_1 \\ & \ddots & \ddots & \vdots & & \vdots & \vdots & \vdots \\ & & a_{ij}^{(i)} & \vdots & & \vdots & a_{in}^{(i-1)} & b_i^{(i-1)} \\ & & & \ddots & & \vdots & \vdots & \vdots \\ & & & & a_{n-1,n-1}^{(n-2)} & a_{n-1,n}^{(n-2)} & b_{n-2}^{(n-2)} & \\ & & & & & a_{nn}^{(n-1)} & b_n^{(n-1)} & \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \cdot\left(-\frac{a_{in}^{(i-1)}}{a_{nn}^{(n-1)}}\right) \end{array}$$

На втората стъпка се взема $(n-1)$-вия ред, като след нейното изпълнение се анулират извъндиагоналните елементи в $(n-1)$-вия стълб. Така, след $n-1$ стъпки на обратния ход, матрицата от коефициенти се свежда до диагоналната матрица:

$$\tilde{A}^{(n-1)(n-1)} = \left[\begin{array}{ccccccc|c} a_{11}^{(,n-1)} & & & & & & & b_1^{(,n-1)} \\ & \ddots & & & & & & \vdots \\ & & a_{ij}^{(i-1,n-i-1)} & & & & & b_i^{(i-1,n-i-1)} \\ & & & \ddots & & & & \vdots \\ & & & & a_{n-1,n-1}^{(n-2,1)} & & & b_{n-2}^{(n-2,1)} \\ & & & & & a_{nn}^{(n-1)} & & b_n^{(n-1)} \end{array}\right]$$

Решението на системата се получава след разделяне на i-тия ред със съответния диагонален елемент:

$$\tilde{A}^{(n-1)(n-1)} = \left[\begin{array}{cccc|c} a_{11}^{(,n-1)} & & & & b_1^{(,n-1)} \\ & \ddots & & & \vdots \\ & & a_{ij}^{(i-1,n-i-1)} & & b_i^{(i-1,n-i-1)} \\ & & & \ddots & \vdots \\ & & & a_{nn}^{(n-1)} & b_n^{(n-1)} \end{array}\right] \begin{array}{l} \\ \\ \cdot a_{11}^{(,n-1)} \\ \\ \cdot a_{ij}^{(i-1,n-i-1)} \\ \\ \cdot a_{nn}^{(n-1)} \end{array}$$

Така в трансформирания вектор $b^{(n-1,n-1)} = x$ получаваме решението на системата [90, 88].

Следната рекурентна формула описва изчислителната сложност на правия ход:

$$\begin{aligned} \mathcal{N}_{прав}(n) &= 2(n-1)n + n + \mathcal{N}_f(n-1) \\ &= \sum_{i=2}^{n}(2i^2 - i) = \sum_{i=1}^{n}(2i^2 - 1) - 1 \\ &= 2\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2} - 1 \sim \frac{2n^3}{3}. \end{aligned}$$

На обратния ход на всяка стъпка i се изпълняват $(i-1)$ събирания, $(i-1)$ умножения и едно деление. Така за изчислителната сложност получаваме:

$$\mathcal{N}_{обратен} = \sum_{i=1}^{n}(2i-1) = 2\frac{n(n+1)}{2} - n = n^2.$$

Следователно изчислителната сложност на метода на Гаус се определя от правия ход, понеже той има по-висока степен [90]:

$$\mathcal{N}^{Гаус} \sim \frac{2n^3}{3} = O(n^3).$$

В общия случай е възможно някой от диагоналните елементи $a_{ii}^{(k)} = 0$. Тогава, при реализация на описания по-горе метод на Гаус, ще стигнем до делене на нула. Тази проблем се преодолява като се смени редът i с друг ред j, в който $a_{ij}^{(k)} \neq 0$. По-точно, за да се избегне деленето с малки числа, при изпълнение на i-тата стъпка от правия ход i-тият ред се замества с реда, в който коефициентът $a_{ij}^{(k)}$ е най-голям по абсолютна стойност, т.е.

$$|a_{ij}^{(k)}| = \max_{j=i,\ldots,n} |a_{ij}^{(k)}|$$

[90]. Този вариант на метода се нарича *метод на Гаус с избор на главен елемент*.

За определени класове задачи изчислителната сложност на *метода на Гаус* може да се подобри. Така например *методът на квадратния корен* се прилага при симетрични и положително определени матрици. При него матрицата A се факторизира във вида $A = LL^t$, където L е долна триъгълна матрица. Изчислителната сложност на метода на квадратния корен е $\mathcal{N}^{LL^t}(n) \sim \frac{n^3}{3} = O(n^3)$, т.е. той е асимптотично два пъти по-бърз от *метода на Гаус*.

1.1.2 LU факторизация

LU факторизацията е изразяване на матрицата A като произведение на две триъгълни матрици $A = LU$. Тук L е долна триъгълна матрица с единици по главния диагонал, а U е горна триъгълна матрица. Тази факторизация се изчислява с помощта на модифициран метод на Гаус и се използва във високопроизводителните библиотеки (LAPACK[7], MKL[44], ACML[5], PLASMA[26], ATLAS[83] и други) за решаване на системи линейни уравнения. Описание на LU факторизацията е представено в [28].

Правият ход на метода на Гаус може да се запише във вида

$$\underbrace{L_{n-1}L_{n-2}\dots L_2L_1}_{\tilde{L}}A = U,$$

където $L_1, L_2, \dots, L_{n-1}$ са долни триъгълни матрици с единици по главния диагонал. Непосредствено се проверява, че $\tilde{L}$ и $L = \tilde{L}^{-1}$ са също долни триъгълни матрици с единици по главния диагонал. Така получаваме:

$$\tilde{L}A = U \quad \Longleftrightarrow \quad A = LU, \qquad L = \tilde{L}^{-1}.$$

След факторизацията на A, системата от линейни алгебрични уравнения се свежда до решаване на две системи с триъгълни матрици. Полагаме

$$L\underbrace{Ux}_{y} = b,$$

след което се:

1. Решава системата $Ly = b$ с право заместване;

2. Решава системата $Ux = y$ с обратно заместване.

Изчислителната сложност на факторизацията е $O\left(\frac{2}{3}n^3\right)$, докато правото и обратното заместване са със сложност $O\left(n^2\right)$.

1.2 Йерархични матрици. Методи за решаване на системи линейни уравнения с помощта на йерархична полусепарабелна компресия

Йерархичните матрици се използват за апроксимация на разредени по данни (data-sparse) матрици. Под разредени по данни се разбират матрици, които имат *структура* позволяваща апроксимация с помощта на компресирани матрици, които се записват с помощта на по-малък брой елементи. В общия случай, разредените по данни матрици не удовлетворяват условието да имат $O(n)$ ненулеви елемента. Хакбуш въвежда понятието Йерархичните матрици в [39], като разработва теория и алгоритми за работа с т.н. $\mathcal{H}$-матрици. Първоначално изследването на $\mathcal{H}$-матриците е свързано със създаването ефективни алгоритми за решаване на системи от линейни алгебрични уравнения получени при дискретизация на елиптични гранични задачи с метода на граничните елементи. По-късно е представено, че те са приложими за широк клас линейни системи, чиято матрица има извъндиагонални блокове с нисък ранг.

Методите използващи йерархични матрици са част от по-общата група от методи за решаване на системи чрез т.н. структурирани матрици. В [8] е направен обзор на съществуващите методи използващи такива матрици, включително и йерархични полусепарабелни матрици. В настоящата дисертация е изследвана ефективността на алгоритми на базата на този клас методи.

STRUMPACK (STRUctured Matrices PACKage) е паралелна софтуерна библиотека, която използва йерархична полусепарабелна компресия за решаване на системи от линейни алгебрични уравнения с плътни матрици [66]. Алгоритъмът включва три стъпки:

1. *Йерархична полусепарабелна компресия (апроксимация) на матрицата на системата.* Изчислителна сложност на тази стъпка е $O(r^2n)$, където r е максималният ранг на извъндиагоналните блокове на апроксимиращата матрица изчислен в процеса на компресия.

2. *ULV-подобна факторизация.* На тази стъпка се факторизира компресираната матрица. За целта се прилага вариант на метода на Гаус, подобен на използвания при представянето на LU факторизацията. Найнапред се изключват $O(r)$ неизвестни, след което се изключват останалите $O(n-r)$. Изчислителната сложност на тази стъпка е $O(r^2n)$.

3. *Решение.* Тази стъпка използва компресираната и факторизирана матрица от коефициенти на системата и дясната страна за намиране на решението. Изчислителната сложност на тази стъпка е $O(rn)$.

Така общата изчислителна сложност на метода е $O(r^2 n)$. Както ще видим по-късно, тази оценка е в сила при определени предположения.

1.2.1 Йерархична полусепарабелна компресия

В този раздел ще разгледаме накратко йерархичните полусепарабелни матрици (Hierarchically Semi-Separable – HSS). Те са въведени от Мартинсон в [56]. Теоретична обосновка на HSS методите е представена в [87]. В [66] са описани алгоритмите използвани в STRUMPACK за решаване на системи от линейни уравнения с плътни матрици. Йерархичната компресия може да се приложи върху всяка неособена матрица, но е *ефективна* само, ако изходящата матрица A има подходяща *структура* – т.е. извъндиагоналните ѝ блокове имат нисък ранг. Под *ефективна* компресия разбираме апроксимация на матрицата, което води до съществено намаляване на изчислителната сложност на операциите с компресираната матрицата, както и на паметта необходима за нейното съхраняване.

Означаваме HSS компресираната апроксимация на матрицата A с H. Алгоритъмът може да се опише по следния начин:

1. Разделяме матрицата A на четири блока. Предполагаме, че извъндиагоналните блокове имат нисък ранг (и могат да се декомпозират по сингулярни стойности (Singular Value Decomposition – SVD) или друга факторизация, която изчислява ранг):

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix}.$$

   Матриците U, B и V се наричат *генератори*. Ако извъндиагоналните блокове имат нисък ранг, матриците U са „високи и тънки“, B са малки и квадратни (или близки до квадратни) и матриците V са „ниски и широки“. Съотношението на броя на колоните и редовете зависи от ранга на извъндиагоналните блокове. D са непроменените диагонални блокове на изходящата матрица A. Означението „big“ ще бъде обяснено по-долу в точка 3.

2. Предполагаме, че диагоналните блокове D също имат извъндиагонални блокове с нисък ранг. Те се компресират по аналогичен начин, като процесът продължава рекурсивно. Второто ниво на рекурсивна компресия

има вида:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & U_3^{\text{big}} B_{3,6} V_6^{\text{big}*} \\ U_6^{\text{big}} B_{6,3} V_3^{\text{big}*} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix}$$

3. Съществува рекурсивна зависимост между *генераторите* на различните нива компресия. Това обяснява и използването на означенията „big“. В сила са следните зависимости:

$$U_3^{\text{big}} = \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 \quad \text{и} \quad V_3^{\text{big}} = \begin{bmatrix} V_1^{\text{big}} & 0 \\ 0 & V_2^{\text{big}} \end{bmatrix} V_3 \tag{1.2}$$

Третото ниво на рекурсивна HSS компресия се записва във вида:

$$A = \begin{bmatrix} \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix} & \begin{bmatrix} U_1^{\text{big}} & 0 \\ 0 & U_2^{\text{big}} \end{bmatrix} U_3 B_{3,6} V_6^* \begin{bmatrix} V_4^{\text{big}*} & 0 \\ 0 & V_5^{\text{big}*} \end{bmatrix} \\ \begin{bmatrix} U_4^{\text{big}} & 0 \\ 0 & U_5^{\text{big}} \end{bmatrix} U_6 B_{6,3} V_3^* \begin{bmatrix} V_1^{\text{big}*} & 0 \\ 0 & V_2^{\text{big}*} \end{bmatrix} & \begin{bmatrix} D_4 & U_4^{\text{big}} B_{4,5} V_5^{\text{big}*} \\ U_5^{\text{big}} B_{5,4} V_4^{\text{big}*} & D_5 \end{bmatrix} \end{bmatrix} \tag{1.3}$$

HSS компресията на матрица A с размерност $n \times n$ се основава на рекурсивно клъстеризиране (разделяне) на индексното множество $\{1, \ldots, n\}$. Това се представя чрез *HSS дървото* на A. Всеки връх τ се асоциира с подмножество I_τ от множеството $\{1, \ldots, n\}$. Коренът на дървото се асоциира с цялото множество от индекси $\{1, \ldots, n\}$, а всеки връх τ, който не е листо с наследници ν_1 и ν_1,

$$I_\tau = \nu_1 \cup \nu_2.$$

Лисата на дървото представят разделяне на $\{1, \ldots, n\}$. Броят на нивата на дървото зависи от нивата на компресия и самият брой няма значение, стига да няма празни върхове. HSS представянето на A има следната дървовидна структура:

- За всяко листо τ съответният диагонален блок $D_\tau = A(I_\tau, I_\tau)$ се оставя непроменен (некомпресиран).
- За всяко τ, което не е листо, с наследници ν_1 и ν_1, съответните извъндиагонални блокове $A_{\nu_1,\nu_2} = A(I_{\nu_1}, I_{\nu_2})$ и $A_{\nu_2,\nu_1} = A(I_{\nu_2}, I_{\nu_1})$ се представят като:

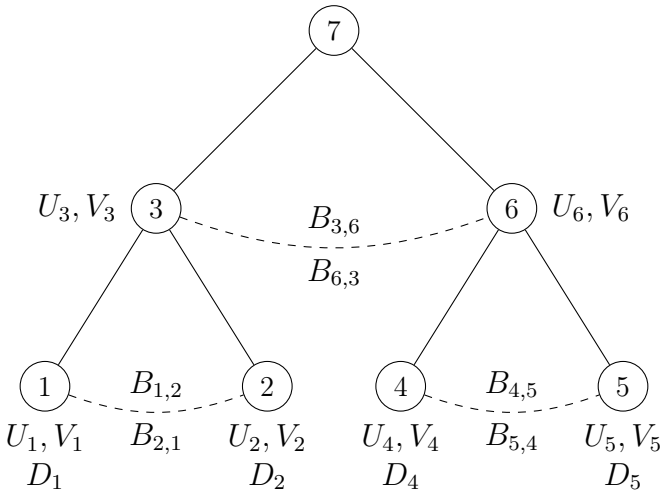
$$A_{\nu_1,\nu_2} = U_{\nu_1}^{\text{big}} B_{\nu_1,\nu_2} V_{\nu_2}^{\text{big}*}.$$

Освен това рекурсивното представяне от равенство 1.2 също е в сила:

$$U_\tau^{\text{big}} = \begin{bmatrix} U_{\nu_1}^{\text{big}} & 0 \\ 0 & U_{\nu_2}^{\text{big}} \end{bmatrix} U_\tau \quad \text{и} \quad V_\tau^{\text{big}} = \begin{bmatrix} V_{\nu_1}^{\text{big}} & 0 \\ 0 & V_{\nu_2}^{\text{big}} \end{bmatrix} V_3$$

Генераторите с означения „big“ могат да не се изчисляват извън листата на дървото. U във връх τ се изчислява от U_τ и от U^{big} в наследниците ν_1 и ν_2. В листата $U = U^{\text{big}}$.

Фигура 1.1 показва дървото съответстващо на предходния пример за тринивова компресия.



Фигура 1.1: Тринивово HSS дърво.

В общия случай равенство 1.3 не е точно, а приблизително. Това означава, че в резултат на HSS компресията получаваме апроксимация на A, която означаваме с H, т.е.

$$H \approx A.$$

Когато това се подразбира, няма да използваме в описанието на алгоритмите H вместо A. За целта се избира подходящ трешхолд (прагова стойност) ε, който е необходим при изчисляването на *генераторите*. Когато се използва по-голям трешхолд, се получават по-малки *генератори* и съответно компресираната матрица заема по-малко памет и позволява по ефективни операции с нея, но това е за сметка на точността. При избор на по-малък трешхолд е точно обратното – точността а компресираната матрица H е по-висока, но изчислителната ефективност е по-малка.

Както ще покажем в следващите глави, подреждането на неизвестните при асемблиране на матрицата A влияе съществено на ефективността на HSS компресията. Ако матрицата A се пренареди произволно, това с голяма вероятност може да унищожи каквато и да е подходяща за този метод *структура*. Тази специфика на HSS компресията е разгледана и в [6, 62, 82].

За определени класове задачи е възможно така да се пренаредят неизвестните, че да се подобри съществено *структурата* на матрицата на системата. Така например в [64] са разгледани няколко метода за клъстеризация при използването на хребетообразна регресия с ядро (Kernel Ridge Regression). В Глава 3 ще предложим и анализираме няколко метода за пренареждане на неизвестните за система от линейни алгебрични уравнения, получена при дискретизация на елиптична задача с *дробна* степен на оператора на Лаплас (дробно-дифузионна задача).

1.2.2 Компресия със случайни извадки

Алгоритъмът за HSS компресия в STRUMPACK се основава на използването на случайни извадки (randomized sampling), който прилага умножаване на множество от случайни вектори с изходната матрица A. Този метод е предложен от Мартинсон в [56]. В алгоритъма не се изисква да разполагаме с матрицата A в явен вид. Вместо това е нужна само функция за умножение на A с вектор. Премуществата на този подход, както и адаптивен алгоритъм за случайни извадки, са разгледани от Горман и др. в [37]. Използването на случайни извадки е полезно също така при интегрирането на HSS ядра в солвъри за задачи с разредени матрици [33, 86].

В общия случай изчислителната сложност на умножението на плътна матрица с вектор е $O(n^2)$. Това води до сложност на HSS компресирането $O(rn^2)$. Нека припомним, че с r означаваме максималния ранг на извъндиагоналните блокове, намерен в процеса на компресия. За определени класове задачи r е много по-малко от n. Така например за двумерни задачи на Поасон (МКЕ) r е константа, а за тримерни задачи на Хелмхолц (МГЕ) расте бавно при нарастване на n [85]. Ако разполагаме с бърз алгоритъм за умножение на компресираната матрица с вектор, сложността на компресията може да се намали до $O(r^2n)$.

Нека предположим, че максималният ранг r се знае предварително (на практика той се изчислява в процеса на компресията). Нека R^r и R^c са $n \times d$ „високи и тънки“ матрици, където $d = r + p$, p е малка константа на (oversampling constant). В [56] се препоръчва $p = 10$. $S^r = AR^r$ и $S^c = A * R^c$ са извадки за базисите по редове (^{r}) и колони (^{c}) от матрицата A. За възел τ, който не е лист с наследници ν_1 и ν_2, D_τ се дефинира като:

$$D_\tau = \begin{bmatrix} D_{\nu_1} & A_{\nu_1,\nu_2} \\ A_{\nu_2,\nu_1} & D_{\nu_2} \end{bmatrix}$$

На ниво на компресия l некомпресираните блокове са представени с $n \times n$ блочно-диагонална матрица

$$D^{(l)} = \begin{bmatrix} D_{\tau_1} & & & \\ & D_{\tau_2} & & \\ & & \ddots & \\ & & & D_{\tau_q} \end{bmatrix},$$

където върховете на графа на ниво l са $\{\tau_1, \tau_2, \ldots, \tau_q\}$. За всяко ниво се конструират матрици на извадките по редове $S^{r,(l)}$ и колони $S^{c,(l)}$:

$$S^{r,(l)} = (A - D^{(l)})R^r = AR^r - D^{(l)}R^r$$
$$S^{c,(l)} = (A - D^{(l)})R^c = AR^c - D^{(l)}R^c$$

Тези изчисления се реализират от листата към корена на HSS дървото.

Основен компонент на алгоритъма на случайните извадки е интерполативната декомпозиция (Interpolative Decomposition (ID))[19]. ID изчислява факторизацията на $m \times n$ матрица Y с ранг k, като изразява колоните на Y като линейна комбинация от техни подмножества:

$$[X, J] = \mathrm{ID}(Y) : Y = Y(:, J)X,$$
$$Y \text{ е } m \times k \text{ матрица }, \quad X \text{ е } k \times n \text{ матрица.}$$

Както отбелязахме по-горе, равенствата при прилагене на HSS компресия са приблизителни. Въвежда се параметър за праг на грешката ε, който се използва за получаване на приблизителна линейна комбинация:

$$[X, J] = \mathrm{ID}(Y, \varepsilon) : Y \simeq Y(:, J)X,$$
$$Y \text{ е } m \times k' \text{ матрица }, \quad X \text{ е } k' \times n \text{ матрица,}$$

където изчисленият ранг $k' \leqq n$.

В [37] се въвежда използването на два прага – ε_{rel} за относителна и ε_{abs} за абсолютна грешка. В числените експерименти, представени в Глави 2, 3 и 4, ще използваме една и съща стойност за абсолютния праг $\varepsilon_{\mathrm{abs}} = 10^{-8}$, като вариираме относителния праг.

Интерполативната декомпозиция ID може да се реализира (например) с QR факторизация. Така получаваме:

$$\begin{aligned} Y &= QR\Pi^{-1} \\ &= Q\left[R_1 R_2\right]\Pi^{-1} && = (QR_1)(\left[I \; R_1^{-1}R_2\right]\Pi^{-1}) \\ &= Y(:, J)X, \end{aligned}$$

където Π е пренареждане (пермутация) по стълбове, R_1 е матрица с размерност $k \times k$ и QR_1 е първият стълб на пренареденатa матрица Y.

Алгоритъмът за компресия може да се представи в следния обобщен вид:

1. Генериране на случайни матрици R^r и R^c с размерност $n \times d$.
2. Изчисляване на извадките $S^r = AR^r$ и $S^c = AR^R$.
3. Обхождане на дървото от листата към корена, като на всяко ниво се изпълняват операциите:
   а) Конструиране на локалните извадки.
   б) Изчисляване на *генераторите* с интерполативна декомпозиция.
   в) Актуализация на извадките и случайните вектори, с цел ускоряване на създаването на локалните извадки в следващите нива.

В резултат от интерполативната декомпозиция се получава подматрицата на матрицата A $B_{\nu_1,\nu_2} = A(I^r_{\nu_1}, I^c_{\nu_2})$, както и структурата на *генераторите* U_τ V_τ:

$$U_\tau = \Pi^r_\tau \begin{bmatrix} I \\ E^r_\tau \end{bmatrix} \qquad \text{и} \qquad V_\tau = \Pi^c_\tau \begin{bmatrix} I \\ E^c_\tau \end{bmatrix} \tag{1.4}$$

Тази структура се използва впоследствие при ULV-подобната факторизация.

Компресията на матрицата дава възможност за бързо умножение с вектор с изчислителна сложност $O(nr)$.

1.2.3 ULV-подобна факторизация и решение

Компресираната матрица H в HSS форма може да се факторизира със специален вид LU факторизация, наречена ULV факторизация [17]. Тази факторизация използва ортогонални трансформации за последователно изключване на първите $n - r$ неизвестни. Останалите r неизвестни се изключват с LU факторизация. В STRUMPACK е реализирана ULV-подобна факторизация. При нея вместо ортогонални трансформации се използва HSS структурата на компресираната матрицата H. Алгоритмите за тази компресия са описани в [66].

В съответствие с равенство 1.4 *генераторите* U се представят във вида $U_\tau = \Pi^r_\tau \begin{bmatrix} I \\ E^r_\tau \end{bmatrix}$. За целта се използва трансформацията

$$\Omega = \begin{bmatrix} -E^r_\tau & I \\ I & 0 \end{bmatrix} \Pi^{r-1}_\tau, \qquad \Omega_\tau U_\tau = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

След прилагане на трансформациите Ω_1 и Ω_2 към HSS компресираната матрица

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} D_1 & U_1^{\text{big}} B_{1,2} V_2^{\text{big}*} \\ U_2^{\text{big}} B_{2,1} V_1^{\text{big}*} & D_2 \end{bmatrix},$$

получаваме равенството

$$\begin{bmatrix} \Omega_1 & 0 \\ \Omega_2 & 0 \end{bmatrix} A = \begin{bmatrix} \Omega_1 D_1 & \begin{bmatrix} 0 \\ B_{1,2} V_2^* \end{bmatrix} \\ \begin{bmatrix} 0 \\ 2B_{2,1} V_1^* \end{bmatrix} & \Omega_2 D_2 \end{bmatrix}.$$

За всеки връх τ, матрицата $W_\tau = \Omega_\tau D_\tau$ се разделя във вида $W_\tau = \begin{bmatrix} W_{\tau;1} \\ W_{\tau;2} \end{bmatrix}$, където $W_{\tau;2}$ има r_τ^r реда. Изпълнява се LQ декомпозиция на $W_{\tau;1}$: $W_{\tau;1} = \begin{bmatrix} L_\tau & 0 \end{bmatrix} Q_\tau$. Така се получава представянето

$$\begin{bmatrix} \Omega_1 & 0 \\ 0 & \Omega_2 \end{bmatrix} A \begin{bmatrix} Q_1^* & 0 \\ 0 & Q_2^* \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} L_1 & 0 \end{bmatrix} & 0 \\ W_{1;2} Q_1^* & B_{1,2} V_2^* Q_2^* \\ 0 & \begin{bmatrix} L_2 & 0 \end{bmatrix} \\ B_{2,1} V_1^* Q_1^* & W_{2;2} Q_2^* \end{bmatrix} =$$

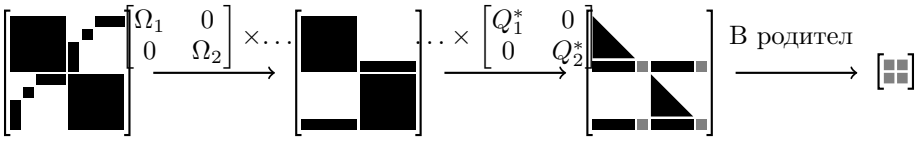
$$= \begin{bmatrix} L_1 & 0 & 0 & 0 \\ W_{1;2} Q_{1;1}^* & W_{1;2} Q_{1;2}^* & B_{1,2} V_2^* Q_{2;1}^* & B_{1,2} V_2^* Q_{2;2}^* \\ 0 & 0 & L_2 & 0 \\ B_{2,1} V_1^* Q_{1;1}^* & B_{2,1} V_1^* Q_{1;2}^* & W_{2;2} Q_{2;1}^* & W_{2;2} Q_{2;2}^* \end{bmatrix}$$

Матриците L_1 и L_2 са долни триъгълни и съответните неизвестни могат да се пресметнат със стандартно последователно заместване. Подматрицата

$$\begin{bmatrix} W_{1;2} Q_{1;2}^* & B_{1,2} V_2^* Q_{2;2}^* \\ B_{2,1} V_1^* Q_{1;2}^* & W_{2;2} Q_{2;2}^* \end{bmatrix}$$

съответства на останалите $O(r)$ неизвестни. Тя се сформира във върха родител, където процесът се повтаря. В корена на дървото остават $O(r)$ неизвестни за система с плътна матрица, за решаването на която се прилага LU факторизация. Процесът на ULV-подобна факторизация е илюстриран на Фигура 1.2

След прилагане на ULV-подобната факторизация системата от линейни алгебрични уравнения $Ax = b$ се свежда до решаване на две системи с триъгълни матрици. Изчислителната сложност на тази последна стъпка е $O(rn)$ [66].



Фигура 1.2: ULV-подобна факторизация.

Глава 2

Метод на граничните елементи за числено решаване на двумерна задача за обтичане на крилни профили

В тази глава се разглежда числен метод за компютърна симулация на ламинарен поток около крилни профили на Жуковски. Методът на граничните елементи има съществени предимства при числено решаване на външни гранични задачи за многосвързани области с криволинейни граници. В дисертацията е приложен методът описан в [63], като е разработена програмна реализация за обтичане на каскада от крилни профили от идеален флуид. Методът е базиран на колокация на сплайни с на части линейна интерполация. Допълнителни детайли за метода на граничните елементи могат да бъдат намерени в [12]. В [69] е представен паралелен код на програмния език C за решаване на разглежданата задача.

След дискретизиране на интегралните уравнения по метода на граничните елементи се получава система от линейни уравнения с плътна матрица. Резултатите от прилагане на изследваните в дисертацията методи и алгоритми се сравняват с резултати получени по метода на последователно изключване (метод на Гаус) реализиран в няколко популярни софтуерни пакета. Върху CPU процесори в сравнителния анализ на производителността използваме Intel Math Kernel Library (MKL) и Parallel Linear Algebra for Scalable Multi-core

Architectures (PLASMA), докато за Intel Xeon Phi копроцесорите (накратко наричани MIC от името на архитектурата Many Integrated Core) MKL производителността се сравнява и с Matrix Algebra on GPU and Multicore Architectures (MAGMA) за MIC архитектура (наричан за кратко MAGMA MIC).

Резултати включени в тази глава на дисертация са публикувани в:

[72] D. Slavchev and S. Margenov. Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, *Numerical Methods and Applications*, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8

[73] D. Slavchev and S. Margenov. Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around Airfoils. In K. Georgiev, M. Todorov, and I. Georgiev, editors, *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, pages 369–381, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30

[74] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing

2.1 Постановка на задачата

2.1.1 Метод на граничните елементи за пресмятане на токовата функция на идеален флуид в неограничена двумерна област

Нека $\Omega \subset \mathbb{R}^2$ е неограничена многосвързана област с достатъчно гладка вътрешна граница S. Токовата функцията Ψ удовлетворява уравнението на Лаплас

$$\nabla^2 \Psi \equiv \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2} = 0 \tag{2.1}$$

в $\Omega \subset \mathbb{R}^2$ и може да се представи във вида

$$\Psi(P) = -\frac{1}{4\pi} \int_S \gamma(\sigma) \ln\left(r^2(P,Q)\right) d\sigma_Q + \Psi_\infty(P) + C_0, \qquad P \in \Omega. \tag{2.2}$$

където $r^2(P,Q) = (x-\xi)^2 + (y-\eta)^2$, $P = (x,y)$, $Q = (\xi,\eta)$ и $d\sigma_Q$ е мярка върху S. Първият член на дясната страна съответства на прост слой с плътност $\gamma(\sigma)$, $\Psi_\infty(P)$ е хармонична функция, добавена, за да бъдат удовлетворени условията на външната граница, т.е. при $P \to \infty$. При тези предположения за полето на скоростите $\overrightarrow{C} = (u,v)$,

$$u = \frac{\partial \Psi}{\partial y}, \qquad v = \frac{\partial \Psi}{\partial x}$$

са в сила уравненията

$$u = \frac{1}{2\pi}\int_S \gamma(\sigma)\frac{y-\eta}{r^2}d\sigma, \quad v = -\frac{1}{2\pi}\int_S \gamma(\sigma)\frac{x-\zeta}{r^2}d\sigma.$$

Използват се и следните уравнения

$$\left(\frac{\partial \Psi}{\partial n}\right)_e = -\frac{\gamma(s)}{2} - \frac{1}{2\pi}\int_S \frac{cos(r,n_s)}{r}d\sigma, \quad s \in S$$
$$\left(\frac{\partial \Psi}{\partial n}\right)_i = \quad \frac{\gamma(s)}{2} - \frac{1}{2\pi}\int_S \frac{cos(r,n_s)}{r}d\sigma, \quad s \in S$$

в които $\partial\Psi/\partial n$ е производната по външната нормала на границата S. Тук с $(.)_i$ и $(.)_e$ са означени съответно границата на функцията в скобите от вътрешната и външната страна на границата на Ω. Скокът на $\partial\Psi/\partial n$ по границата S съответства на плътността на слоя $\gamma(s)$

$$\left(\frac{\partial \Psi}{\partial n}\right)_i - \left(\frac{\partial \Psi}{\partial n}\right)_e = \gamma(s), \quad s \in S$$

Нека означим с $\overrightarrow{n}$ и $\overrightarrow{t}$ външния нормален и тангенциален единичен вектор. Тогава

$$\overrightarrow{C} = c_n \cdot \overrightarrow{n} + c_t \cdot \overrightarrow{t}$$

и следователно

$$\left(\frac{\partial \Psi}{\partial n}\middle|\right) = -(c_t)_{i.e}$$
$$(C_t(P))_e - (C_t(P))_i = \gamma(P), \quad P \in S.$$

Граничните условия на метод на граничните елементи могат да се запишат във вида

$$\left|\overrightarrow{C}(P)\right| = |\gamma(P)|, \quad P \in S.$$

2.1.2 Обтичане на крилни профили

В този раздел ще разглеждаме задачата за обтичане на крилни профили на Жуковски. Приемаме, че флуидното течение в безкрайност е с хомогенна скорост $\overrightarrow{C}_\infty = (1, 0)$. Тук с S са означени контурите на крилните профили. За разглежданата задача токовата функцията Ψ удовлетворява уравнението на Лаплас (2.1). Крилните профили S са непроницаеми. Следователно е в сила граничното условие $\Psi|_S = K = \text{const}$. За да удовлетворим условията в $\overrightarrow{C}_\infty$, избираме такова Ψ_∞, че

$$\overrightarrow{C}_\infty = \left(\frac{\partial \Psi_\infty}{\partial y}, -\frac{\partial \Psi_\infty}{\partial x} \right).$$

В конкретната задача използваме

$$\Psi_\infty(P) = \gamma_\infty(P).$$

Така интегралното уравнение (2.2) се записва във вида

$$\gamma(P) - \frac{1}{4\pi} \int_S \gamma(\sigma) \ln \left(r^2(P, Q) \right) d\sigma_Q + C = 0. \tag{2.3}$$

За получаване на единствено решение на интегралното уравнение се използва условието на Рунге-Кута $\gamma(A) = 0$, където A са върховете на острите ъгли на крилните профили.

2.1.3 Дискретизация

За числено решаване на интегралното уравнение (2.3) прилагаме метода на граничните елементи. Така непрекъснатата задача се свежда до система от линейни алгебрични уравнения, която можем да запишем във вида:

$$(A\gamma)\, s = f(s).$$

Приближеното решение се търси във вида

$$\gamma_h(S) = \sum_{i=1}^{n} \gamma_i \phi_i(s).$$

Тук $\{\phi_i(s)\}_{i=1}^n$ е Лагранжевият базис на пространството от на части линейни функции по границата S, върху която е дефинирана мрежата S_h. С $\gamma_i =$

$\gamma_h(s_i)$, $i = 1, \ldots, n$ са означени неизвестните стойности на приближеното решение във възлите на мрежата. За реализация на метода на граничните елементи прилагаме метод на колокацията с колокационни точки в средите на елементите от S_h. Следвайки [69] получаваме системата от линейни уравнения

$$\sum_{i=1}^{n} \gamma_i \Psi_{ji} = f_j, \qquad j = 1, 2, \ldots, n, \tag{2.4}$$

където $\Psi_{ji} = \Psi_i(s_j)$, $f(s_j) = f_j$, $\Psi_i(s) = (A\phi_i)(s)$. Матрицата в метода на граничните елементи D има следната блочна структура

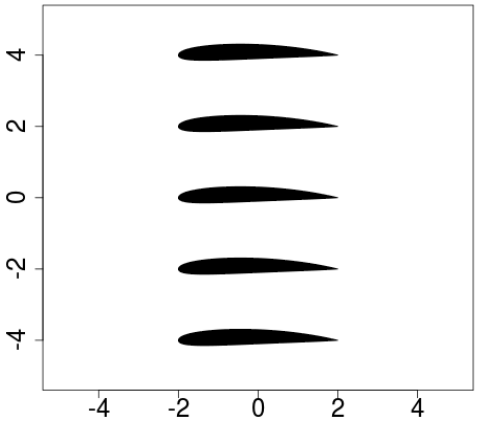
$$D = \begin{pmatrix} D_{11} & D_{12} & \ldots & D_{1m} \\ \vdots & \vdots & D_{lk} & \vdots \\ D_{m1} & D_{m2} & \ldots & D_{mm} \end{pmatrix}.$$

Тук m е броят на крилните профили, а D_{lk} е блок на матрицата съответстващ на k-тият крилен профил и колокационните точки на l-тият крилен профил.

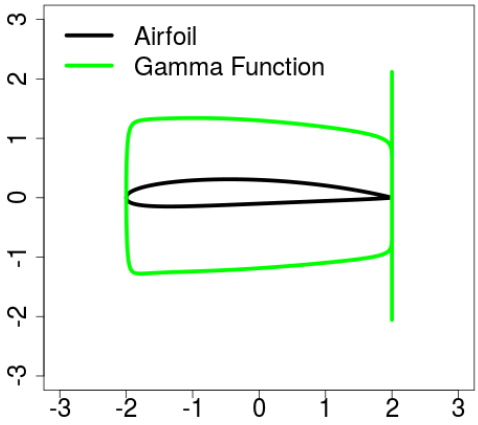
2.1.4 Постановка на числените експерименти

В Раздел 2.2 са представени числени експерименти за задачата за обтичане на пет крилни профила разположени вертикално един над друг, както е показано на Фигура 2.1а. Границите на профилите са дискретизирани с гранични елементи. На всеки граничен елемент съответства линейно уравнение от системата (2.4) относно средата на елемента P и $\gamma(P)$. Числеността решение на задачата върху границата е на части линейна функция определена от нейните възлови параметри $\gamma(s_i)$, $i = 1, \ldots, n$. Така стойността на $\gamma(s_i)$ определя приноса на граничния елемент s_i за формиране на токовата функция. Решението по метода на граничните елементи на разгледаната задача е визуализирано на Фигура 2.1б за средния крилен профил.

След като линейната система (2.4) е решена (т.е. стойностите на $\gamma(s_i)$, $i = 1, \ldots, n$ са определени), може да се пресметне векторното поле $\overrightarrow{C} = (u, v)$ във всяка точка от изчислителната област на разглежданата задача за обтичане. От векторната норма $\left|\overrightarrow{C}\right|_P$ в точка P може да се изчисли наляганието и скаларната компонента скоростта. На Фигура 2.2а е визуализирана скаларната компонента на скоростта $\left|\overrightarrow{C}\right|_P$. Също така могат да се намерят токовите линии представени на Фигура 2.2б.

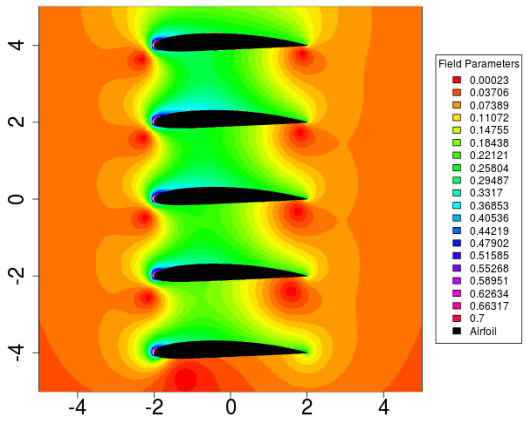


(а) Крилни профили

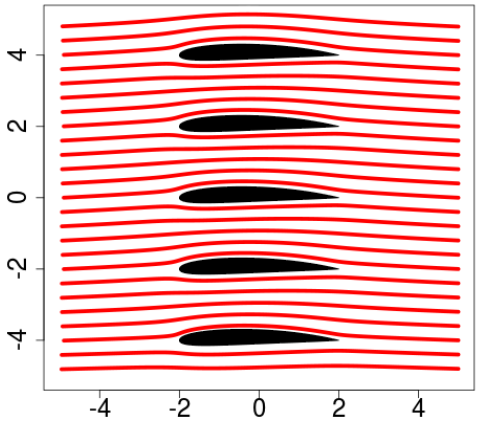


(б) Стойности на γ за средния крилен профил

Фигура 2.1: Постановка на задачата и нейното числено решение по метода на граничните елементи



(а) Скаларна компонента на скоростта



(б) Токови линии

Фигура 2.2: Визуализация на числено решение по метода на граничните елементи на задачата за обтичане на профили на Жуковски

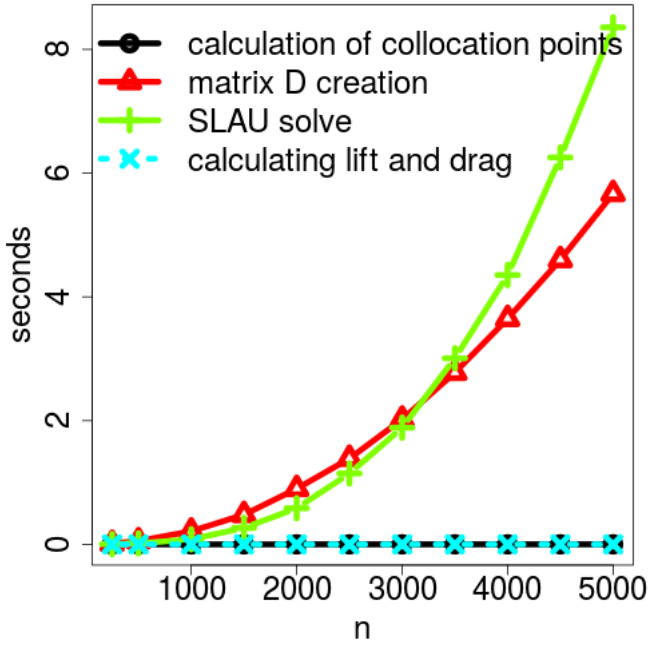
2.2 Анализ на числени експерименти върху компютърни системи с обща памет

Резултатите включени в тази глава са за високопроизводителни изчислителни системи с обща памет. Те са публикувани в [73], където е анализиран солвър реализиращ метода на Гаус (или съответната LU факторизация) и в [72] за солвъра базиран на Йерархична компресия. В [74] са публикувани резултати

за системи с разпределена памет.

Софтуерната имплементация на изследваните алгоритми е на програмния език C. OpenMP стандартът се използва за реализация на паралелните алгоритми.

Анализът на изчислителната сложност е в основата на методологията на изследване на ефективността на методите и алгоритмите за решаване на задачи с голяма дискретна размерност (в случая много големи n). Тук ще направим сравнителен анализ на сложността на основните части (блокове) на метода на граничните елементи за разглежданата двумерна задача за обтичане. След това ще покажем как този тип асимптотични оценки съответстват на изчислителните времена. Построяването на мрежата за дискретизация на границите на крилните профили, в т.ч. пресмятането на координатите на възлите на колокация, има сложност $O(n)$. Формирането на матрицата D има сложност $O(n^2)$. Изчислителната сложност на метода на Гаус за решаване системи линейни алгебрични уравнения с плътни матрици е $O(n^3)$ [46]. Сложност $O(n)$ има и изчисляването на коефициентите на челно съпротивление и подемна сила.



Фигура 2.3: Сравнение на изчислителното време за решаване на системата линейни уравнения и времената за реализиране на другите части от алгоритъма

На Фигура 2.3 са представени времена за изпълнение на анализираните части на алгоритъма. Ясно се вижда, че над определен размер на задачата

($n > 3\ 000$) изчислителното време за решаване на системата започва да доминира над генерирането на матрицата D, като за $n > 4\ 000$ това е силно изразено. Изчисляването на полето на скоростите и на токовите линии има сложност $O(nk)$. Тук k е броят на точките във външната област, в които се търсят стойностите на токовата функцията и скоростите. Ще отбележим, че ако тези пресмятания се правят върху равномерна мрежа със стъпка близка до стъпката на дискретизация на границите на крилните профили ще получим $k = O(n^2)$. В [69] е разгледана паралелизация на тези изчисления, докато фокусът на изследване в тази работа е върху най-тежката изчислителна част – решаването на системата линейни алгебрични уравнения.

2.2.1 LU факторизация

Сравнителният анализ на паралелната производителност и скалируемост на метода на Гаус е направен на базата на преки солвъри реализиращи LU факторизация. За целта използваме следните библиотеки за паралелни високопроизводителни изчисления:

- Експериментите върху процесори с общо предназначение (CPU): Intel®'s Math Kernel Library (MKL) и Parallel Linear Algebra Software for Multicore Architectures (PLASMA);

- Експериментите върху MIC копроцесор: Intel®'s Math Kernel Library (MKL) и Matrix Algebra on GPU Multicore Architectures (MAGMA) MIC.

CPU процесори с обща памет

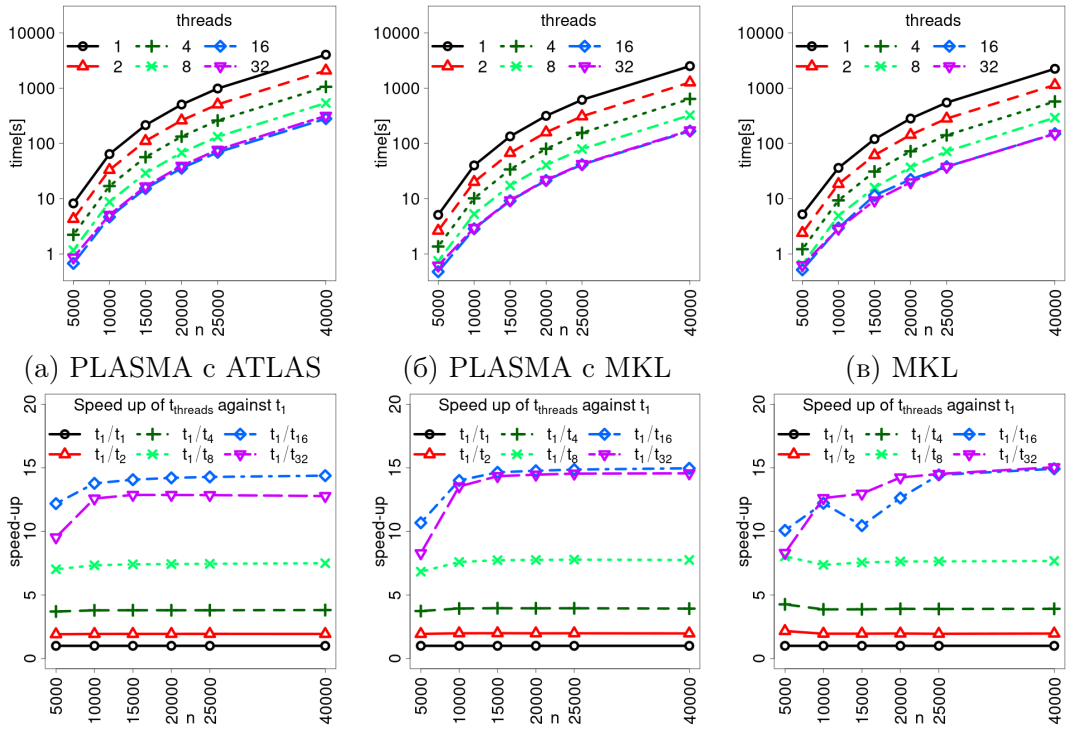
В тази точка изследваме паралелната ефективност на софтуерните библиотеки PLASMA и MKL върху един сървър от суперкомпютъра AVITOHOL. Всеки сървър има по два CPU процесора Intel Xeon E5-2650v2 8C 2.6GHz всеки с по 8 ядра или общо 16 ядра. Този модел процесори позволява паралелното използване на по две нишки, чрез хипертрединг, до максимум 32 нишки.

PLASMA (Parallel Linear Algebra Software for Multicore Architectures) [15, 25, 41] е софтуерен пакет за решаване на системи линейни уравнения с плътни матрици, който имплементира функциите на стандарта LAPACK. PLASMA използва оптимизирани блочни алгоритми.

Ефективността на PLASMA е базирана на силно оптимизирания пакет BLAS (Basic Linear Algebra Subprograms), в който са реализирани основните операции в линейната алгебра – умножение на вектори, матрици и вектори с матрици. За това ниво на изчисления използваме MKL BLAS и ATLAS (Automatically Tuned Linear Algebra Software) BLAS [83, 84].

Експериментите с MKL и PLASMA с MKL BLAS са компилирани с `icc` 17.0.2 20170213 и MKL 2017 update 2. Експериментите с PLASMA и ATLAS са компилирани с `gcc` версия 7.2.0, тъй като пакетът ATLAS е оптимизиран за използване на `gcc` и авторите не предлагат оптимизирани флагове за icc компилатора [84].

В Таблица 2.1 и Фигура 2.4 са представени резултатите от проведените числени експерименти за решаване на линейните системи получени при прилагане на метода на граничните елементи за дискретизация на задачата за обтичане на крилни профили. Сравнени последователните и паралелни времена за PLASMA + ATLAS, PLASMA + MKL и MKL за $n = 5\,000$ и $n = 40\,000$, варирайки броя на нишките.



(а) PLASMA с ATLAS (б) PLASMA с MKL (в) MKL

(г) PLASMA с ATLAS (д) PLASMA с MKL ускорение (е) MKL ускорение

Фигура 2.4: Сравнение на времената (а-в) и ускоренията (г-е) за решаване на системите с използване на софтуерните библиотеки PLASMA с ATLAS, PLASMA с MKL и MKL.

Резултатите показват добро ускорение за всички тествани библиотеки до 16 нишки. Ускорението достига до 15 и е близо до теоретичния максимум 16. Виждаме също така, че използването на хипертрединг за разглежданите зада-

Таблица 2.1: Последователни и паралелни времена за решаване на системата върху CPU процесори с обща памет

Софтуер		Plasma + ATLAS		PLASMA + MKL		MKL	
Нишки	n	време [s]	ускорение	време [s]	ускорение	време [s]	ускорение
1	5 000	8.42	1.00	5.03	1.00	5.30	1.00
16	5 000	0.67	12.57	0.47	10.69	0.47	11.26
32	5 000	0.88	9.59	0.65	7.76	0.65	8.12
1	40 000	4008.76	1.00	2497.12	1.00	2233.93	1.00
16	40 000	282.94	14.17	166.41	15.01	147.64	15.13
32	40 000	325.17	12.33	169.58	14.73	148.59	15.03

чи не е ефективно. За MKL и PLASMA с MKL BLAS времената за изпълнение за 32 нишки са почти същите като тези за 16 нишки, докато за PLASMA с ATLAS BLAS използването на хипертрединг води до съществено забавяне.

Паралелната ефективност на PLASMA с MKL и MKL е близка, като достига 94% за най-голямата задача ($n = 40\ 000$). И двата варианта превъзхождат PLASMA с ATLAS повече от 1.5 пъти. Възможно обяснение за това е, че `gcc` компилатора не използва векторизация толкова добре, колкото `icc` или MKL има по-ефективни BLAS функции от ATLAS.

MIC ускорители

В този раздел анализираме паралелната ефективност на Intel Xeon Phi 7120P копроцесори (MICs). MIC-овете са проектирани за масивни паралелни и векторни изчисления необходими при високопроизводителните изчисления. Използваният модел MIC има 61 ядра, като всяко ядро може да изпълнява едновременно инструкции от 4 нишки, т.е. могат да се използват максимум 244 нишки. Копроцесорите могат да се използват в два режима:

1. Native: Цялата програма се изпълнява върху копроцесора;
2. Offload: Програмата се изпълнява върху CPU процесора, като само специфични части от нея се изпращат на копроцесора.

В дисертацията се използва вторият режим, като MKL и MAGMA MIC библиотеките се грижат за изчисленията и данните, които се изпращат към копроцесора. В този режим едно от ядрата на MIC-а се запазва за комуникацията с CPU процесора и съответно могат да се ползват до 60 ядра (до 240 нишки).

За да се използва MAGMA MIC е нужно да се зададат следните променливи на средата:

```
MKL_MIC_ENABLE=1
```

```
OFFLOAD_DEVICES=0
MIC_ENV_PREFIX=MIC
MIC_OMP_NUM_THREADS=m
MIC_KMP_AFFINITY="proclist=[1-m],explicit"
```

Тук:

- `MKL_MIC_ENABLE` активира използването на копроцесора;
- `OFFLOAD_DEVICES` задава кои копроцесори ще се използват. Ние използваме един с индекс 0;
- `MIC_OMP_NUM_THREADS` задава броя на OpenMP нишките, които ще се използват;
- `MIC_ENV_PREFIX` задава префикса който ще се използва за MIC средата;
- `MIC_KMP_AFFINITY` задава броя ядра m, които ще се използват;
- m се замества с броят нишки/ядра, които ще се използват; Разглеждаме експерименти с няколко стойности на m, като максимумът е $m = 240$.

За MKL използваме опцията компилатора да помогне за изпращането на инструкциите към копроцесора (compiler-assisted offload). Това позволява една и съща програма да се използва и са CPU и за MIC експериментите. Следните променливи на средата са използвани:

```
MKL_MIC_ENABLE=1
OFFLOAD_DEVICES=0
MIC_ENV_PREFIX=MIC
MIC_OMP_NUM_THREADS=m
MIC_KMP_AFFINITY="proclist=[1-m],explicit"
MKL_HOST_WORKDIVISION=0
MKL_MIC_WORKDIVISION=1
MKL_MIC_THRESHOLDS_DGEMM=20,20,20
```

Първите пет параметъра са същите, докато останалите са, както следва:

- `MKL_HOST_WORKDIVISION` и `MKL_MIC_WORKDIVISION` са две реални числа, които определят каква част от работата се извършва от CPU процесорите и каква част от копроцесора. В тези експерименти се разглежда само работата на копроцесора.
- `MKL_MIC_THRESHOLDS_DGEMM` е минималният размер на задача, която ще се изпрати за решаване върху копроцесора. Използваме малки стойности, за да е сигурно, че експериментите се изпълняват върху копроцесора.

Таблица 2.2: Последователни и паралелни времена и ускорения за решаване на системата върху MIC копроцесорите

Пакет		MAGMA MIC		MKL	
Нишки	n	време [s]	ускорение	време [s]	ускорение
1	5 000	11.70	1.00	17.64	1.00
60	5 000	5.81	2.01	2.86	6.16
120	5 000	6.76	1.73	3.55	4.97
240	5 000	5.39	2.17	3.80	4.64
1	40 000	4896.49	1.00	2101.93	1.00
60	40 000	665.53	7.36	154.23	13.63
120	40 000	432.89	11.31	93.80	22.41
240	40 000	208.48	23.49	64.43	32.62

В Таблица 2.2 и Фигура 2.5 са представени резултати от числени експерименти за решаване на системите при $n = 5\ 000$ и $n = 40\ 000$, варирайки броя на нишките от 1 до 240.

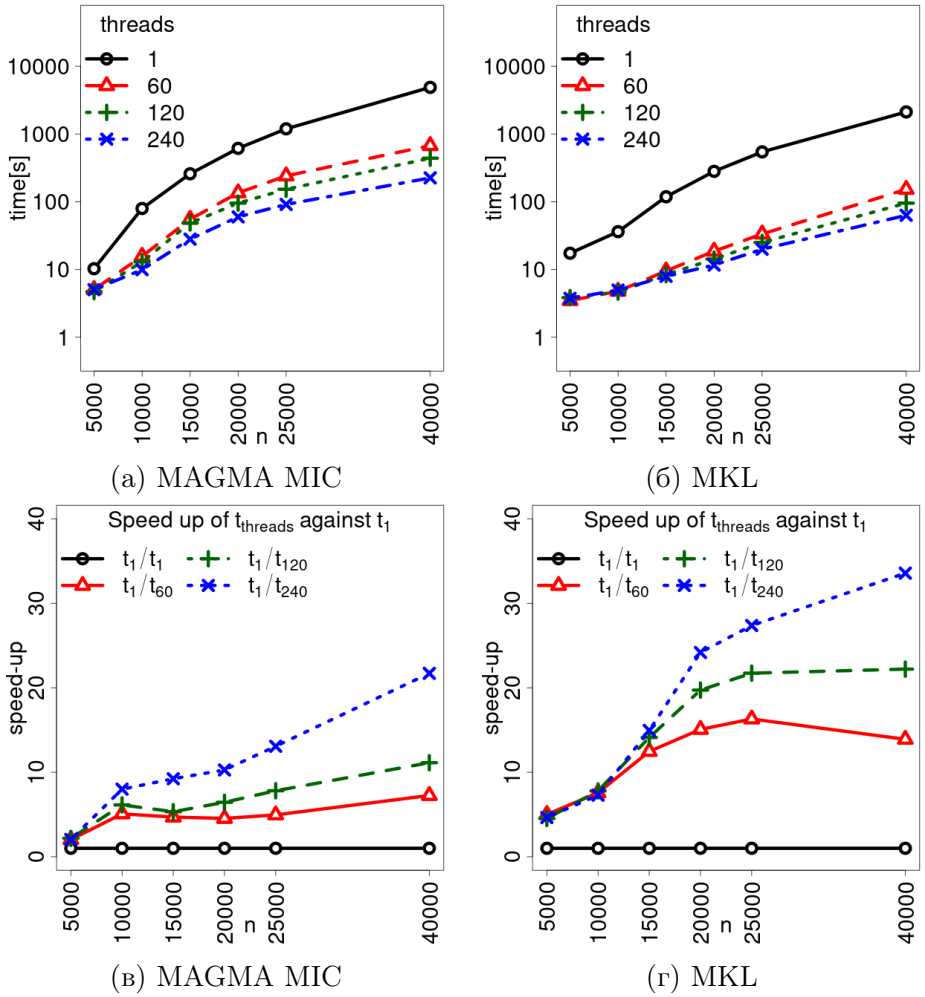
Резултатите показват много по-добра производителност на MKL в сравнение с MAGMA MIC. Това може да се дължи на по-добрата комуникация между нишките в MKL. За по-голямата задача постигнатото паралелно ускорение е 32.

На Фигура 2.6 е сравнена производителността на използваните софтуерни пакети за CPU и MIC архитектурата.

Производителността на PLASMA с MKL за решаване на системите върху CPU е по-добра от тази на MAGMA MIC. Това може да се дължи на по-добра комуникация между нишките.

В заключение отбелязваме, че библиотеката MKL има по-добра производителност, както на CPU, така и на MIC копроцесора. Най-доброто време върху копроцесора е почти 4 пъти по-добро от това на CPU процесора.

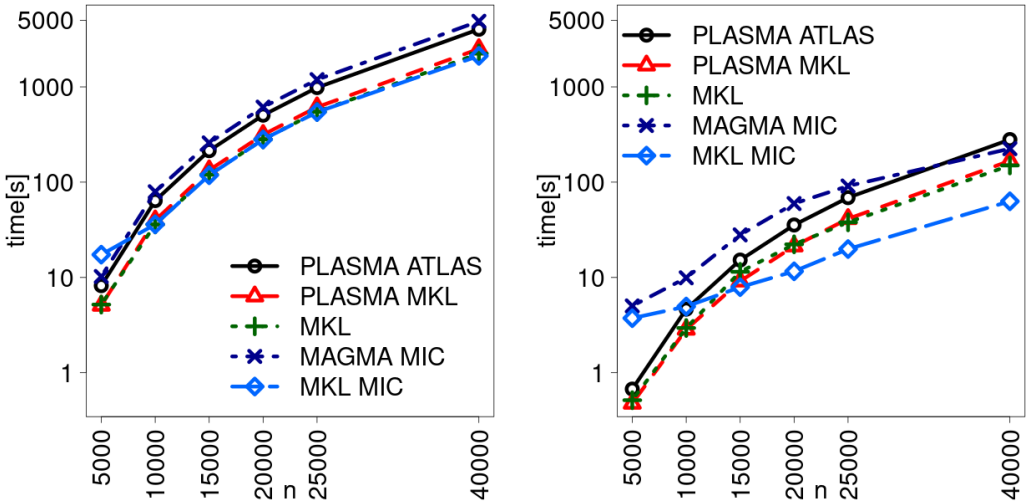
Преимуществата на библиотеката MKL определят рамката за сравнителен анализ на методите за решаване на изследваните системи с плътни матрици на базата LU факторизация и полусепарабелна йерархична факторизация. Така за оценка на производителността на софтуерния пакет STRUMPACK се използват времената получени с MKL.



(а) MAGMA MIC (б) MKL

(в) MAGMA MIC (г) MKL

Фигура 2.5: Сравнение на последователните и паралелните времена (а-б) и ускоренията (в-г) за решаване на системата върху MIC копроцесорите, с използване на софтуерните библиотеки MAGMA MIC и MKL

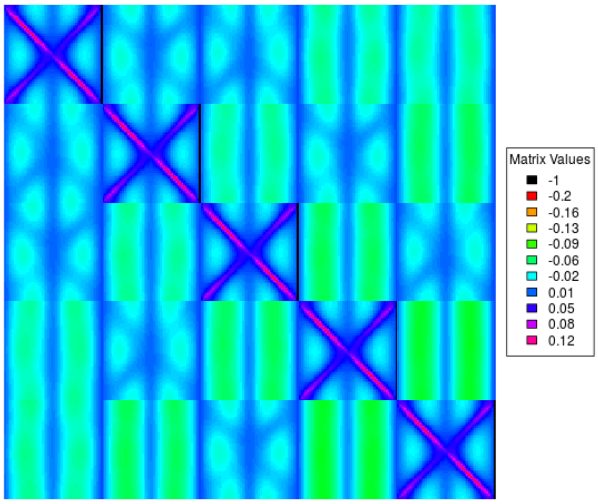


(а) Последователно (б) Паралелно с 16 нишки

Фигура 2.6: Сравнение на производителността на софтуерните библиотеки за CPU и MIC.

2.2.2 Йерархична полусепарабелна компресия

Йерархичната полусепарабелна компресия имплементирана в софтуерния пакет STRUMPACK е приблизителна. Това означава, че компресираната матрица H апроксимира изходящата матрица A. Потребителят задава два прага на грешката – абсолютен ε_{abs} и относителен ε_{rel} [37]. В представените в дисертацията експерименти абсолютният праг е фиксиран на $\varepsilon_{\text{abs}} = 10^{-8}$, като варираме относителния праг $\varepsilon_{\text{rel}} = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ и 10^{-12}. Ефективността на компресията зависи от *структурата* на изходящата матрица. Под *подходяща* се разбира такава *структура*, за която извъндиагоналните блокове имат нисък ранг. Структурата на матрицата D, породена от метода на граничните елементи при числено решаване на задачата за обтичане на профили на Жуковски, е визуализирана на Фигура 2.7.

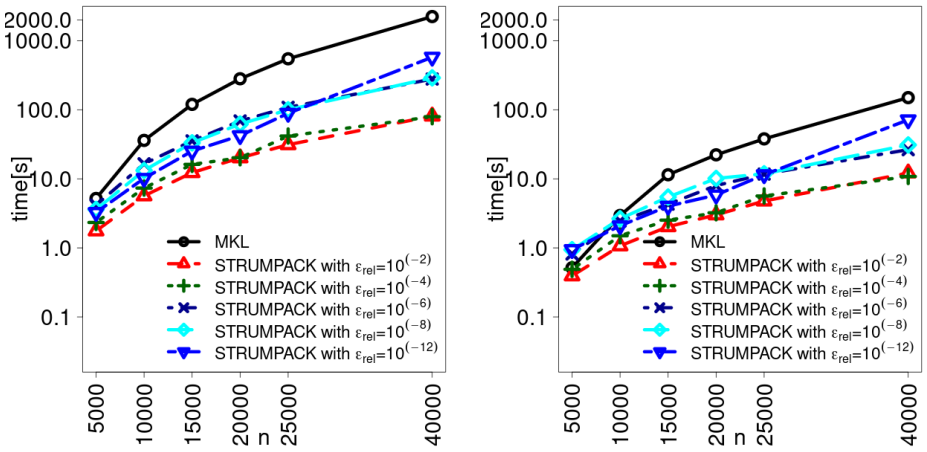


Фигура 2.7: Визуализация на матрицата D получена при прилагане на метода на граничните елементи за числено решаване на задачата за обтичане на профили на Жуковски

Сравнителен анализ на йерархична и LU факторизация върху CPU с обща памет

В този раздел анализираме производителността на метода на полусепарабелна йерархична (HSS) компресия и неговата софтуерна реализация STRUMPACK в сравнение с метода на Гаус, и неговата най-добра (на базата на анализа, изложен по-горе) реализация MKL, където алгоритъмът използва блочна LU факторизация. На Фигура 2.8 са представени последователните (Фигура 2.8а) и паралелните (Фигура 2.8б) времена за решаване на системата линейни уравнения. Резултатите потвърждават по-добрата изчислителна сложността на HSS компресията ($O(n^2 r)$) в сравнение с LU факторизацията ($O(n^3)$). Ясно се вижда и въздействието на избрания относителен праг на грешката ε_{rel}.

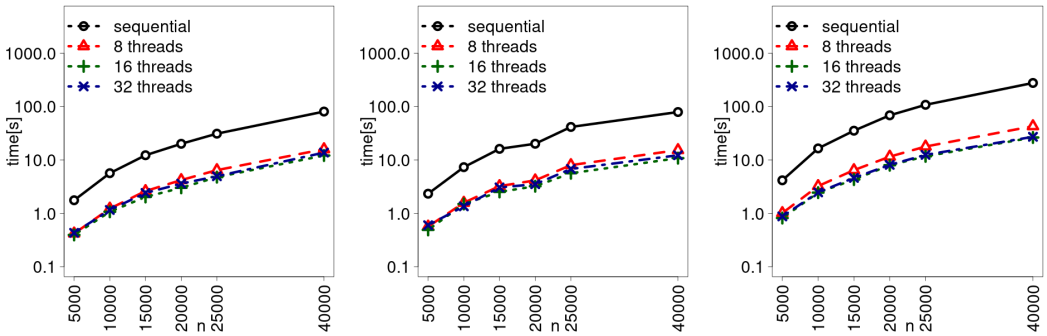
За последователните експерименти STRUMPACK е много по-ефективен от най-добрия пряк солвър, който използваме – MKL. На Фигура 2.9 са показани последователните и паралелни времена за изпълнение, а на Фигура 2.10 – паралелните ускорения на STRUMPACK. Най-голямо ускорение се получава при най-големия праг на грешката $\varepsilon_{\text{rel}} = 10^{-2}$. Това е пряк резултат от по-голямата компресия в този случай, водещ до по-малка стойност на ранга r. В сравнение с преките солвъри (Таблица 2.1) STRUMPACK показва по-малки



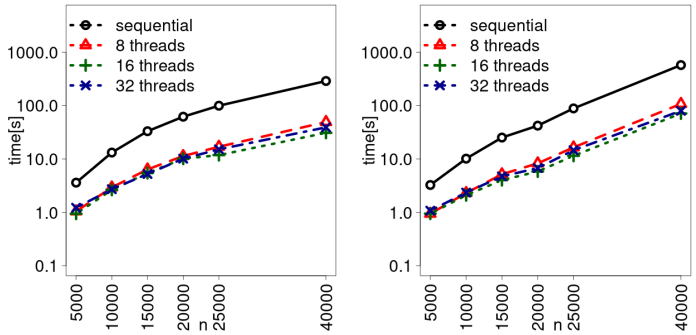
(а) Последователно (б) Паралелно с 16 нишки

Фигура 2.8: Производителност на STRUMPACK сравнена с MKL

паралелни ускорения, съответно от $\sim$2 до $\sim$5. Това се дължи на по-сложната рекурсивна структура на HSS компресията. Най-добри паралелни времена се получават в случая на 16 нишки, т.е. хипертредингът не води до допълнително паралелно ускорение. Аналогичен резултат получихме в предишната секция при всички тестове с преки Гаусови солвъри.

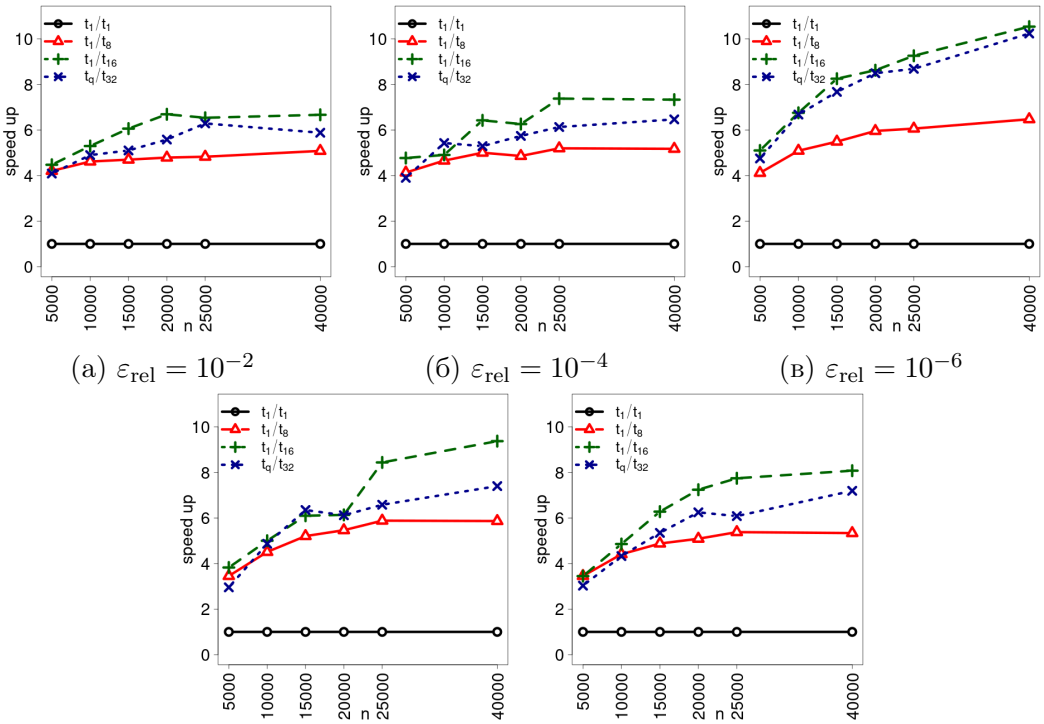


(а) $\varepsilon_{\text{rel}} = 10^{-2}$ (б) $\varepsilon_{\text{rel}} = 10^{-4}$ (в) $\varepsilon_{\text{rel}} = 10^{-6}$



(г) $\varepsilon_{\text{rel}} = 10^{-8}$ (д) $\varepsilon_{\text{rel}} = 10^{-12}$

Фигура 2.9: Последователни и паралелни времена



(а) $\varepsilon_{\text{rel}} = 10^{-2}$ (б) $\varepsilon_{\text{rel}} = 10^{-4}$ (в) $\varepsilon_{\text{rel}} = 10^{-6}$

(г) $\varepsilon_{\text{rel}} = 10^{-8}$ (д) $\varepsilon_{\text{rel}} = 10^{-12}$

Фигура 2.10: Паралелни ускорения

Анализ на грешката за HSS-базирания солвър

Нека припомним отново, че HSS компресията е приблизителна, т.е. компресираната матрица H е приближение на матрицата A. Полученото с помощта на HSS компресия решение на системата линейни алгебрични уравнения е апроксимация на точното. За оценка на грешката приемаме за референтно решението по метода на Гаус с пряк солвър на базата на LU факторизация (виж Раздел 2.2.1). Тук анализираме относителната грешка R_{relative} дефинирана както следва:

$$R_{\text{relative}} = \frac{\left\|x^{\text{Gauss}} - x^{\text{HSS}}\right\|_{l_2}}{\left\|x^{\text{Gauss}}\right\|_{l_2}} = \frac{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}} - x_i^{\text{HSS}})^2}}{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}})^2}}, \tag{2.5}$$

където x^{Gauss} е решението получено с гаусовият солвър от пакета MKL, което се използва като референтно, а x^{HSS} е решението получено от пакета STRUMPACK, при използване на йерархична полусепарабелна компресия.

В Таблица 2.3 са показани относителните грешки R_{relative}, варирайки размерността на задачата $n \in \{5\ 005, 10\ 005, 15\ 005, 20\ 005, 25\ 005, 40\ 005\}$, както и праговете на относителна грешка $\varepsilon_{\text{rel}} \in \{10^{-6}, 10^{-8}, 10^{-12}\}$.

Точността и изчислителната ефективност на йерархичният метод зависят от максималния ранг на извъндиагоналните блокове r, който се определя от избраните прагове на грешката и зависи от структурата на изходната матрица A. По-големият ранг r съответства на по-малка относителна грешка R_{relative}, както и на по-дълго време за решаване на системата.

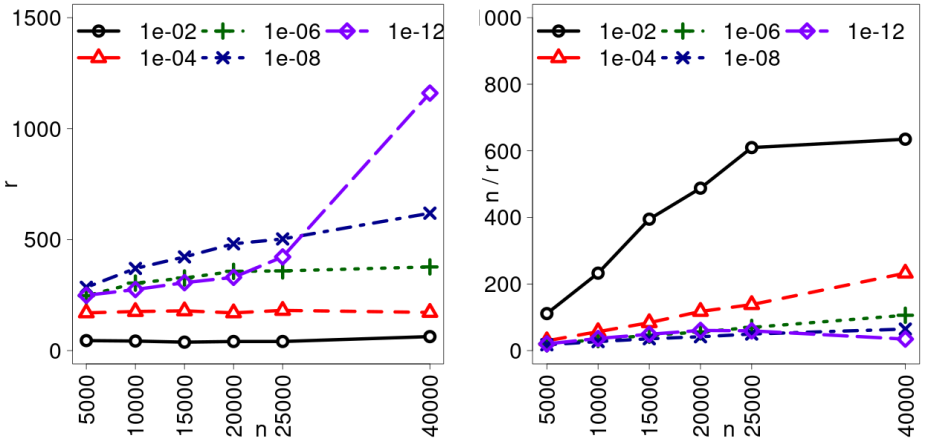
Таблица 2.3: Относителна грешка R_{relative}

n	$\varepsilon_{rel} = 10^{-6}$	$\varepsilon_{rel} = 10^{-8}$	$\varepsilon_{rel} = 10^{-12}$
5005	1.1	0.085	0.00019
10005	0.75	0.17	0.00075
15005	0.29	0.23	0.00097
20005	0.28	0.34	0.0038
25005	0.3	1.48	0.013
40005	0.37	1.59	0.027

Относителната грешка, както и ефективността на метода зависят от изчисления в процеса на компресията максимален ранг r на извъндиагоналните блокове. На Фигура 2.11a са показани ранговете изчислени за разгледаните размери на задачата и прагове $\varepsilon_{\text{relative}} \in \{10^{-6}, 10^{-8}, 10^{-12}\}$. Ще отбележим, че за разглежданата задача за обтичане на крилни профили, праговете 10^{-2} и

10^{-4} са неприложими на практика поради неприемливо големите относителни грешки. На Фигура 2.11б е показано отношението между размера на задачата n и максималния ранг r.

Анализът на представените резултати показва, че за постигане на висока точност на метода може да е необходим много малък праг съответстващ на голям ранг r. В такъв случай методът на HSS компресия може да не бъде достатъчно ефективен. Това означава, че структурата на матрицата A не е достатъчно подходяща.



(а) Ранг r (б) Отношение на r/n

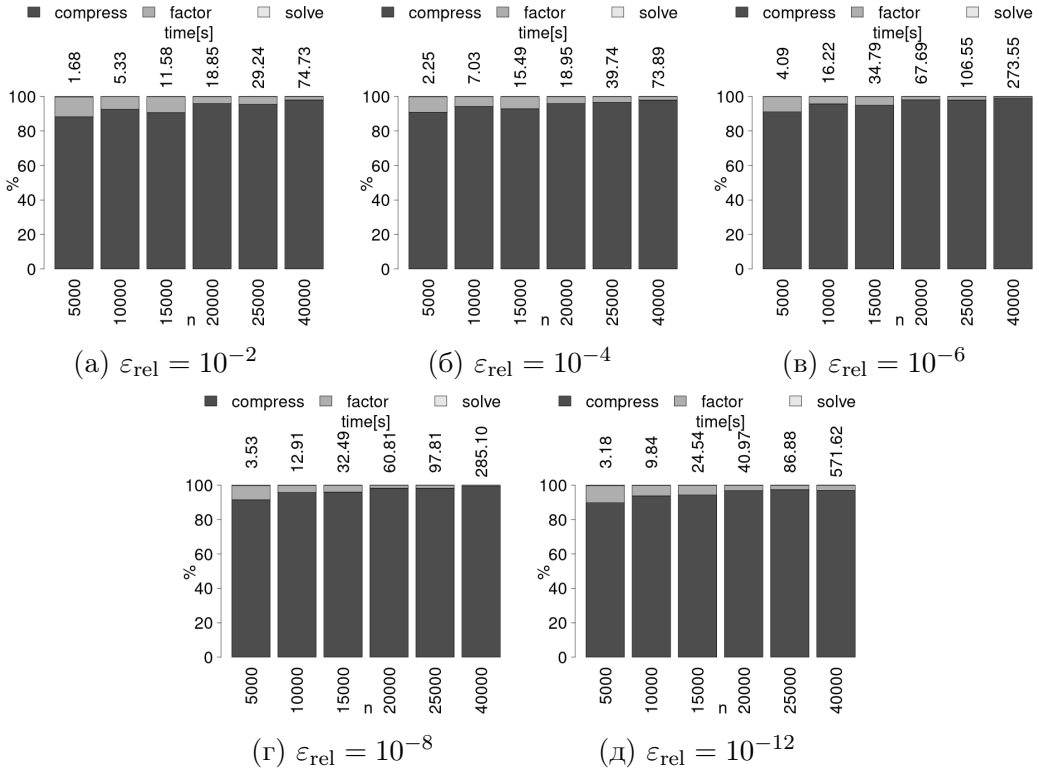
Фигура 2.11: Максимален извъндиагонален ранг r

Сравнителен анализ на времената за изпълнение на стъпките на HSS базирания солвър

Трите стъпки на HSS базирания солвър (HSS компресия, ULV-подобна факторизация и решаване на системата) имат различна изчислителна сложност. На Фигура 2.12 и Фигура 2.13 за всяка стъпка е показан процента от цялото време, съответно за последователно и паралелно изпълнение.

HSS компресията отнема най-голяма част от времето – от ∼90% до почти ∼100%. Това се дължи на по-високата изчислителна сложност на тази стъпка – $O(n^2 r)$. ULV-подобната факторизация (изчислителна сложност $O(nr^2)$) е на второ място – от ∼5% до ∼1%, докато решението на системата с факторизираната компресирана матрица (изчислителна сложност $O(nr)$) отнема незначително малка част от времето – под 1%.

Когато компресията е по-ефективна (тоест рангът r е по-малък), ULV-подобната факторизация отнема по-малка част от времето. Това се дължи на



(а) $\varepsilon_{\text{rel}} = 10^{-2}$ (б) $\varepsilon_{\text{rel}} = 10^{-4}$ (в) $\varepsilon_{\text{rel}} = 10^{-6}$

(г) $\varepsilon_{\text{rel}} = 10^{-8}$ (д) $\varepsilon_{\text{rel}} = 10^{-12}$

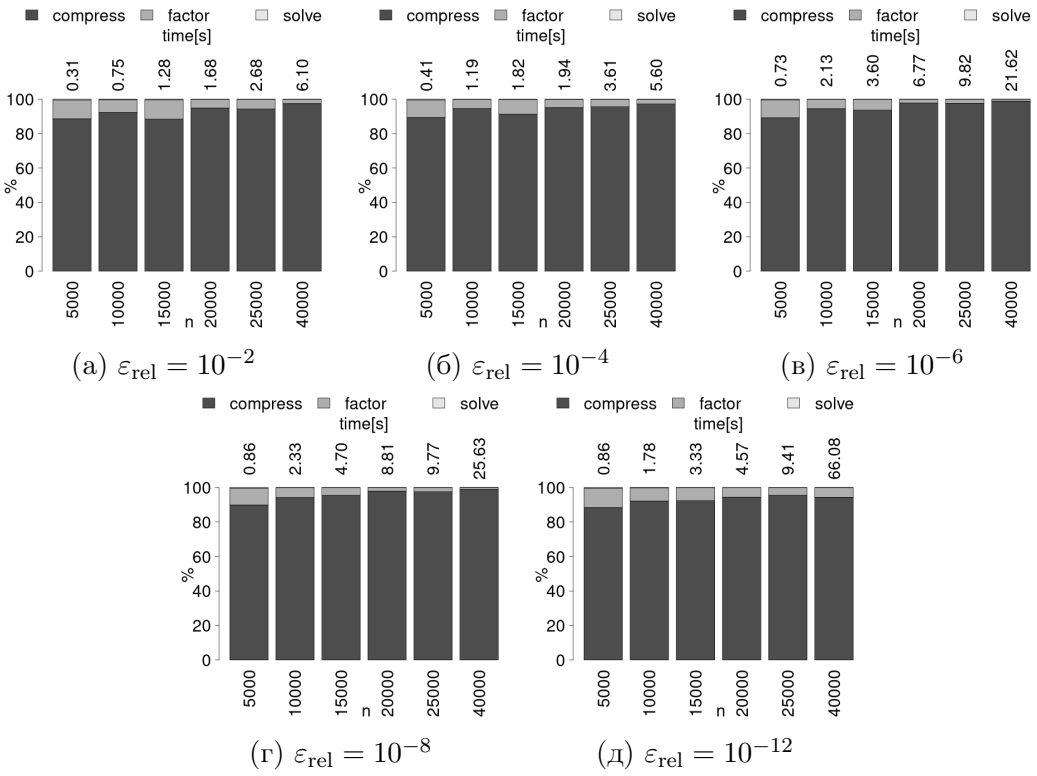
Фигура 2.12: Сравнение на времето за изпълнение на трите стъпки на HSS базирания солвър при последователно изпълнение

по-малкия брой на числата в компресираната матрица H и на съответното намаляване на броя на аритметичните операции при факториацията.

Следващият важен въпрос анализиран в този раздел е паралелната ефективност на отделните части от алгоритъма. На Фигура 2.14 са показани паралелните ускорения за трите стъпки при различните стойности на ε_{rel}. HSS компресията е с най-добро ускорение, достигащо до $\sim$10 или $\sim$12, в зависимост от зададеният праг на относителна грешка ε_{rel}.

ULV-подобната факторизация показва подобно ускорение за $\varepsilon_{\text{rel}} = 10^{-2}$, 10^{-4} и 10^{-6}. За $\varepsilon_{\text{rel}} = 10^{-8}$ и 10^{-12} паралелното ускорение намалява и е от $\sim$5 до $\sim$6. Това се дължи на по-ниското ниво на компресия за тези експерименти.

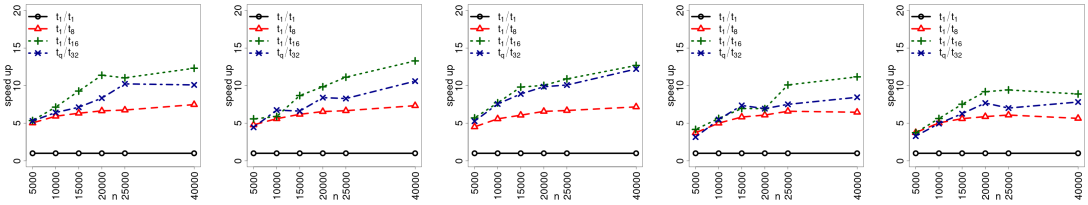
При решаване на системата с факторизираната компресирана матрица се наблюдава най-малко паралелно ускорение от $\sim$2 до $\sim$5. Това обаче няма съществено значение за паралелната ефективност на метода поради факта, че тази стъпка отнема много по-малко време от другите две.



(а) $\varepsilon_{\text{rel}} = 10^{-2}$ (б) $\varepsilon_{\text{rel}} = 10^{-4}$ (в) $\varepsilon_{\text{rel}} = 10^{-6}$

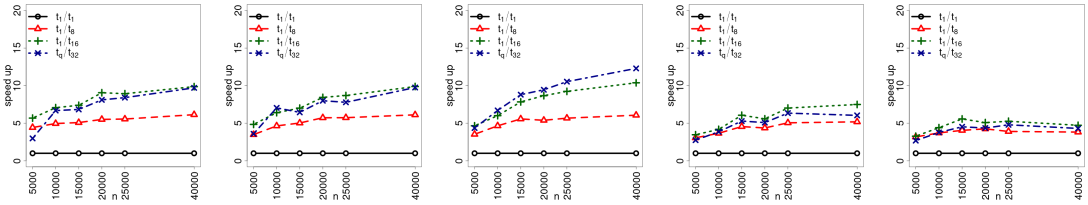
(г) $\varepsilon_{\text{rel}} = 10^{-8}$ (д) $\varepsilon_{\text{rel}} = 10^{-12}$

Фигура 2.13: Сравнение на времето за изпълнение на трите стъпки на HSS базирания солвър при паралелно (16 нишки) изпълнение



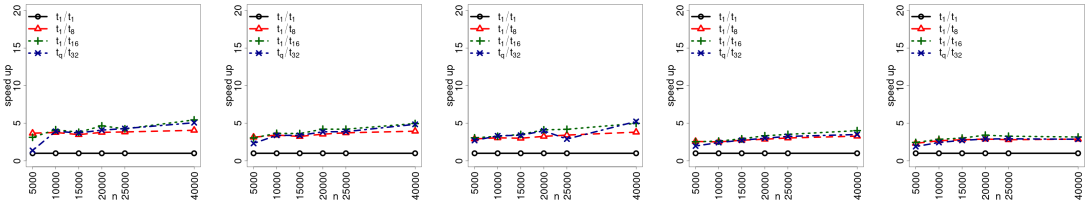
$\varepsilon_{\text{rel}} = 10^{-2}$ $\varepsilon_{\text{rel}} = 10^{-4}$ $\varepsilon_{\text{rel}} = 10^{-6}$ $\varepsilon_{\text{rel}} = 10^{-8}$ $\varepsilon_{\text{rel}} = 10^{-12}$

(а) HSS компресия



$\varepsilon_{\text{rel}} = 10^{-2}$ $\varepsilon_{\text{rel}} = 10^{-4}$ $\varepsilon_{\text{rel}} = 10^{-6}$ $\varepsilon_{\text{rel}} = 10^{-8}$ $\varepsilon_{\text{rel}} = 10^{-12}$

(б) ULV-подобна факторизация



$\varepsilon_{\text{rel}} = 10^{-2}$ $\varepsilon_{\text{rel}} = 10^{-4}$ $\varepsilon_{\text{rel}} = 10^{-6}$ $\varepsilon_{\text{rel}} = 10^{-8}$ $\varepsilon_{\text{rel}} = 10^{-12}$

(в) Решаване на системата с факторизираната компресирана матрица

Фигура 2.14: Паралелно ускорение на отделните стъпки на йерархичния солвър от пакета STRUMPACK: а)HSS компресия; б)ULV-подобна факторизация; в) решаване на системата с факторизираната компресирана матрица

2.3 Паралелна скалируемост върху компютърни системи с разпределена памет

Този раздел е посветен на анализ на някои специфични особености и трудности при работа с изчислителни системи с хибридна архитектура. Те имат разпределена памет на нивото на сървърите, от които са изградени и обща памет в рамките на всеки сървър. Представените резултати са публикувани в [74]. Анализираните числени експерименти са проведени върху сървърна система разположена в ИИКТ-БАН, която е част от инфраструктурата на ЦВП по Информатика и ИКТ. Използвани са до два сървъра, всеки с по 24 процесорни ядра. Изпълнението на паралелни програми върху такива системи с разпределена памет изисква изпращане на съобщения между сървърите. За целта е използван стандарта Message Passing Interface (MPI). Сървърите са с обща памет, т.е. в рамките на всеки от тях можем да приложим OpenMP. Както и в предишния раздел, сравнителният анализ включва паралелните библиотеки MKL и STRUMPACK. За решаване на системата от линейните алгебрични уравнения, получена при дискретизация на задачата за обтичане на профили на Жуковски са използвани един или два сървъра, свързани с етернет. Анализирани са следните варианти на изпълнение на MKL и STRUMPACK:

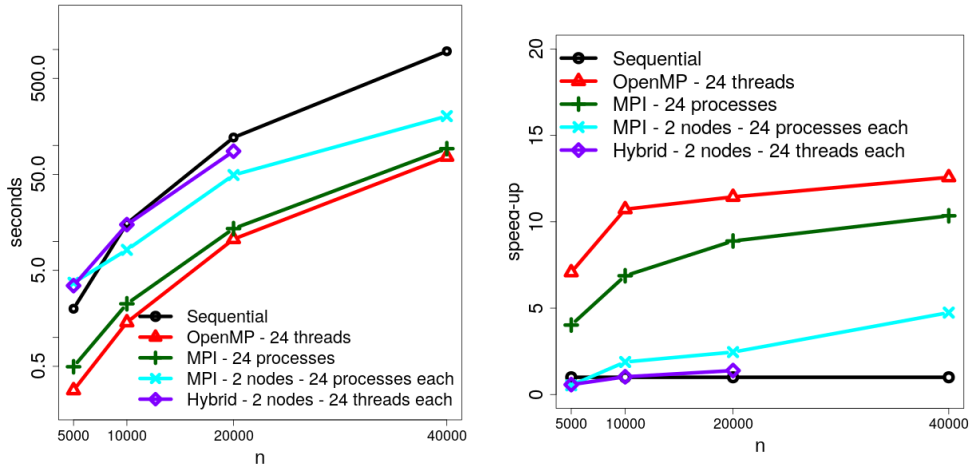
- Последователно;
- Паралелизация с 24 OpenMP нишки;
- Паралелизация с 24 MPI процеса на 1 сървър;
- Паралелизация с 48 MPI процеса на 2 сървъра;
- Хибридна паралелизация с 2 MPI процеса на 2 сървъра, всеки с 24 OpenMP нишки.

В представените експерименти са използвани следните софтуерни пакети и библиотеки: MKL, Intel MPI 2019.5.281, STRUMPACK 3.2.0, METIS 5.1.0, ParMETIS 4.0.3 и SCOTCH 6.0.8. Всички библиотеки (с изключение на тези от Intel), както и разработеният код са компилирани с gcc 9.2.0.

2.3.1 LU факторизация

На Фигура 2.15 са показани времената за решение на системата от линейни алгебрични уравнения с MKL. Най-добро време се получава с OpenMP при използване на един сървър. Паралелното ускорение на един сървър е най-добро за OpenMP, следвано от MPI (Фигура 2.15б). Това се обяснява с относително

по-бавните етернет комуникации между процесите при използване на повече от един сървър.



(а) MKL времена (б) MKL паралелни ускорения

Фигура 2.15: Паралелни времена и ускорения за решаване на системите с MKL.

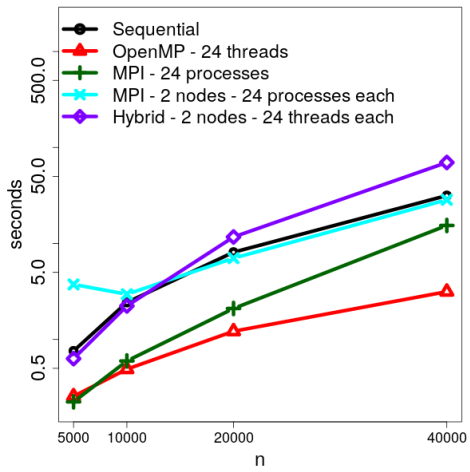
2.3.2 HSS компресия

При решаване на системата с използвана HSS компресия се получават по-малки паралелни ускорения отколкото при прекия Гаусов солвър (виж Фигура 2.16, 2.17, 2.18 и 2.19). Това може да се обясни с рекурсивната структура на HSS компресията.

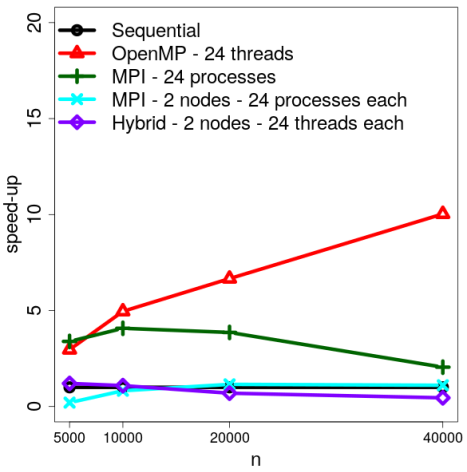
При числените експерименти с най-ниския праг на относителна грешка ($\varepsilon_{\text{rel}} = 10^{-8}$) се получава най-добро време за изпълнение при използване на MPI върху един сървър. Това най-вероятно се дължи на факта, че OpenMP паралелизацията в STRUMPACK е направена по-късно в разработката на пакета от MPI.

Поведението на паралелното ускорение се променя съществено при използване на два сървъра. Това се дължи на (относително) бавната връзка между тях – 1000 Mb Ethernet. Можем да направим извода, че ефективността може значително да се подобри при по-бърза комуникационна среда (например InfiniBand), както и при увеличаване на размера на решаваните системи линейни алгебрични уравнения.

В Таблица 2.4 са показани времената за изпълнение на отделните части на метода на HSS компресия реализиран в пакета STRUMPACK за $n = 40\ 000$.

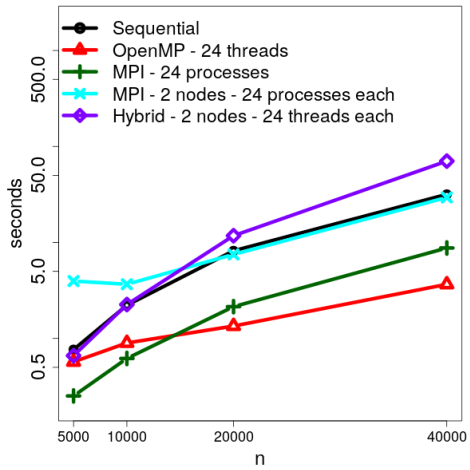


(а) STRUMPACK времена при $\varepsilon_{rel} = 10^{-2}$

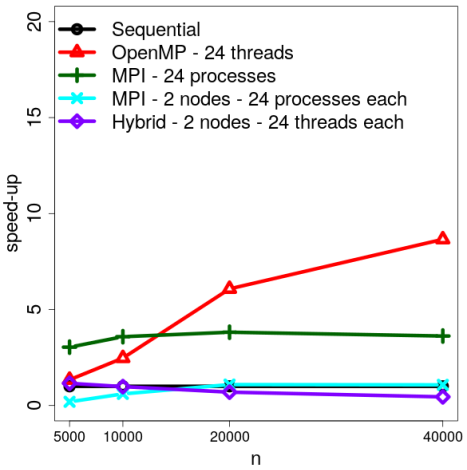


(б) STRUMPACK ускорения при $\varepsilon_{rel} = 10^{-2}$

Фигура 2.16: Времена и паралелни ускорения при $\varepsilon_{rel} = 10^{-2}$.



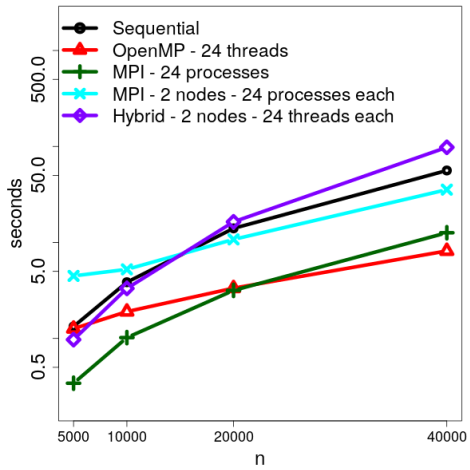
(а) STRUMPACK времена при $\varepsilon_{rel} = 10^{-4}$



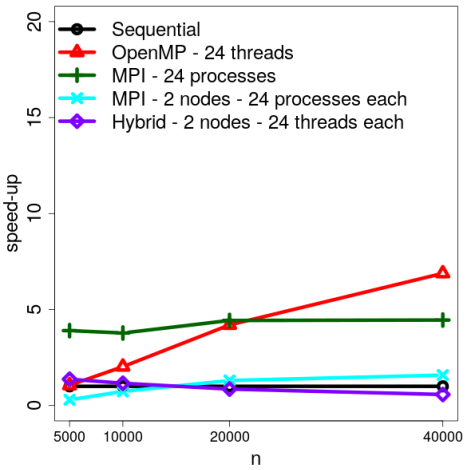
(б) STRUMPACK ускорения при $\varepsilon_{rel} = 10^{-4}$

Фигура 2.17: Времена и паралелни ускорения при $\varepsilon_{rel} = 10^{-4}$.

И тук, както и в случа на системи с обща памет, времето за компресията е силно доминиращо.

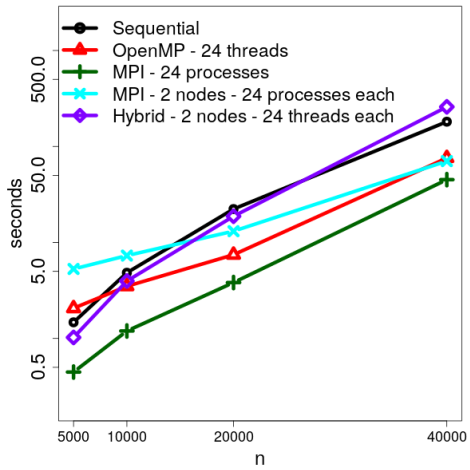


(а) STRUMPACK времена при $\varepsilon_{rel} = 10^{-6}$

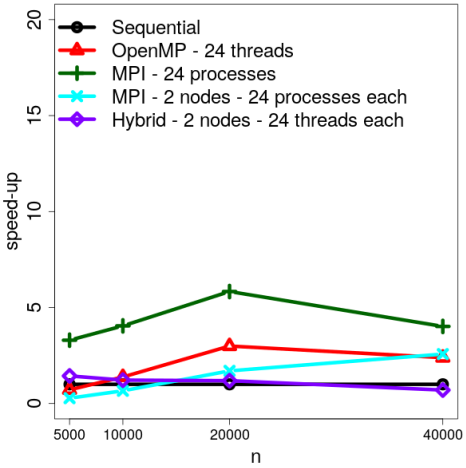


(б) STRUMPACK ускорения при $\varepsilon_{rel} = 10^{-6}$

Фигура 2.18: Времена и паралелни ускорения при $\varepsilon_{rel} = 10^{-6}$.



(а) STRUMPACK времена при $\varepsilon_{rel} = 10^{-8}$



(б) STRUMPACK ускорения при $\varepsilon_{rel} = 10^{-8}$

Фигура 2.19: Времена и паралелни ускорения при $\varepsilon_{rel} = 10^{-8}$.

Таблица 2.4: Времена за изпълнение на отделните части на метода на HSS компресия реализиран в пакета STRUMPACK при $n = 40\ 000$

Паралелизация	$\varepsilon_{\text{rel}} = 10^{-2}$				$\varepsilon_{\text{rel}} = 10^{-4}$			
	HSS	ULV	Решение	Общо	HSS	ULV	Решение	Общо
Последователно	31.39	0.15	0.01	31.55	31.52	0.17	0.011	31.7
OpenMP - 24 нишки	3.07	0.07	0.005	3.15	3.56	0.099	0.0053	3.66
MPI - 1 сървър	15.36	0.031	0.006	15.4	8.72	0.031	0.0061	8.76
MPI - 2 сървъра	28.36	0.078	0.047	28.5	29.18	0.18	0.074	29.45
Хибридно	68.92	0.068	0.009	69.0	70.32	0.076	0.01	70.4

Паралелизация	$\varepsilon_{\text{rel}} = 10^{-6}$				$\varepsilon_{\text{rel}} = 10^{-8}$			
	HSS	ULV	Решение	Общо	HSS	ULV	Решение	Общо
Последователно	55.97	0.21	0.015	56.2	177.58	3.5	0.087	181.17
OpenMP - 24 нишки	7.91	0.24	0.0065	8.16	71.06	4.5	0.2	75.77
MPI - 1 сървър	12.56	0.049	0.0086	12.62	44.44	0.61	0.056	45.11
MPI - 2 сървъра	34.99	0.37	0.12	35.48	68.68	1.42	0.28	70.39
Хибридно	97.9	0.095	0.013	98.0	257.7	3.05	0.11	260.86

2.4 Заключителни бележки

В тази глава са анализирани методи за решаване на системи от линейни алгебрични уравнения с плътни матрици, които се получават при дискретизация по метода на граничните елементи на интегралното уравнение (2.2), което описва двумерно ламинарно течение около крилни профили на Жуковски. Общият извод е, че разгледаните блочни методи показват добро паралелно бързодействие и ускорения.

Централно място в представените резултати заема изследването на метода на йерархична полусепарабелна компресия (HSS компресия). Експерименталният сравнителен анализ е на базата на неговата реализация в софтуерния пакет STRUMPACK. Той показва по-добро бързодействие от преките Гаусови солвъри използващи блочна LU факторизация. В същото време получените паралелни ускорения със STRUMPACK са по-малки, което се обуславя от по-сложната йерархична структура на алгоритъма.

Точността и изчислителната ефективност на HSS компресията зависят от праговете на относителна и абсолютна грешка. Това са параметри, които се избират от потребителя. Представеният анализ показва как да получим най-добра ефективност при зададена точност.

Структурата на матрицата е определяща за качеството на йерархичната компресия. За разгледаната задачата структурата на матрицата е *подходяща*, т.е. позволява ефективното прилагане на HSS компресия. При други задачи обаче такава *структура* може да не съществува. В Глава 3 е изследвана ролята на пренареждане на неизвестните за подобряване на структурата матрицата.

Глава 3

Метод на крайните елементи за числено решаване на двумерна стационарна задача за дробна дифузия

Дробният оператор на Лаплас на степен α, който се нарича също така дробен лапласиан, се означава с $(-\Delta)^\alpha$. В теорията на стохастичните процеси дробният лапласиан се определя като безкрайно малък (infinitesimal) генератор на устойчиви процеси на Леви [9, 80].

Дробните елиптични оператори по пространството на степен $\alpha \in (0, 1)$ описват процеси на *аномална* дифузия. Свързаните с тях гранични задачи са нелокални и, в общия случай, численото решение на такива задачи е изчислително скъп процес. *Аномалната* дифузия се среща често в природата [47, 58]. Такъв тип нелокални модели се прилага например в обработката на изображения [14, 30, 35, 54], финансовата математика [16, 21], електромагнитостатиката [27], перидинамиката [67], моделирането на течения в порести среди [36, 22] и много други.

В тази глава ще разгледаме стационарната гранична задача за *дробна* дифузия. Представените числени експерименти са за моделна задача в квадратна и кръгла област. Дробният лапласиан се дефинира с помощта на потенциал на Риц (Riesz). Теоретичната постановка на задачата и разработеният специализиран метод на крайните елементи за нейното числено решаване са представени в статията на Акоста и съавтори [3]. В [4] авторите описват алгоритмичната реализация на метода. В същата статия е включен и код на MatLab за моделната двумерна задача в квадратна и кръгла област, който

е използван за генериране на изследваните в тази глава системи от линейни алгебрични уравнения с плътни матрици.

Още в увода отбелязахме, че за такива системи методите от тип гаусова елиминация имат изчислителна сложност $O(n^3)$. Целта на изследванията в дисертацията е подобряване на изчислителната ефективност на солвъри за системи с плътни матрици за разглежданите класове задачи. В Глава 2 бяха представени резултати за системи получени при дискретизация по метода на граничните елементи. Тук прилагаме метода на крайните елементи за задачата с дробна дифузия, който води до друг тип нелокални интегрални уравнения. И отново, като алтернатива на метода на Гаус, изследваме ефективността на йерархичните методи използващи HSS компресия.

В предходната Глава 2 бяха анализирани няколко софтуерни пакета реализиращи методи от тип гаусова елиминация. В тази глава, за нуждите на сравнителния анализ, е използван само най-ефективният от тях: Intel's Math kernel Library (MKL). Анализирана е производителността на алгоритъма базиран на йерархична полусепарабелна компресия (HSS), който е реализиран в пакета STRUctured Matrix PACKage (STRUMPACK). Изследвани са няколко метода за пренареждане на неизвестните с цел подобряване ефективността на HSS компресията.

Основните резултати представени в тази глава са публикувани в следните статии:

[70] D. Slavchev. On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2

[71] D. Slavchev. Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5

[76] D. Slavchev, S. Margenov, and I.G. Georgiev. On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using hss compression. In *AIP Conference Proceedings*, volume 2302, page 120008, 2020. doi: 10.1063/5.0034506

3.1 Постановка на задачата

Нека най-напред разглеждаме цялото евклидово пространство $\mathbb{R}^d$. Тук псевдодиференциалният оператор със символ $|\xi|^{2\alpha}$ за функции $u(x)$, $x \in \mathbb{R}^d$ от

класа на Шварц $\mathcal{S}$ се дефинира във вида

$$(-\Delta)^{\alpha} u = \mathcal{F}\left(|\xi|^{2\alpha} \mathcal{F} u\right), \tag{3.1}$$

$\alpha \in (0,1)$, където $\mathcal{F}$ е трансформацията на Фурие. Този оператор клони към недробния лапласиан при $\alpha \to 1$ и към единица при $\alpha \to 0$.

Така дробният лапласиан може да се представи във вида

$$(-\Delta)^{\alpha} u(x) = C(d,\alpha)\ \text{P.V.} \int_{\mathbb{R}^n} \frac{u(x)-u(y)}{|x-y|^{d+2\alpha}}, \tag{3.2}$$

където P.V. означава главна стойност, а $C(d,\alpha)$ е нормализиращата константа

$$C(d,\alpha) = \frac{2^{2\alpha}\alpha\Gamma\left(\alpha+\frac{d}{2}\right)}{\pi^{d/2}\Gamma(1-\alpha)},$$

необходима за съгласуваност с (3.1).

Съществуват различни дефиниции на дробен лапласиан в отворена ограничена област $\Omega \subset \mathbb{R}^d$. Така например:

- *Спектралният* дробен лапласиан се дефинира на базата на факторизация във вида $-\Delta = W^T D W$, като $(-\Delta)^{\alpha}_S = W^T D^{\alpha} W$. В този случай дробната степен действа върху диагоналната матрица D от собствените стойности на оператора на Лаплас в Ω. В обзорната статия [42] са анализирани числени методи за решаване на гранични задачи с дробна дифузия от този тип.

- Действието на *интегралния* дробен лапласиан $(-\Delta)^{\alpha}_I u$ се определя, като ресекция на интегралното представяне (3.2) върху Ω. Така полученият нелокален оператор не е еквивалентен на спектралния дробен лапласиан [61]. В [43] са разгледани разликите между двете дефиниции, като в частност са анализирани разликите в асимптотите на граничния слой.

- Трета възможност е дефиницията на дробен лапласиан $(-\Delta)^{s}_R$, при която в (3.2) интегрирането се ограничава върху Ω (вместо $\mathbb{R}^d$). Този оператор е известен като безкрайно малък генератор на *цензурирани* устойчиви процеси на Леви (Lévi) [11].

В тази глава е използвана втората дефиниция на дробен лапласиан, което е в съответствие с възприетата постановка на задачата в статията на Акоста [4]. За простота на записа ще игнорираме долния индекс I. Така разглеждаме следната гранична задача за *дробния* оператор на Лаплас

$$\begin{cases} (-\Delta)^{\alpha} u(x) = f(x), & x \in \Omega \\ u(x) = 0, & x \in \Omega^c. \end{cases} \tag{3.3}$$

Тук $\Omega \subset \mathbb{R}^d$ е ограничена отворена област, Ω^c е допълнението на Ω в $\mathbb{R}^d$ и $f(x)\, x \in \Omega$, е дясна част с достатъчна гладкост.

3.1.1 Функционални пространства и регулярност на решенията

Дробното пространство на Соболев (Sobolev) $H^\alpha(\Omega)$ се дефинира като

$$H^\alpha(\Omega) = \left\{ v \in L^2(\Omega) : |v|_{H^\alpha(\Omega)} < \infty \right\},$$

тук $|\cdot|_{H^\alpha(\Omega)}$ е полунормата на Ароншайн-Слободецкий (Aronszajn - Slobodeckij)

$$|v|^2_{H^\alpha(\Omega)} = \iint_{\Omega^2} \frac{|v(x) - v(y)|^2}{|x-y|^{d+2\alpha}} dxdy.$$

Известно е, че $H^\alpha(\Omega)$ е хилбертово пространство с норма $\|\cdot\|_{H^\alpha(\Omega)} = \|\cdot\|_{L^2(\Omega)} + |\cdot|_{H^\alpha}(\Omega)$. Скаларното произведение в $H^\alpha(\Omega)$ се определя от билинейната форма $\langle \cdot, \cdot \rangle_{H^\alpha}$,

$$\langle u, v \rangle_{H^\alpha(\Omega)} = \iint_{\Omega^2} \frac{(u(x) - u(y))(v(x) - v(y))}{|x-y|^{d+2\alpha}} dxdy. \tag{3.4}$$

Ще разгледаме също така пространството от функции с носител в Ω, дефинирано с равенството

$$\tilde{H}^\alpha(\Omega) = \left\{ v \in H^\alpha(\mathbb{R}^d) : \text{supp } v \subset \overline{\Omega} \right\}$$

Известно е, че билинейната форма $\langle \cdot, \cdot \rangle_{H^\alpha(\mathbb{R}^d)}$ индуцира норма в $\tilde{H}^\alpha(\Omega)$, като е в сила следното твърдение.

Твърдение 3.1 (Неравенство на Поинкаре)**.** *Съществува константа $c = c(\Omega, d, \alpha)$, такава че*

$$\|v\|_{l^2(\Omega)} \leq c|v|_{H^\alpha(\mathbb{R})}, \quad \forall v \in \tilde{H}^\alpha(\Omega).$$

По аналогичен начин се дефинират дробни пространства на Соболев от по-висок ред

$$H^{k+\alpha}(\Omega) = \left\{ v \in H^k(\Omega) : |D^\beta| \in H^\alpha(\Omega), \quad \forall \beta : |\beta| = k \right\},$$

както и съответната норма

$$\|v\|_{H^{k+\alpha}(\Omega)} = \|v\|_{H^k(\Omega)} + \sum_{|\beta|=k} \left|D^\beta v\right|_{H^\alpha(\Omega)}.$$

Вариационната формулировка на (3.3) се получава, като уравнението се умножи с тестова функция и се интегрира по части. Така за слабото решение получаваме уравнението : търси се $u \in H^\alpha(\Omega)$, такова че

$$\frac{C(d,\alpha)}{2} \langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \int_\Omega fv, \quad v \in \tilde{H}^\alpha(\Omega). \tag{3.5}$$

Скаларното произведение на u и v може да се запише във вида

$$\langle u, v \rangle_{H^\alpha \mathbb{R}^d} = \iint_{\mathbb{R}^d \times \mathbb{R}^d} \frac{(u(x) - u(y))\,(v(x) - v(y))}{|x - y|^{d+2\alpha}} dxdy.$$

Ще обърнем внимание на факта, че интегрирането е върху цялото пространство $\mathbb{R}^d$.

Коректността на вариационната постановка на задачата (3.5), както и съществуването и единствеността на решение в $\tilde{H}^\alpha(\Omega)$ следват от лемата на Лакс-Милграм (Lax-Milgram). При определени условия за гладкост на границата $\partial\Omega$ е в сила следващата теорема за регулярност [38, 81].

Теорема 3.1. *Нека $u \in \tilde{H}^\alpha(\Omega)$ е решението на (3.5). Тогава*

$$u \in \begin{cases} H^{2\alpha+r}(\Omega), & \text{за } \alpha + r < 1/2, \\ H^{s+1/2-\epsilon}(\Omega), \quad \forall \epsilon > 0, & \text{за } \alpha + r \geq 1/2. \end{cases}$$

3.1.2 Постановка на метода на крайните елементи

Нека $\mathcal{T}$ е допустима триангулация на областта Ω съставена от $N_\mathcal{T}$ триъгълни крайни елементи. В МКЕ триангулацията се нарича още мрежа. Разглеждаме крайноелементното пространство $\mathbb{V}_h$ от непрекъснати на части линейни функции върху $\mathcal{T}$. Нека $\{\varphi_1, \ldots, \varphi_N\} \subset \mathbb{V}_h$ е лагранжев възлов базис съответстващ на върховете на триъгълниците от $N_\mathcal{T}$ означени с $x_1, \ldots, x_N$, които не са на границата $\partial\Omega$. Тогава $\varphi_i(x_j) = \delta_i^j$. Нека $T \in \mathcal{T}$ е даден елемент от триангулацията и нека означим с h_T и ρ_T съответно диаметъра и радиуса на вписаната в T окръжност, като $h = \max_{T \in \mathcal{T}} h_T$. Разглеждаме регулярни по форма триангулации, за които съществува $\tau > 0$ независещо от $\mathcal{T}$, такова че

$$h_T \leq \tau \rho_T, \quad \forall T \in \mathcal{T}.$$

При тези предположения за всяко $\alpha \in (0, 1)$ дискретният аналог на вариационната задача (3.5) има вида

$$\frac{C(d, \alpha)}{2} \langle u_h, v_h \rangle_{H^\alpha(\mathbb{R}^n)} = \int_\Omega f v_h, \quad v_h \in \mathbb{V}_h. \tag{3.6}$$

Тук с $u_h = \sum_j u_j, \varphi_j$ е означено численото решение по метода на крайните елементи (МКЕ) на стационарната задача за дробна дифузия (3.3). Решаването на дискретната вариационна задача (3.6) се свежда до система от линейни алгебрични уравнения във вида

$$KU = F, \tag{3.7}$$

където $U = (u_j) \in \mathbb{R}^N$ са неизвестните възлови стойности. За матрицата на коравина K и дясната част F са в сила формулите

$$K_{ij} = \frac{C(d,\alpha)}{2} \langle \varphi_i, \varphi_j \rangle_{H^\alpha(\mathbb{R}^d)}, \qquad K = (K_{ij}) \in \mathbb{R}^{N\times N},$$
$$f_j = \int_\Omega f\varphi_j, \qquad F = (f_j) \in \mathbb{R}^N.$$

Матрицата на коравина K е симетрична и положително определена, и следователно (3.7) има единствено решение.

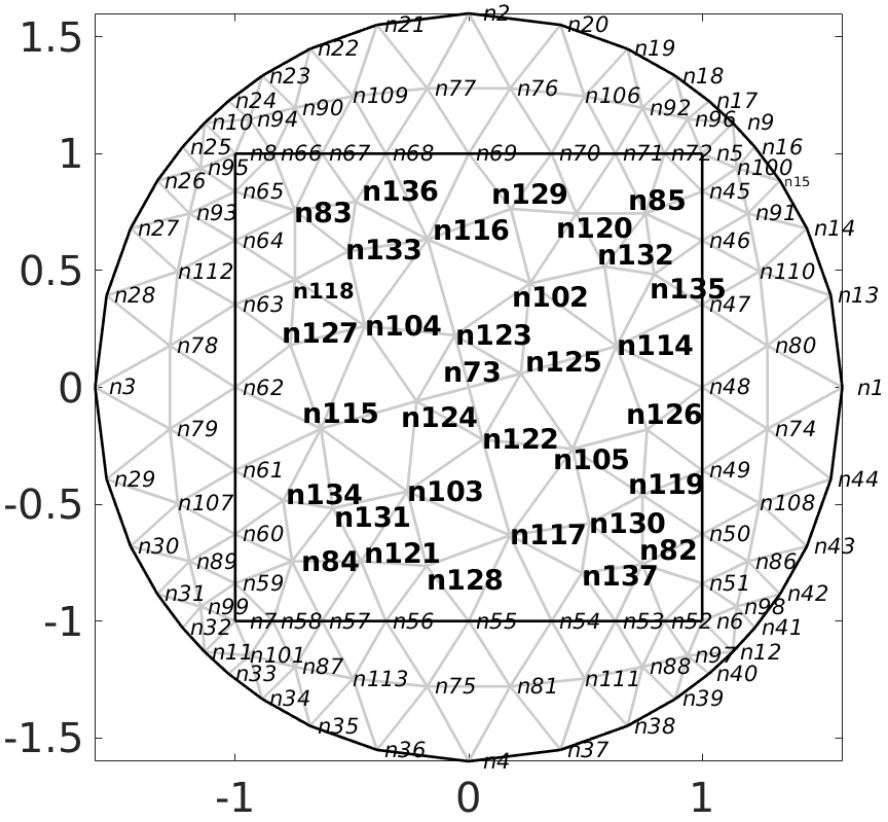
Нека напомним, че интегралите в скаларното произведение $\langle \varphi_i, \varphi_j \rangle_{H^\alpha(\mathbb{R}^d)}$ са върху цялото пространство $\mathbb{R}^d$. На практика това не е възможно. В приложения в [3] вариант на МКЕ, интегрирането се редуцира до кръгова област $B \supset \Omega$, такава че разстоянието между границата $\overline{\Omega}$ и допълнението B^c е достатъчно голямо. Получено е достатъчно условие (оценка отдолу) за радиуса на B, при което са в сила съответните оценки за грешката на МКЕ. Добавя се допълнителната триангулация $\tilde{\mathcal{T}}_A$ върху $B\backslash\Omega$, такава че общата триангулация $\tilde{\mathcal{T}} = \mathcal{T} \cup \mathcal{T}_A$ върху B е допустима. На Фигура 3.1 е показан пример на такава триангулация за квадратна област Ω. Възлите отбелязани с удебелен шрифт съответстват на неизвестните U.

Нека означим с $M_{\tilde{\mathcal{T}}}$ броя на елементите в триангулацията върху кръговата област B. Тогава елементите на матрицата на коравина K могат да се запишат във вида

$$Kij = \frac{C(d,\alpha)}{2} \sum_{\ell=1}^{N_{\tilde{\mathcal{T}}}} \left(\sum_{m=1}^{N_{\tilde{\mathcal{T}}}} I_{\ell,m}^{i,j} + 2J_\ell^{i,j} \right), \quad \ell, m \in [1, \ldots, N_{\tilde{\mathcal{T}}}], \tag{3.8}$$

където за въведените интеграли I и J са в сила формулите

$$I_{\ell,m}^{i,j} = \int_{T_\ell}\int_{T_m} \frac{(\varphi(x) - \varphi_i(y))\,(\varphi_j(x) - \varphi_j(y))}{|x-y|^{2+2\alpha}} dxdy \tag{3.9}$$
$$J_\ell^{i,j} = \int_{T_\ell}\int_{B^c} \frac{\varphi_i(x)\varphi(x)}{|x-y|^{2+2\alpha}} dydx. \tag{3.10}$$



Фигура 3.1: Пример на допустима триангулация за квадратна област Ω с обвиващ кръг B.

3.2 Пренареждане на неизвестните

Както беше отбелязано в Раздел 1.2.1 за определени класове задачи е възможно неизвестните да бъдат пренаредени така, че да се подобри съществено *структурата* на матрицата на системата. Ще припомним, че за задачата разглеждана в предходната Глава 2 не беше прилагано пренареждане. Причината е, че естествената (последователна) номерация на възлите по границите на крилните профили съответства достатъчно добре на свойствата на матрицата.

Представеният в тази глава анализ показва, че за матриците породени от задачата за дробна дифузия пренареждането е необходимо. На Фигура 3.4а и Фигура 3.5а са показани оригиналните номерации на възлите в триангулацията. Те се получават от MatLab функцията `initmesh` при генериране на крайноелементните мрежи. *Структурата* на матрицата съответстваща на това подреждане не е подходяща за йерархичния солвър от пакета STRUMPACK.

В [70] са разгледани първите два варианта за пренареждане на неизвест-

ните – по хоризонтални ленти („stripes“) и по спирала около центъра „snake“. Следвайки *линейния* подход пренареждане по координата ос Y („top“) е анализирано в [71]. В [76] са изследвани две принципно различни нови пренареждания, съответстващи на метода на вложените сечения (Nested Dissection) и на рекурсивната бисекция (Recursive Bisection). Описание на MatLab кодове реализиращи използваните пренареждания е представен в Приложение А в края на дисертацията.

Нека означим с $P_i(X_i, Y_i) = x_i$, $i \in [1, \ldots, N]$, $x_i \in \Omega$ номерацията на възлите на триангулацията $\mathcal{T}$. Целта на изследванията е да се намери подходящо пренареждане $\tilde{P} = (\tilde{P}_i)_{i=1}^{N}$, което подобрява структурата на матрицата на коравина K, като по този начин се подобрява изчислителната ефективност на йерархичната HSS компресия. Изследваните пренареждания и структурата на получените матрици са визуализирани на Фигура 3.4 и 3.5. Ще отбележим, че във визуализацията матрицата е концентрирана около извъндиагоналните елементи. Диагоналните елементи от друга страна са с много по-високи стойности от скалата във фигурите.

3.2.1 Пренареждане по Y координата – „top“

При това пренареждане съответстващите на възлите от триангулацията на Ω неизвестни се сортират по тяхната Y координата. Трябва да отбележим, че избирането на Y е условно. Пренареждане по X координатата или дори по диагонал води до аналогична изчислителна ефективност. В представените числени експерименти това пренареждане се нарича „top“. Новата номерация и структурата на получената матрица са визуализирани на Фигура 3.4б и Фигура 3.5в.

3.2.2 Пренареждане по линии – „stripes“

При това пренареждане новата номерация на възлите от мрежата е по хоризонтални линии. Започва се от „горния ляв“ връх. Тоест елементът P_i в Ω с най-голяма сума $\tilde{P}_1 = P_{i_1} : \max(-X_i + Y_i)$, $i \in [1, \ldots, N]$ се избира за първи елемент $\tilde{P}_1$ в пренареждането $\tilde{P}$. След това от съседните на този елемент се избира втори елемент $\tilde{P}_2$, такъв че ъгълът между оста $\overrightarrow{Oy}$ и вектора $\overrightarrow{\tilde{P}_1\tilde{P}_2}$ да бъде минимален в интервала $[0°, 180°]$. Ако такъв елемент не може да бъде открит, за следващ се избира такъв елемент, за който е в сила условието $P_i : \max(-X_i + Y_i)$, $i \in [1, ..., N], i \notin P^i$, където P^i е множеството от върхове, които вече са били избрани.

Пример за това пренареждане и структурата на получената матрица са показани на Фигура 3.4в и Фигура 3.5г.

3.2.3 Пренареждане по спирала – „snake“

Този алгоритъм пренарежда неизвестните по спирала наподобяваща навита змия. Избира се горния ляв възел за първи възел в пренареждането $\tilde{P}_1 = P_i : \max(-X_i + Y_i)$, $i \in [1, \ldots, N]$. Добавя се помощен възел $\tilde{P}_0 = \tilde{P}_{\tilde{X}_1 - a, \tilde{Y}_1 + a}$, където $a > 0$, $a \in \mathbb{R}^+$. Т.е. $\tilde{P}_0$ е помощен възел от мрежата намиращ се „горе вляво“ на първия възел в пренареждането.

Следващият възел $\tilde{P}_j = P_{i_j}$ се избира от множеството възли P^{*j-1}, съседни на текущият избран възел $\tilde{P}_{j-1}$, които не са били избрани в предходните стъпки. Ако текущият възел няма неизбрани съседи $P^{*j-1} = \emptyset$ се разглеждат всички неизбрани възли P^j. Избира се такъв възел $\tilde{P}_2$, че ъгълът образуван от предишния, текущия и следващия да е минимален:

$$\tilde{P}_j = P_{i_j} : \min \sphericalangle \tilde{P}_{j-2}\tilde{P}_{j-1}P_{i_j}, \quad \begin{cases} P^{*j-1} \neq \emptyset : & P_{i_j} \in P^{*j-1} \\ P^{*j-1} = \emptyset : & P_{i_j} \in P^j \end{cases}$$

Това пренареждане и структурата на получената матрица са визуализирани на Фигура 3.4г и Фигура 3.5б.

3.2.4 Пренареждане по метода на вложените сечения

Методът на вложените сечения (Nested Dissection) е въведен в [31] от Джордж (George). Описание на метода може да се намери и в [90]. Използва се подхода „разделяй и владей“ за разделяне на графа представящ структурата на ненулевите елементи на зададена разредена симетрична матрица.

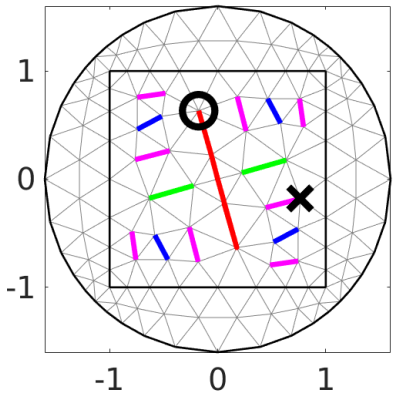
Много популярен такъв пример е симетричната и положително определена матрица на коравина получена при дискретизация на двумерна елиптична задача с линейни крайни елементи дефинирани върху мрежа от триъгълници. Геометричният аналог от този пример е в основата на използваното пренареждане по метода на вложените сечения. Алгоритъмът включва три стъпки:

1. Конструиране на неориентиран граф, съответстващ на триангулацията $\mathcal{T}$ в който върхове са възлите от мрежата $x_i \in \Omega$, $i \in [1, \ldots, N]$, а ребрата са страните на съответните триъгълници.

2. Рекурсивно разделяне на графа използвайки разделители (такива малки подмножества от върхове, че когато се премахнат графът се разделя на подграфи). Целта на това разделяне е получаването на два максимално близки по размер подграфа. Възможно е разделянето да бъде и на повече части. Така например в работата на Джордж разделянето е на четири части [31]. Тази процедура продължава рекурсивно.

3. Пренареждане на възлите в съответствие с рекурсивната структура: първо по подграфи и след това по разделители.

Целта на метода на вложените сечения е намаляване на запълването на разредената матрица в процеса на гаусова елиминация (или факторизация на Холецки) до квадрата на размера на разделителите на всяко ниво. За равнинни графи, като този получен при дискретизация с метод на крайните елементи в ограничена двумерна област, запълването е $O(n \log n)$ [50]. Прилагане на метода на вложените сечения за системи с разредени матрици е представено в [62, 82, 6]. Важно е да отбележим, че разглежданата от нас матрица K е плътна. Ако се дискретизира локален лапласиан (т.е. обикновена дифузия), то получената матрица би била разредена и методът на вложените сечения би дал най-добрия известен резултат за намаляване на запълването и подобряване на ефективността на факторизацията на Холецки.



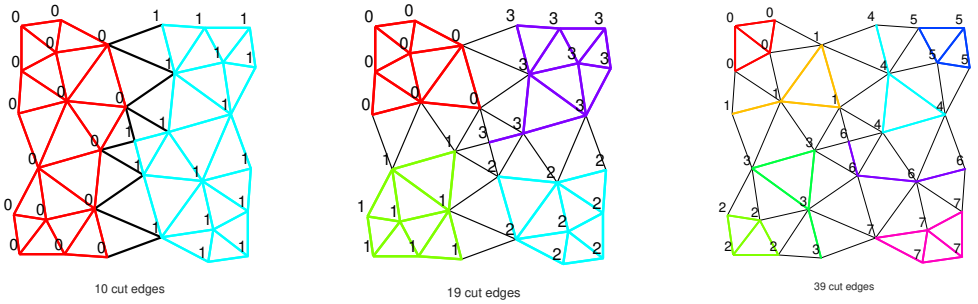
Фигура 3.2: Пренареждане с метод на вложените сечения за квадратна област Ω. С „X“ е означен първия връх, а с „O“ последния.

Разделителите за тестовата задача в квадратна област са показани на Фигура 3.2. Пренареждането и получената структура на матрицата са визуализирани на Фигура 3.4г и Фигура 3.5б. За реализация на това пренареждане е използван пакета от MatLab `meshpart` [34].

3.2.5 Пренареждане по метода на рекурсивна бисекция

Разглеждаме рекурсивната бисекция като алтернативен подход за разделяне на графа на две части [68]. За разлика от метода на вложените сечения, в този случай графът се разделя чрез премахване на множество от ребра. Процесът продължава рекурсивно, като всеки от двата подграфа се разделя на две по

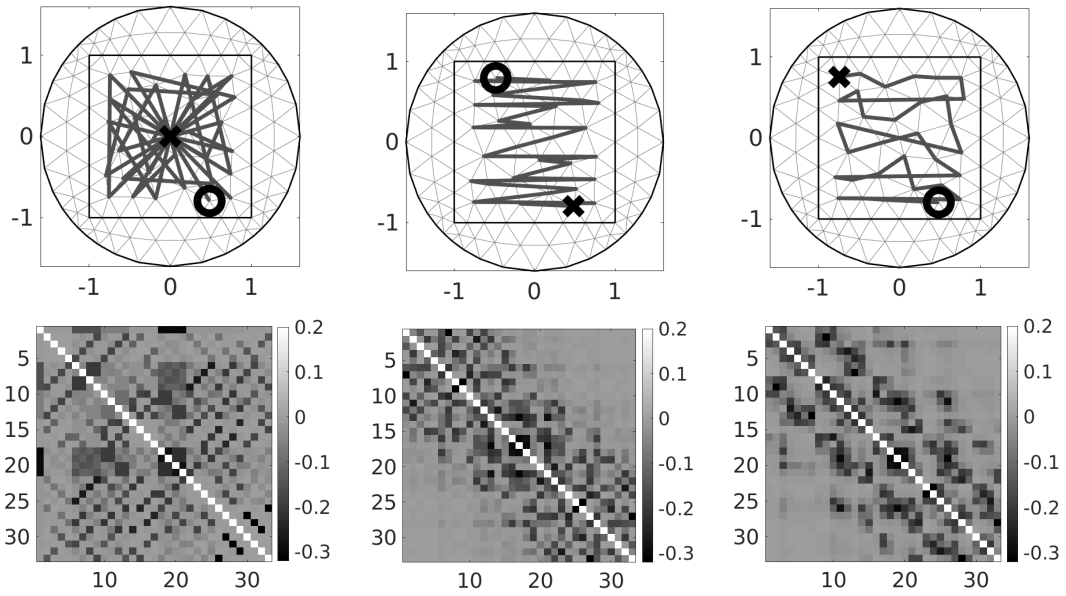
същият начин и така нататък. Частите на всяко ниво на рекурсивното разделяне се номерират последователно. На Фигура 3.3 е визуализиран процеса за квадратна област.



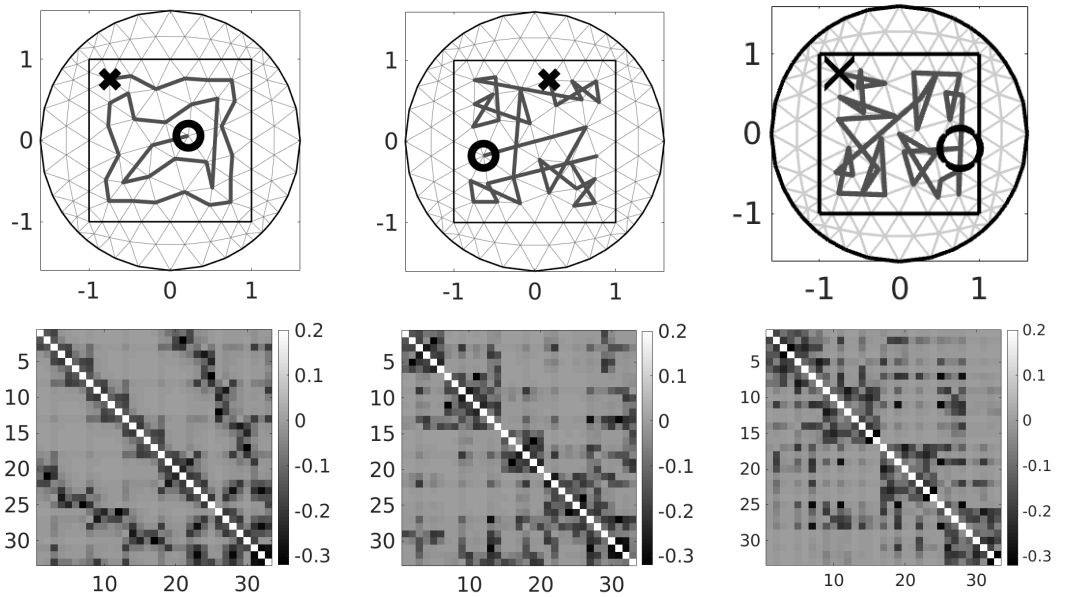
(а) Първо разделяне (б) Второ разделяне (в) Трето разделяне

Фигура 3.3: Три нива на разделяне с рекурсивна бисекция на квадратна област. Числата до върховете и цветовете на ребрата показват подграфите след поредното разделяне. Черните ребра са премахнатите разделители.

Рекурсивната бисекция се използва за балансиране на натоварването при решаване на задачи върху изчислителни системи с разпределена памет [68]. В статията на Реброва и др. [64] са анализирани няколко техники за пренареждане на променливите при прилагане на йерархична полусепарабелна компресия за решаване на системи линейни уравнения с плътни матрици породени от хребетообразна регресия с ядро (Kernel Ridge Regression). В настоящата дисертация използваме геометрични сепаратори от MatLab пакета `meshpart` [34].

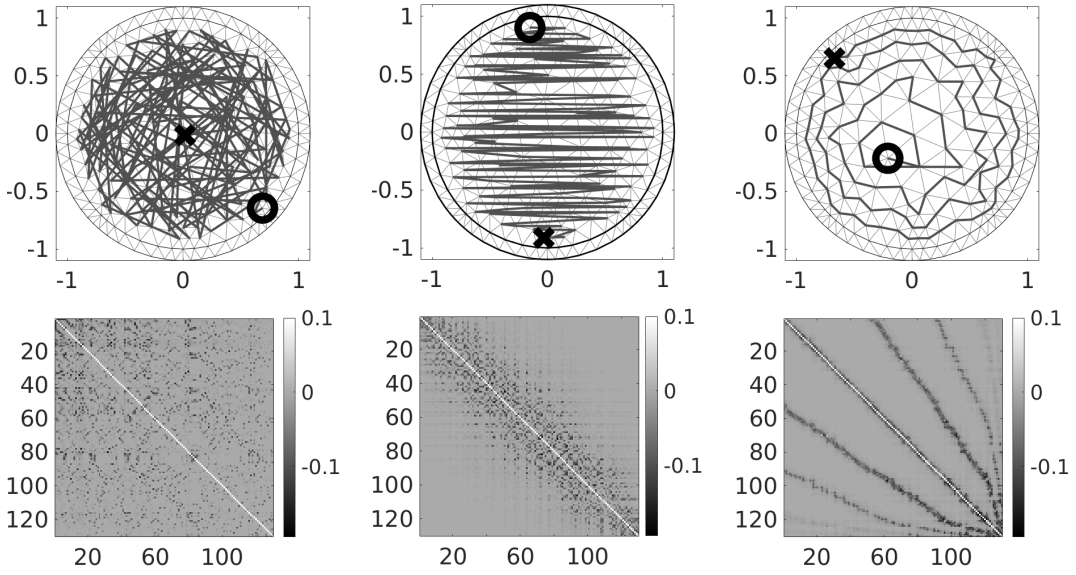


(а) Без пренареждане (б) „top“ (в) „stripes“

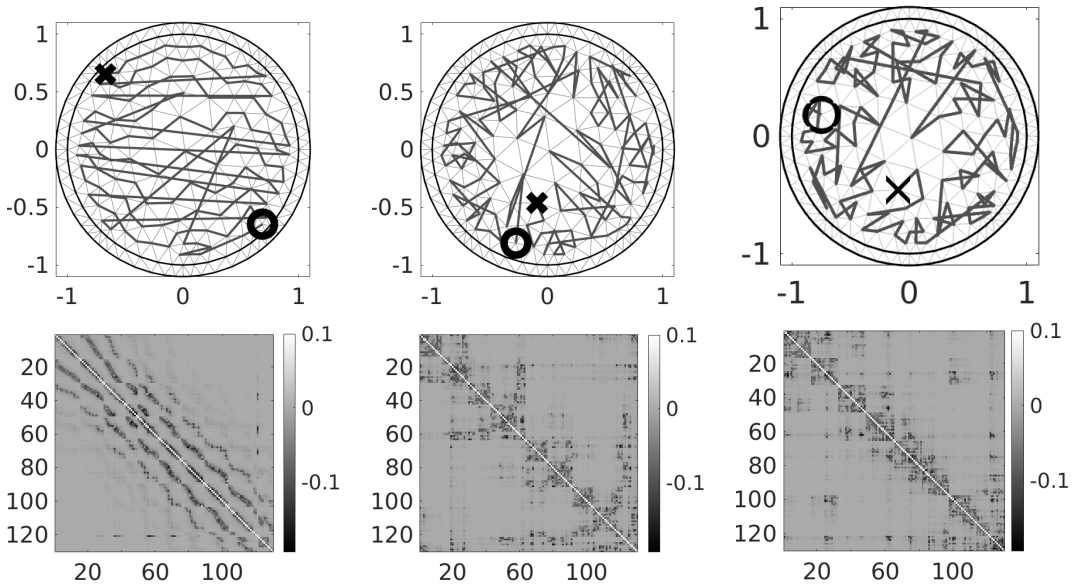


(г) „snake“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.4: Пренареждане на върховете в квадратна област Ω (горе) и структура на съответната матрица K. Тъмносивите линии показват подреждането на възлите от първия (маркиран с „X“) до последния (маркиран с „O“).



(а) Без пренареждане (б) „top“ (в) „snake“



(г) „stripes“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.5: Пренареждане на възлите в кръгла област Ω (горе) и структура на съответстващата матрица K . Тъмносивите линни показват подреждането на възлите от първия (маркиран с „X“) до последния (маркиран с „O“).

3.3 Анализ на числени експерименти върху компютърни системи с обща памет

Експерименталните резултати анализирани в този раздел, са получени върху изчислителна система с обща памет. Използван е един сървър от суперкомпютъра AVITOHOL. Възелът интегрира два процесора Intel Xeon E 2650 v2 CPUs с общо 16 ядра позволяващи използването на до 16 нишки. Този тип процесори дават възможност за хипертрединг с по 2 нишки на ядро, което обаче не води до подобряване на паралелната ефективност на анализираните методи и алгоритми. По тази причина в представените резултати не са включени числени експерименти с хипертрединг.

Оригиналната номерация на възлите (неизвестните) се получава в резултат от генерирането на мрежата реализирано в програмата публикувана в статия [4]. Без пренареждане структурата на плътната матрица породена от дискретизацията с метода на крайните елементи на задачата за *дробна* дифузия не е подходяща за прилагане на HSS компресия. С цел подобряване ефективността на йерархичния солвър от пакета STRUMPACK са анализирани няколко подхода за пренареждане на неизвестните. Резултатите за пренареждания от тип „snake“ и „stripes“ са публикувани в [70], а за „top“ – в [71]. В работа [76] са представени методите на вложените сечения и на рекурсивната бисекция, като е направен сравнителен анализ на ефективността на всичките пет пренареждания. Числените експерименти са за задачи с *дробен* лапласиан със степен $\alpha = 0.5$. Визуализация на численото решение на задачата за квадратна и кръгла област е показана на Фигура 3.6. Анализираните числени експерименти са за системи с плътни матрици с брой на неизвестните от $\sim$2 000 до $\sim$32 000.

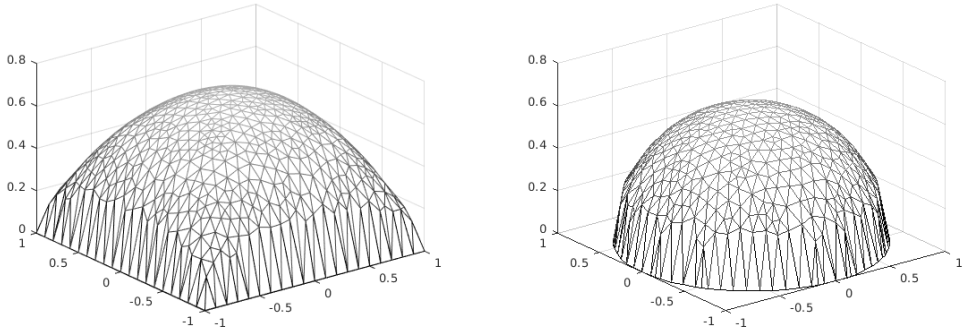
За генериране на плътните системи линейни алгебрични уравнения се използва MatLab програмата на Акоста и др. от [4], като матрицата и десните страни се записват в текстови файлове. Използват се MatLab алгоритмите описани в Раздел 3.2 и в Приложение А за генериране на подходящи пренареждания на неизвестните. Числените експерименти се реализират с програма, която зарежда матрицата, дясната част и индексите в пренареждането в паметта, изпълнява зададеното пренареждане с LAPACK функциите dlapmt и dlapmr (когато се използва такова) и извиква съответния солвър за система линейни алгебрични уравнения с плътна матрица.

В този раздел анализираме производителността на двата разглеждани метода на базата на тяхната реализация в следните софтуерни пакети за изчислителни системи с обща памет:

- За Гаусова елиминация (LU факторизация) – Intel's Math Kernel Library (MKL). Този избор се базира на анализираната в предишната Глава 2

производителност на софтуерни пакети, използващи LU факторизация.

- За йерархичната полусепарабелна компресия (HSS) и ULV-подобна факторизация – STRUctured Matrix PACKage (STRUMPACK).



(а) Квадратна област (б) Кръгла област

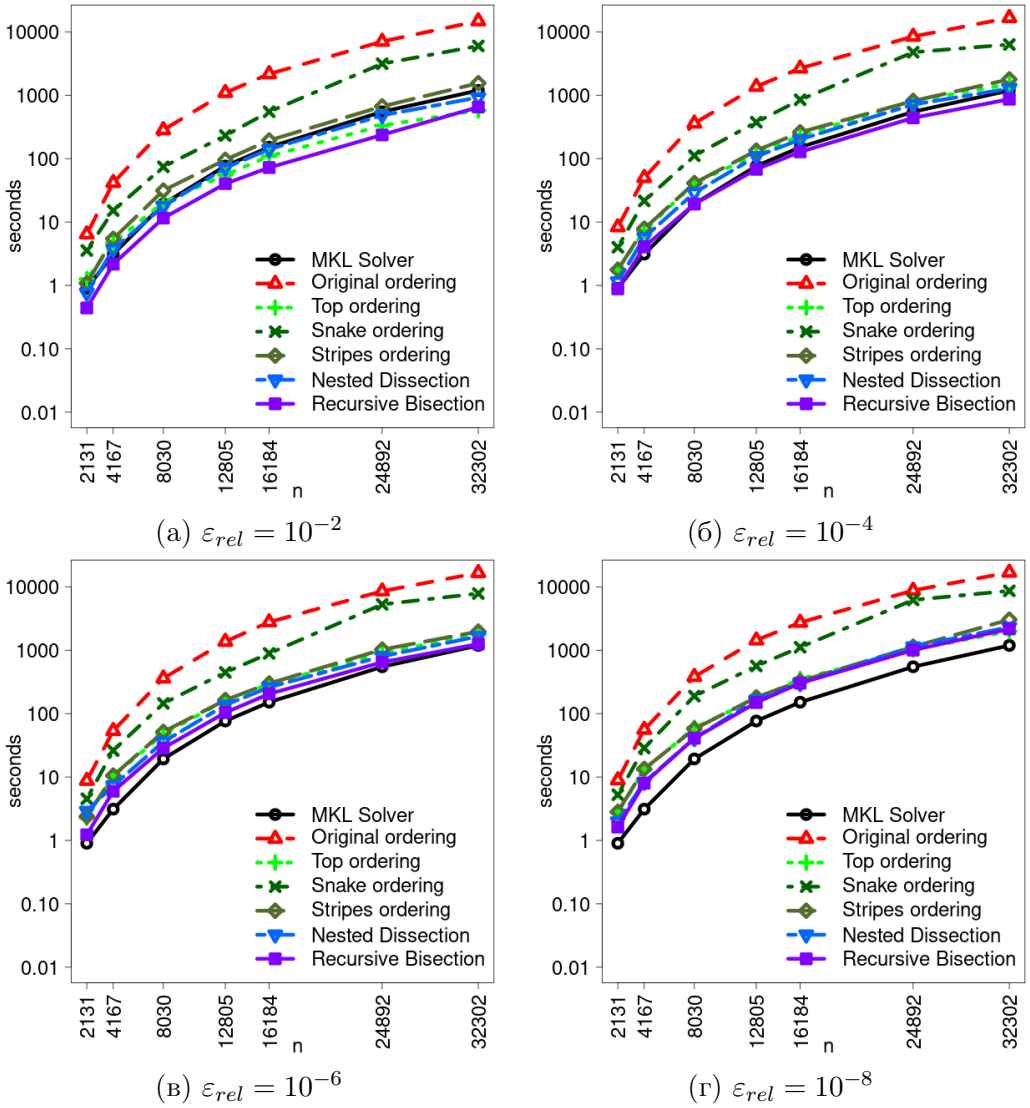
Фигура 3.6: Решение на задачата за *дробна* дифузия за квадратна и кръгла област.

Ефектът от предложените пренареждания на неизвестните се проявява само при HSS компресията. За анализ на относителната грешката на метода на HSS компресия се прилага (2.5), като за референтно се използва решението получено с MKL. Числените експерименти с йерархична полусепарабелна компресия са проведени при използване на абсолютен праг на грешка $\varepsilon_{\text{abs}} = 10^{-2}$ и вариране на относителния праг на грешка $\varepsilon_{\text{rel}} = 10^{-2}, 10^{-4}, 10^{-6}$ и 10^{-8}.

3.3.1 Квадратна област

На Фигура 3.7 са представени последователните времена (при използване на една нишка) за решаване на системата линейни алгебрични уравнения (3.7). При използване на най-голямата стойност на относителен праг $\varepsilon_{\text{rel}} = 10^{-2}$ пренареждането от тип „top“ показва по-добри резултати от метода на вложените сечения. В повечето случаи йерархичният солвър показва най-добри времена при рекурсивната бисекция, следвана от метод на вложените сечения, „top“ и „stripes“. Най-бавно работи STRUMPACK солвърът при пренареждане „snake“, но дори и с него се получават значително по-бързо решаване на задачата в сравнение с началната номерация на неизвестните. Експериментите показват, че MKL има по-добра производителност от STRUMPACK за всички стойности на относителният праг с изключение на $\varepsilon_{\text{rel}} = 10^{-2}$, където резултатите за рекурсивната бисекция, „stripes“ и „top“ пренареждания са по-добри.

Можем да направим също така извода, че като изключим варианта без пренареждане и пренареждането от тип „snake“, STRUMPACK показва относително близка до MKL производителност.



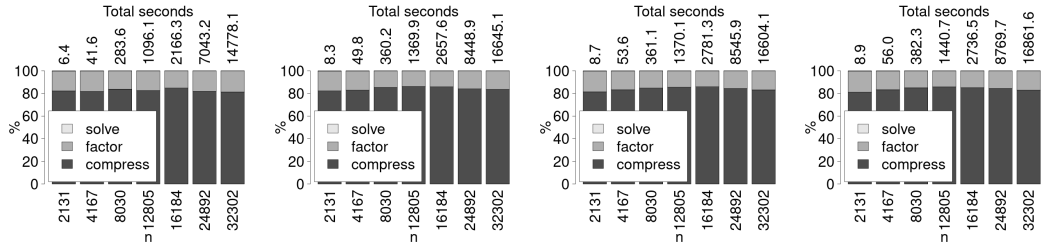
(а) $\varepsilon_{rel} = 10^{-2}$ (б) $\varepsilon_{rel} = 10^{-4}$

(в) $\varepsilon_{rel} = 10^{-6}$ (г) $\varepsilon_{rel} = 10^{-8}$

Фигура 3.7: Сравнителен анализ на времената за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за случая на квадратна област.

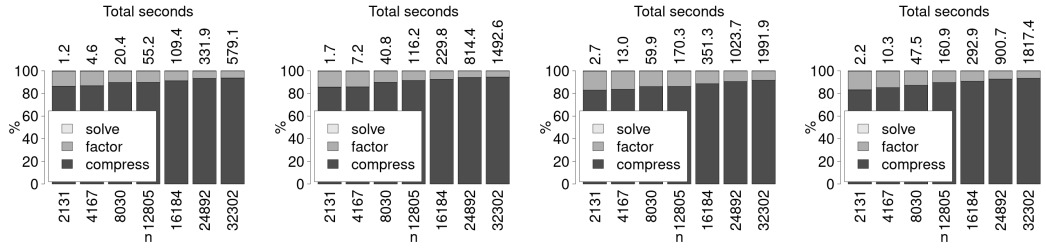
Сравнение на времената за изпълнение на отделните части на йерархичния метод

На Фигура 3.8 и Фигура 3.9 са представени времената за изпълнение за различните части за реализация на метода за решаване на системата с помощта на йерархична полусепарабелна (HSS) компресия в случая на квадратна област. Конструирането на компресираната матрица H отнема най-голяма част от времето за изчисление. Когато компресията е по-ефективна (като бързодействие и максимален извъндиагонален ранг, виж Фигура 3.10 и 3.11), времето за изпълнението ѝ доминира над времето, необходимо за получаване на ULV-подобната факторизация. Решаването на системата с факторизираната матрица отнема много малка част от цялото време.



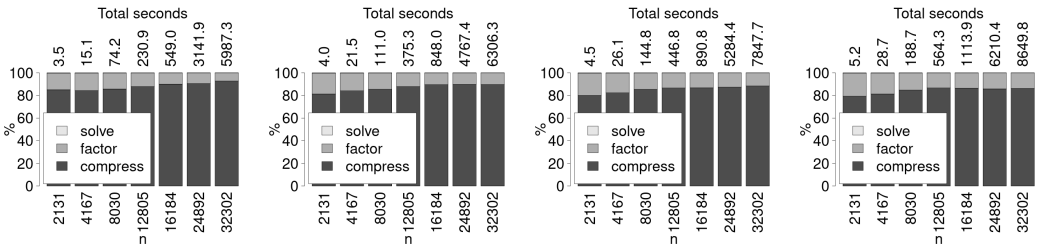
$\varepsilon_{\text{rek}} = 10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

(а) Без пренареждане



$\varepsilon_{\text{rel}}10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

(б) „top“

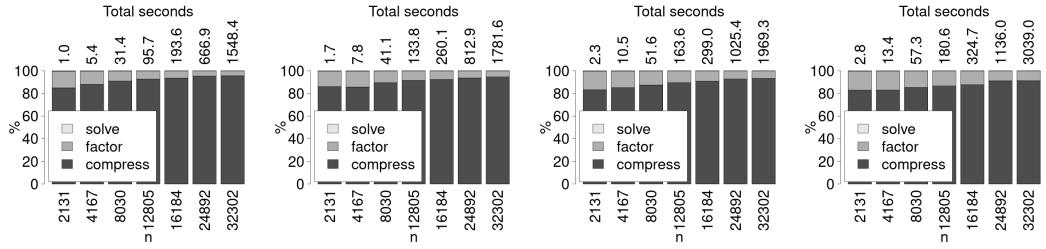


1

$\varepsilon_{\text{rel}}10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

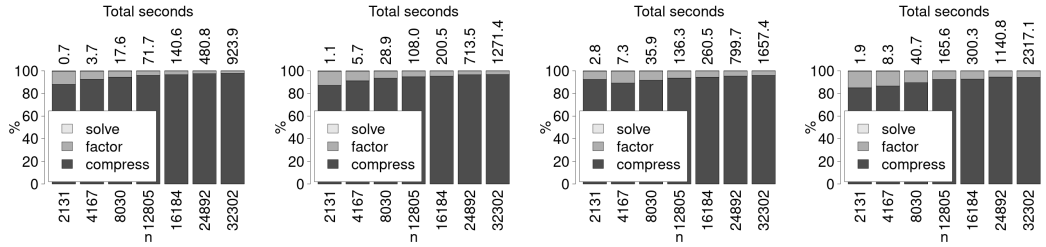
(в) „snake“

Фигура 3.8: Сравнение на последователните времена за йерархична полусепарабелна компресия, ULV-подобна факторизация и решаване на системата с факторизираната матрица за квадратна област при оригинално подреждане на неизвестните и пренареждания от тип „top“ и „snake“.



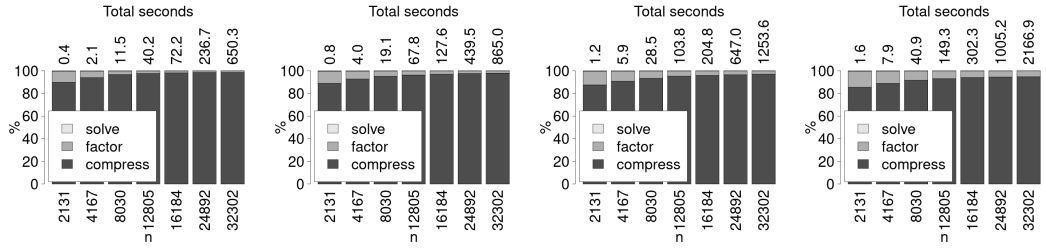
$\varepsilon_{\text{rel}}10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

(а) „stripes“



$\varepsilon_{\text{rel}}10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

(б) Вложени сечения



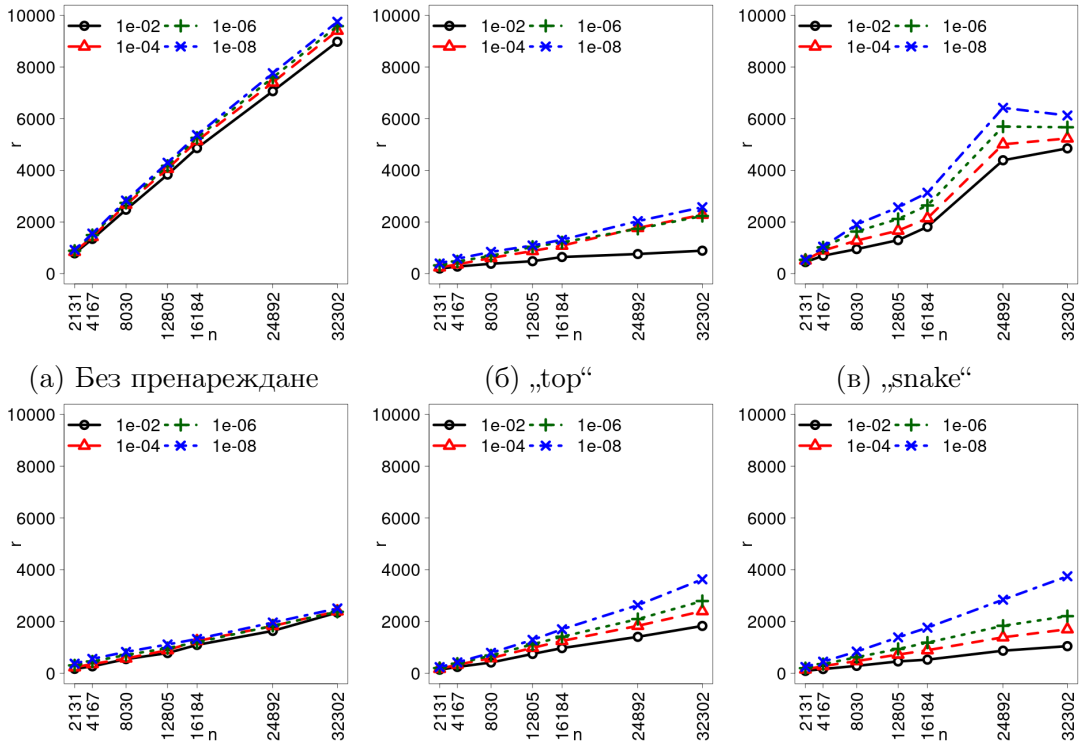
$\varepsilon_{\text{rel}}10^{-2}$ $\varepsilon_{\text{rel}}10^{-4}$ $\varepsilon_{\text{rel}}10^{-6}$ $\varepsilon_{\text{rel}}10^{-8}$

(в) Рекурсивна бисекция

Фигура 3.9: Сравнение на последователните времена за йерархична полусепарабелна компресия, ULV-подобна факторизация и решаване на системата с факторизираната матрица за квадратна област при пренареждания от тип „stripes“, вложени сечения и рекурсивна бисекция.

Извъндиагонален ранг

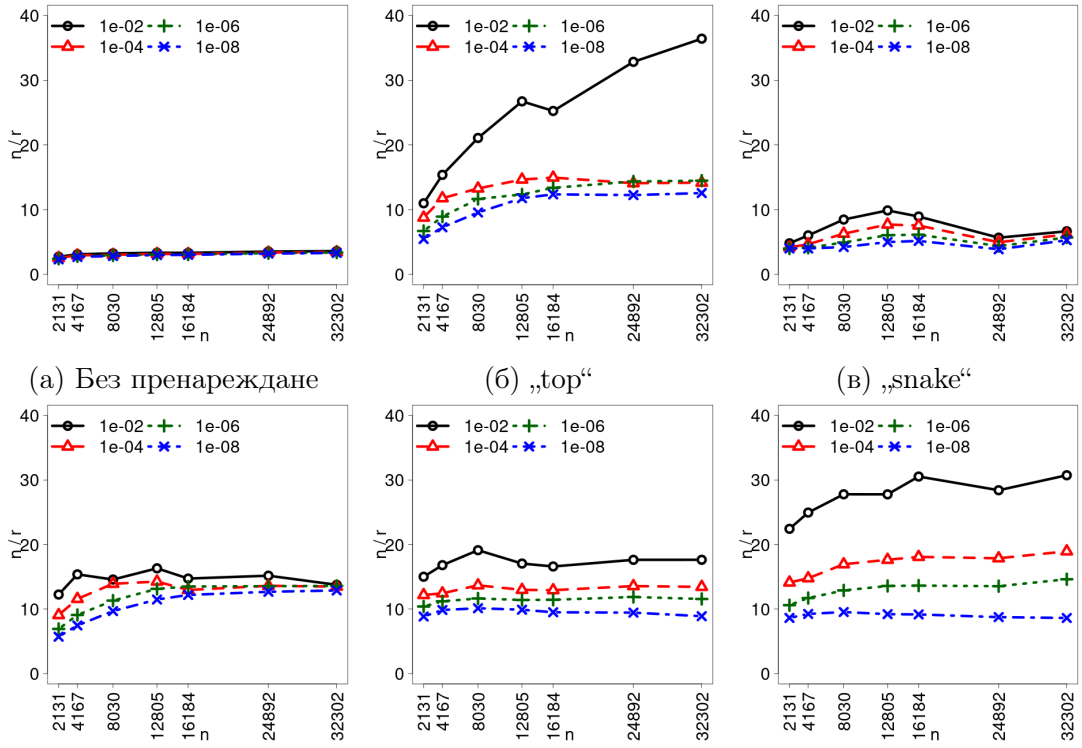
На Фигура 3.10 е представен извъндиагоналният ранг r изчислен в процеса на HSS компресията, като на Фигура 3.11 са показани отношенията n/r, където n е броят на неизвестните. Стойността на r е мярка за ефективността на компресията за дадената матрица. Колкото по-малък е рангът, толкова по-ефективна е компресията. При оригиналното подреждане стойностите на r са най-големи, като варират между $1/3$ и $1/4$ от броя на неизвестните. Следващ по големина ранг се получава при пренареждането „snake“. Рангът за останалите пренареждания има относително близки стойности, като за повечето от разгледаните примери за рекурсивната бисекция се получава най-висока ефективност на йерархичната полусепарабелна компресия. Ролята на ранга се потвърждава и от анализираните в предходния раздел резултати показани на Фигура 3.7.



(а) Без пренареждане (б) „top“ (в) „snake“

(г) „stripes“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.10: Максимален извъндиагонален ранг r.



(а) Без пренареждане (б) „top“ (в) „snake“

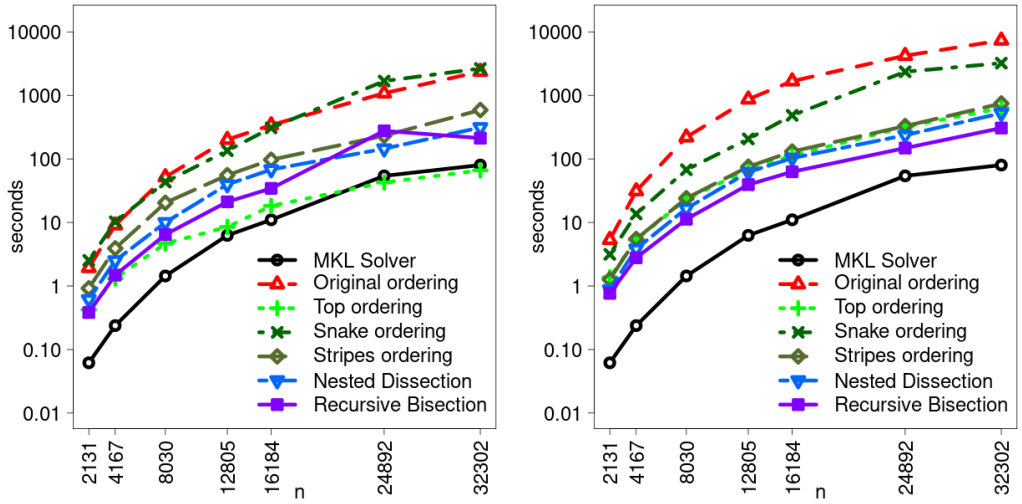
(г) „stripes“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.11: Отношение на броя на неизвестни n и максималния ранг r.

Паралелни времена и ускорения

В този раздел анализираме паралелната ефективност. Проведени са числени експерименти за матрици с нарастваща размерност $n \in [2\,131, 32\,302]$, които са получени при дискретизация на задачата с дробна дифузия в квадратната област. На Фигура 3.12 са представени резултати получени при използване на 16 нишки. Виждаме, че графиките имат подобно поведение, като за по-големите задачи (по-големите стойности на n) ускоренията стават близки до линейни. Това е в съответствие с теоретичните оценки. Отбелязваме, че за $\varepsilon_{\text{rel}} = 10^{-2}$ и пренареждане „top“ йерархична HSS компресия е с по-добри времена за двете най-големи системи, т.е. при $n = 24\,892$ и $n = 32\,302$. В останалите случаи директният гаусов солвър MKL е по-бърз. Това се дължи и на по-сложната рекурсивна *структура* на HSS компресията. Можем да очакваме, че при по-големи стойности на n, STRUMPACK може да покаже по-добри времена от MKL и за по-малки стойности на прага на относителна грешка ε_{rel}.

На Фигура 3.13 са представени паралелните ускорения на отделните части от йерархичният солвър при използване на рекурсивна бисекция. Останалите



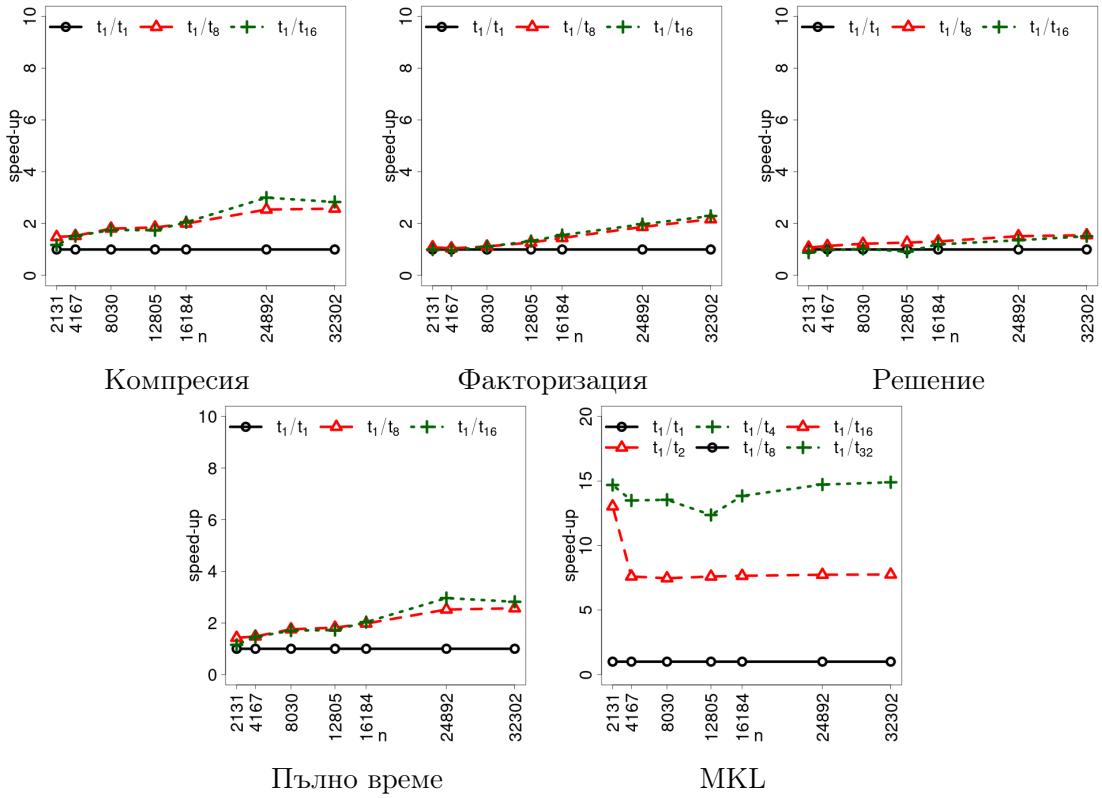
(а) $\varepsilon_{rel} = 10^{-2}$ (б) $\varepsilon_{rel} = 10^{-4}$

(в) $\varepsilon_{rel} = 10^{-6}$ (г) $\varepsilon_{rel} = 10^{-8}$

Фигура 3.12: Сравнение на паралелните времена за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за квадратна област.

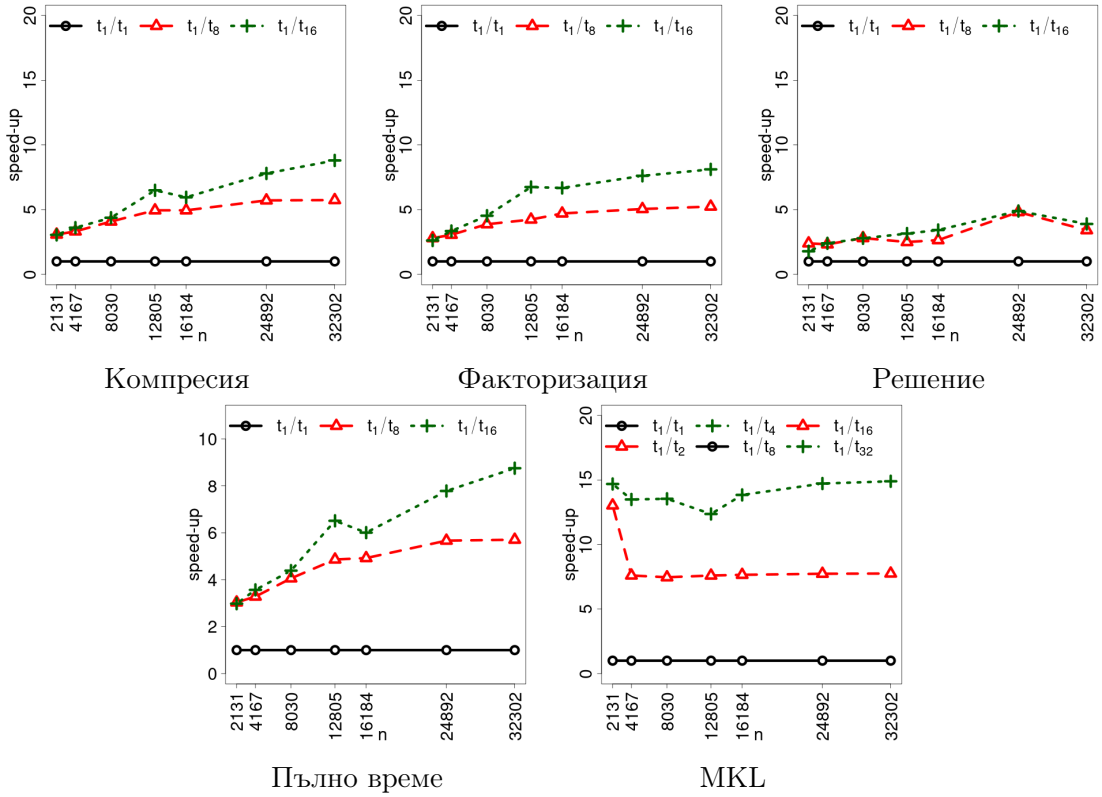
пренареждания и оригиналното подреждане имат подобно паралелно ускорение. Изключение прави „top“, като този случай ще разгледаме допълнително. Най-добро паралелно ускорение показва HSS компресията – достигащо до ∼3-4 (Фигура 3.13а). Ще припомним, че компресията отнема най-голяма част от времето за решаване на задачата (Фигура 3.8 и Фигура 3.9). Това се вижда и от паралелното ускорение на цялото време Фигура 3.13г. Факторизация-

та показва по-ниско паралелно ускорение достигащо до ∼2 (Фигура 3.13б). Решаването на системата с факторизираната матрица показва най-малко паралелно ускорение ∼ 1.5 Фигура 3.13в, но то отнема и най-малка част от общото време и почти не влияе на общото паралелно време. За сравнение е показано и почти оптималното ускорението при решаване на системата с LU факторизация от пакета MKL – Фигура 3.13д.



Фигура 3.13: Паралелно ускорение на отделните стъпки на йерархичния солвър за $\varepsilon_{\text{rel}} = 10^{-2}$ с пренареждане на неизвестните по метода на рекурсивната бисекция и сравнение с ускорението на гаусовият солвър от MKL за квадратна област.

Пренареждането „top“ показва по-добри паралелни времена при относителен праг $\varepsilon_{\text{rel}} = 10^{-2}$. Ускорението е също по-добро достигащо до ∼8 – Фигура 3.14. Такава паралелна ефективност се получава само при $\varepsilon_{\text{rel}} = 10^{-2}$. Това най-вероятно се дължи на специфичната структура на матрицата при квадратна област Ω, която от своя страна води до по-подходяща *структура* на пренареденатa матрицата при пренареждане „top“.



Фигура 3.14: Паралелно ускорение на отделните стъпки на йерархичния солвър за $\varepsilon_{\text{rel}} = 10^{-2}$ при пренареждане "top"и сравнение с ускорението на гаусовият солвър от MKL за квадратна област.

Анализ на грешката за HSS-базирания солвър

Нека припомним, че при прилагане на солвъра от пакета STRUMPACK базиран на йерархична полусепарабелна компресия получаваме *приближение* на решението на системата. Това се дължи на факта, че компресираната матрица H е апроксимация на изходната. Както и в предходната глава, ще анализираме относителната грешка на метода. В този анализ приемаме решението получено с директния гаусов солвър на MKL за референтно. Разглеждаме относителната грешка в l_2 норма, получена по формула (вече дефинирана в (2.5), която за удобство привеждаме отново):

$$R_{\text{relative}} = \frac{\left\|x^{\text{Gauss}} - x^{\text{HSS}}\right\|_{l_2}}{\left\|x^{\text{Gauss}}\right\|_{l_2}} = \frac{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}} - x_i^{\text{HSS}})^2}}{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}})^2}} \qquad \text{(2.5 копие)}$$

В Таблица 3.1 са представени относителните грешки за оригиналното под-

реждане и разгледаните пренареждания на неизвестните за квадратна област. За повечето експерименти относителната грешка е близка до зададения праг на относителна грешка ε_{rel}.

Относителната грешка зависи от ефективността на компресията. Това означава, че когато изчисленият ранг r на извъндиагоналните блокове е по-малък изходящата матрица се компресира по-ефективно, което води до по-малко изчислително време, но и до по-висока относителна грешка R_{relative} поради по-голямата компресия.

Таблица 3.1: Относителна грешка за квадратна област.

(а) Без пренареждане пренареждане „top“.

	Без пренареждане				„top“			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2131	0.0125	0.000124	1.633e-06	1.088e-08	0.122	0.00014	1.469e-06	1.003e-08
4167	0.0244	0.000211	2.195e-06	1.414e-08	0.194	0.000345	1.868e-06	1.95e-08
8030	0.0435	0.000422	7.515e-06	5.049e-08	0.237	0.00681	5.732e-06	3.446e-08
12805	6.74	0.0212	6.258e-06	5.04e-08	0.329	0.00976	4.3e-06	6.554e-08
16184	0.136	0.000818	1.307e-05	1.035e-07	0.335	0.00117	5.96e-06	6.16e-08
24892	0.115	0.000955	1.791e-05	1.659e-07	0.45	0.00192	4.920e-06	9.5e-08
32302	0.145	0.0011	3.409e-05	1.533e-07	0.479	0.00246	9.788e-06	9.09e-08

(б) Пренареждания „snake“ и „stripes“.

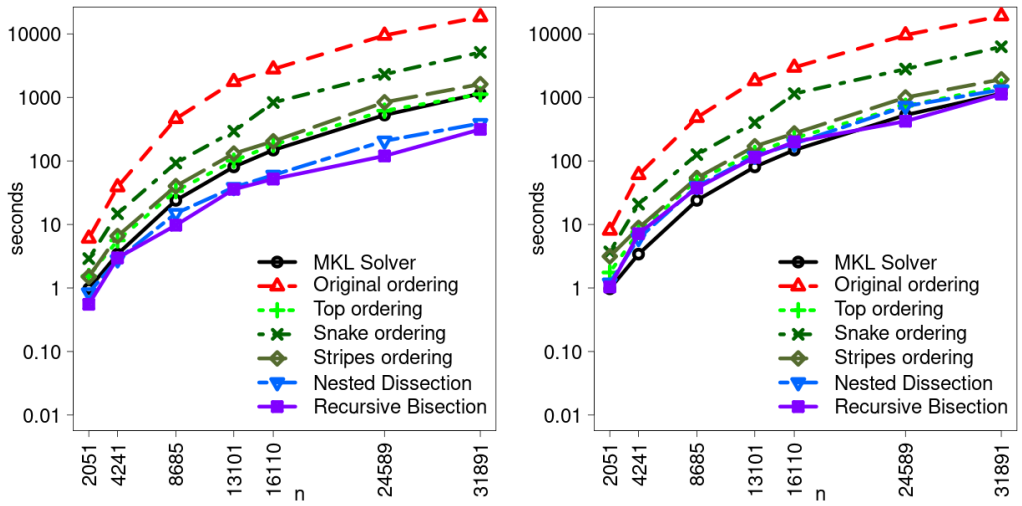
	„snake“				„stripes“			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2131	0.0786	0.000229	1.353e-06	1.144e-08	0.111	0.000324	2.01e-06	1.283e-08
4167	0.172	0.000355	2.212e-06	2.357e-08	0.193	0.000623	2.178e-06	4.491e-08
8030	0.206	0.00107	2.737e-06	5.342-08	0.287	0.00108	7.943e-06	4.421-08
12805	0.324	0.00286	6.105e-06	1.995e-07	0.382	0.00117	6.793e-06	7.977e-08
16184	0.397	0.00363	8.919e-06	9.394e-08	0.393	0.00163	5.446e-06	7.988e-08
24892	0.513	0.00753	1.32e-05	1.174e-07	0.478	0.00176	7.613e-06	1.211e-07
32302	0.587	0.00647	3.129e-05	1.774e-07	0.497	0.00231	9.161-06	1.088e-07

(в) Пренареждания по методите на вложените сечение и рекурсивната бисекция.

	Вложени сечения				Рекурсивна бисекция			
n	$R_{relative}$ for STRUMPACK with r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2131	0.145	0.000403	2.837e-06	3.022e-08	0.0892	0.000338	2.694e-06	2.373e-08
4167	0.247	0.00138	2.866e-06	6.963e-08	0.223	0.000987	3.28e-06	5.245e-08
8030	0.378	0.00221	6.636e-06	7.277e-08	0.373	0.00172	6.402e-06	7.322e-08
12805	0.469	0.00342	8.211e-06	1.159e-07	0.424	0.000929	7.06e-05	1.043e-07
16184	0.499	0.00268	8.668e-06	1.542e-07	0.487	0.00217	8.736e-06	1.745e-07
24892	0.583	0.00318	1.04e-05	1.536e-07	0.536	0.0033	1.738e-05	2.e-07
32302	0.615	0.00539	1.432e-05	2.159e-07	0.612	0.0031	1.419e-05	3.386e-07

3.3.2 Кръгла област

Представените тук експерименти са организирани по аналогичен начин на тези за квадратната област. Целта е да анализираме влиянието на геометрията на областта. Така например границата на областта Ω е много по-гладка. Както ще видим, това не се оказва съществено за ефективността на HSS компресията. Независимо от това, отбелязани са някои специфики, които са анализирани. На Фигура 3.15 са представени последователните времена за решаване на системата линейни алгебрични уравнения. Отново абсолютният праг е фиксиран на $\varepsilon_{\text{abs}} = 10^{-2}$, като се варира относителния праг $\varepsilon_{\text{rel}} = 10^{-2}, 10^{-4}, 10^{-6}$ и 10^{-8}. В повечето от представените експерименти рекурсивната бисекция показва по-добри времена на изпълнение от останалите пренареждания. След рекурсивната бисекция, най-добри времена са получени за вложените сечения, следвани от пренаредженията „top“ и „stripes“. Времето при пренареждане „snake“ отново е най-голямо, но все пак по-добро от времето при оригиналното подреждане. Пренаредженията с рекурсивна бисекция и вложени сечения показват по-добро време от директният солвър за $\varepsilon_{\text{rel}} = 10^{-2}$. MKL има по-добро време от останалите пренареждания, както и за останалите разгледани стойности на ε_{rel}.



(а) $\varepsilon_{rel} = 10^{-2}$ (б) $\varepsilon_{rel} = 10^{-4}$

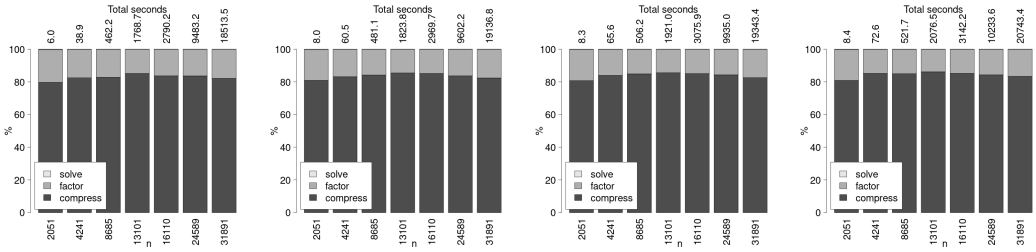
(в) $\varepsilon_{rel} = 10^{-6}$ (г) $\varepsilon_{rel} = 10^{-8}$

Фигура 3.15: Сравнение на времената за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за кръгла област.

Сравнение на времената за изпълнение на отделните части на йерархичния метод

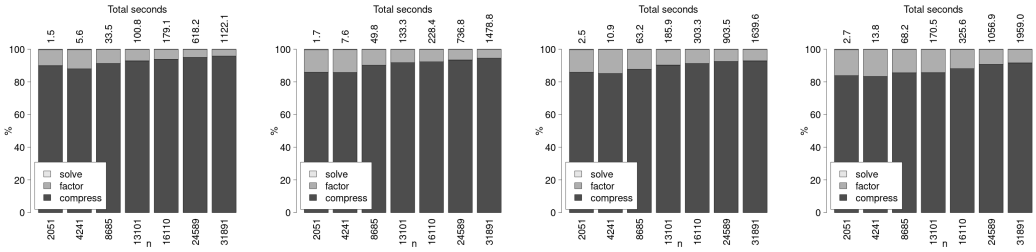
На Фигура 3.16 и Фигура 3.17 е представено сравнение на времето необходимо за йерархичната полусепарабелна компресия, ULV-подобната факторизация и решаването на системата с факторизираната матрица за кръгла област. Както

и при предишните експерименти в квадратна област, представени в Глава 2, времето за изпълнение на HSS компресията е най-голямо. Това се дължи на по-голямата изчислителна сложност на компресията – $O(n^2r)$, в сравнение с факторизацията – $O(nr^2)$ и решаването на системата с факторизираната матрица – $O(nr)$. При пренареждания с по-висока ефективност, ULV-подобната факторизация отнема по-малка част от общото време. Това е така, тъй като при тези пренареждания компресията е по-добра, като оставя по-малко количество „работа“ за факторизацията.



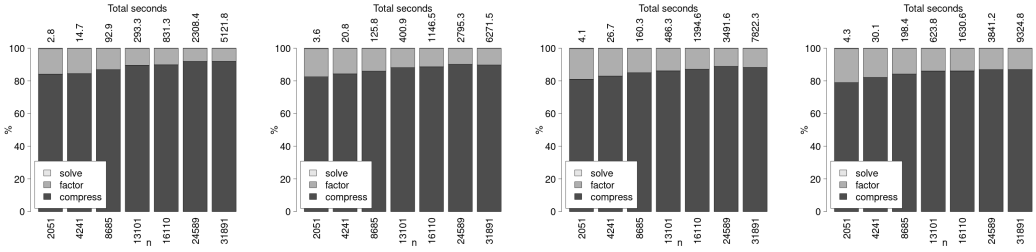
$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

(а) Без пренареждане



$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

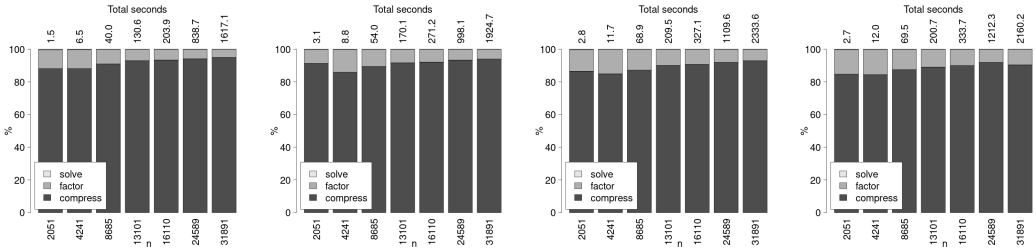
(б) Пренареждане „top“



$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

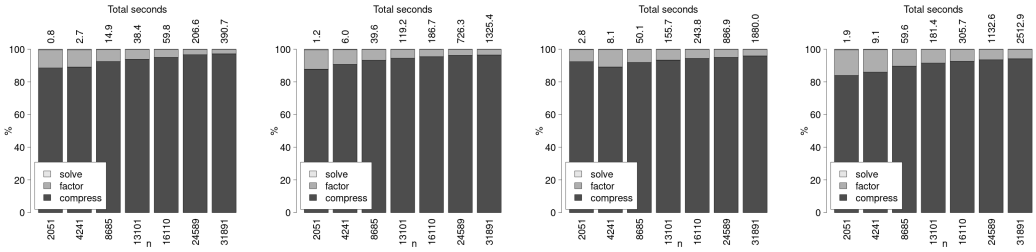
(в) Пренареждане „snake“

Фигура 3.16: Сравнение на последователните времена за йерархична полусепарабелна компресия, ULV-подобна факторизация и решаване на системата с факторизираната матрица за кръгла област при без пренареждане и пренареждания „top“ и „snake“.



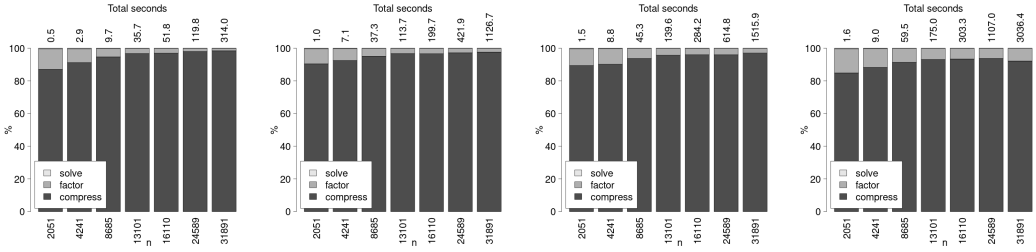
$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

(а) Пренареждане „stripes“



$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

(б) Вложени сечения



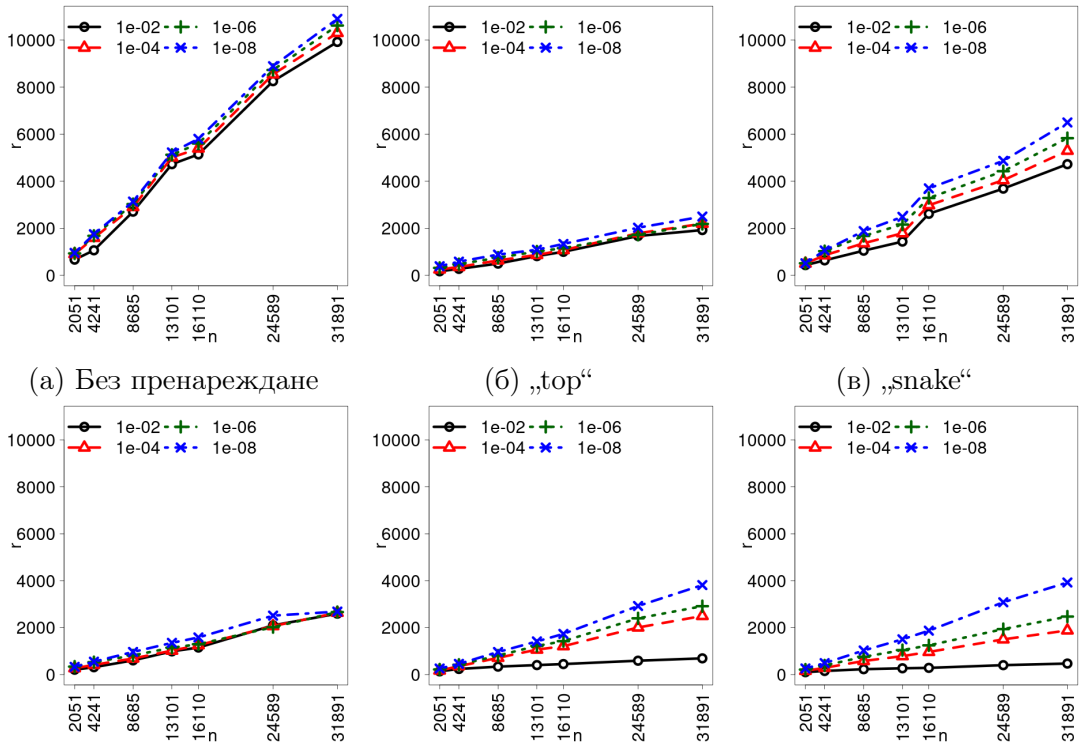
$\varepsilon_{10^{-2}}$ $\varepsilon_{10^{-4}}$ $\varepsilon_{10^{-6}}$ $\varepsilon_{10^{-8}}$

(в) Рекурсивна бисекция

Фигура 3.17: Сравнение на последователните времена за йерархична полусепарабелна компресия, ULV-подобна факторизация и решаване на системата с факторизираната матрица за кръгла област при пренареждания „stripes“, вложени сечения и рекурсивна бисекция.

Извъндиагонален ранг

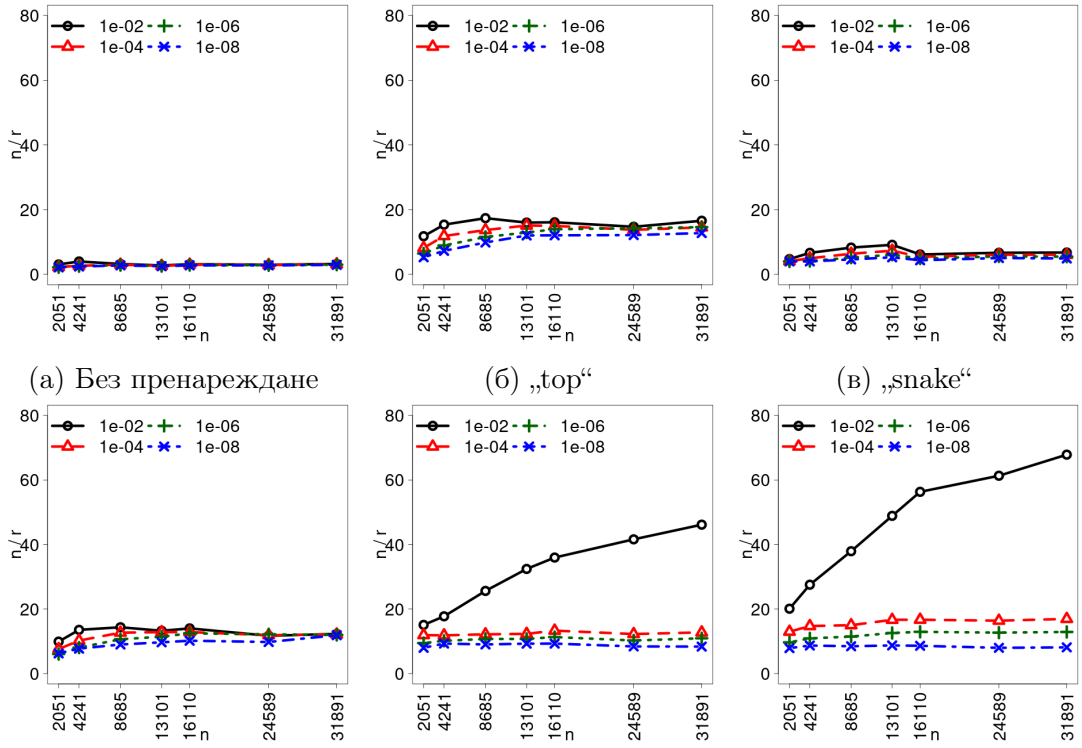
На Фигура 3.18 са представени стойностите на максималния извъндиагонален ранг r за оригиналното подреждане и разгледаните пренареждания. На Фигура 3.19 са показани отношенията на броя неизвестни n и съответния ранг. Ще припомним, че рангът е мярка за ефективността на компресията. При оригиналното подреждане рангът е най-голям, което съответства и на най-големите времена за изпълнение. След това с най-голям ранг (и съответно лоша ефективност на компресията) е пренареждането „snake“. За рекурсивната бисекция, вложените сечения, пренареждания „top“ и „stripes“ се получават близки по стойност рангове. Отново, при повечето експерименти, най-малък извъндиагонален ранг r и съответно най-ефективна компресия наблюдаваме при рекурсивната бисекция.



(а) Без пренареждане (б) „top“ (в) „snake“

(г) „stripes“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.18: Максимален извъндиагонален ранг r.



(а) Без пренареждане (б) „top“ (в) „snake“

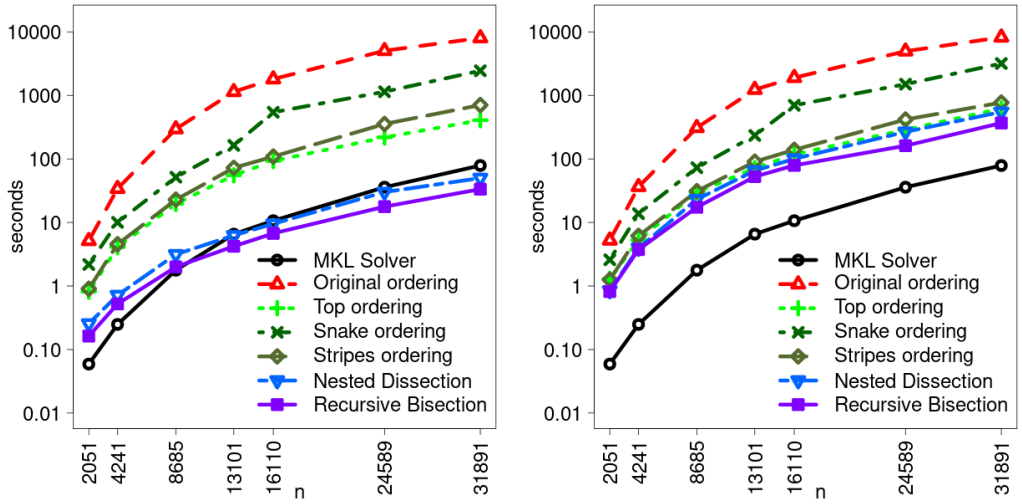
(г) „stripes“ (д) Вложени сечения (е) Рекурсивна бисекция

Фигура 3.19: Отношение на броя на неизвестните n и максималния ранг r за кръгла област.

Паралелни времена и ускорения

На Фигура 3.20 са представени паралелните времена за решаване на системата линейни алгебрични уравнения получени при дискретизация по МКЕ на моделната задача за *дробна* дифузия (с *дробен* лапласиан) в кръгла област, при които са използвани 16 нишки. Както при квадратната област, йерархичният солвър показва по-добри паралелни времена при пренареждане по методите на рекурсивната бисекция и вложените сечения за най-голямата тестова система с 31 819 неизвестни и при относителен праг $\varepsilon_{\text{rel}} = 10^{-2}$. В останалите случаи директният гаусов солвър реализиран в пакета MKL има по-добра паралелна производителност от STRUMPACK. Отбелязваме също така, че при повечето от разгледаните варианти за относителния праг ε_{rel}, рекурсивната бисекция отново е най-ефективното пренареждане. Изключение преви случаят $\varepsilon_{\text{rel}} = 10^{-8}$, за $n = 16\,110$ и $n = 31\,891$. Тогава по-добри резултати показват пренареждането по метода на вложени сечение и пренареждането „top“.

На Фигура 3.21 са показани паралелните ускорения за изпълнение на йе-



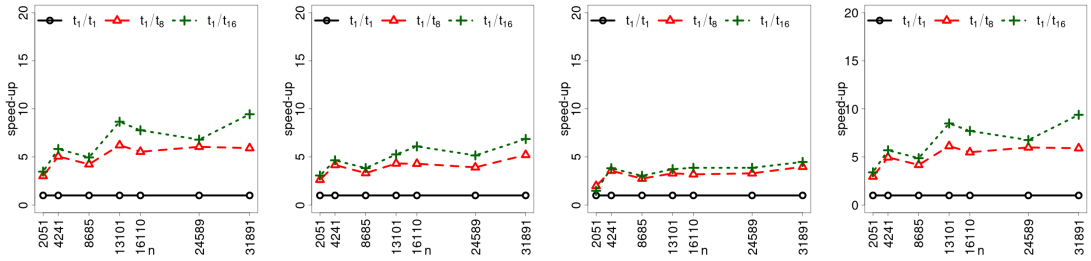
(а) $\varepsilon_{rel} = 10^{-2}$ (б) $\varepsilon_{rel} = 10^{-4}$

(в) $\varepsilon_{rel} = 10^{-6}$ (г) $\varepsilon_{rel} = 10^{-8}$

Фигура 3.20: Сравнение на паралелните времена за решаване на системата линейни алгебрични уравнения с MKL и STRUMPACK с пренареждания „top“, „snake“, „stripes“, вложени сечения (Nested Dissection) и рекурсивна бисекция (Recursive Bisection) за кръгла област.

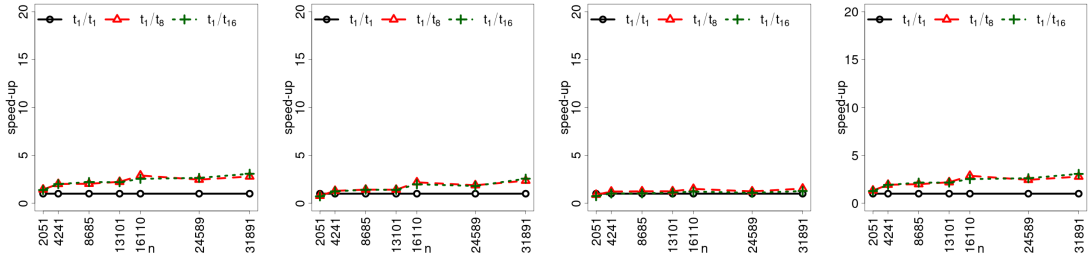
рархичната компресия, факторизацията и решаването на системата с факторизираната матрица при пренареждане по метода на рекурсивна бисекция. При $\varepsilon_{\text{rel}} = 10^{-2}$ паралелното ускорение достига ∼10, както се вижда на Фигура 3.21а. За останалите стойности на ε_{rel} максималните паралелни ускорения са в рамките ∼3-4. На Фигура 3.21б са представени паралелните ускорения за $\varepsilon_{\text{rel}} = 10^{-4}$, като за 10^{-6} и 10^{-8} ускоренията са подобни. За останалите

пренареждания паралелното ускорение варира от ∼2 до ∼8.



Компресия Факторизация Решение Пълно време

(а) $\varepsilon_{\text{rel}} = 10^{-2}$



Компресия Факторизация Решение Пълно време

(б) $\varepsilon_{\text{rel}} = 10^{-4}$

Фигура 3.21: Паралелно ускорение на отделните стъпки на йерархичния солвър за $\varepsilon_{\text{rel}} = 10^{-2}$ при пренареждане на неизвестните по метода на рекурсивна бисекция и сравнение с ускорението при гаусовият солвър от MKL за кръгла област.

Анализ на грешката за HSS-базирания солвър

В Таблица 3.2 са представени относителните грешки R_{relative} на решението получено по метода на HSS компресията. Като следваме възприетия в дисертацията подход, решението получено по метода на Гаус (LU факторизация реализирана в MKL солвъра) се приема за референтно и се прилага формула (2.5). Анализът показва, че относителната грешка за кръглата област има сходно поведение на това при дробно дифузионната задача в квадратна област. Грешката R_{relative} е близка по порядък до зададения относителен праг ε_{rel}, който може да се разглежда като априорна оценка за точността на решението. При оригиналното нареждане относителната грешка R_{relative} е най-малка. Това може да се обясни с по-малката ефективност на HSS компресията, което от друга страна означава по-малка загуба на информация. При кръглата област няма особеност на поведението на относителната грешка при пренареждане

„top“, каквато отбелязахме при квадратната област (виж Таблица 3.1). Относителната грешка R_{relative} се увеличава при увеличаване на размерността на системата n за повечето експерименти и пренареждания.

Таблица 3.2: Относителна грешка за кръгла област.

(а) Без пренареждане и пренареждане „top“.

	Без пренареждане				„top“			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2051	0.171	7.888e−05	7.06e-07	1.176e-08	0.113	0.000246	3.414e-06	2.375e-08
4241	0.0226	3.038e−04	3.666e-06	1.589e-08	0.17	0.000572	2.06e-06	2.126e-08
8685	2.492	7.308e−04	6.773e-06	5.318e-08	0.308	0.00125	5.729e-06	4.802e-08
13101	0.062	6.166e−04	7.861e-06	5.467e-08	0.317	0.000994	3.911e-06	8.789e-08
16110	0.399	8.876e−04	1.379e-05	6.756e-08	0.350	0.0012	5.257e-06	7.007e-08
24589	8.929	7.833e−02	2.184e-05	1.159e-07	0.406	0.0112	1.128e-05	5.333e-07
31891	0.244	1.035e−03	2.362e-05	1.966e-07	0.533	0.00192	7.519e-06	1.565e-07

(б) Пренареждания „snake“ и „stripes“.

	„snake“				„stripes“			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2051	0.0744	0.000197	1.12e-06	1.244e-08	0.122	0.00042	2.86e-06	2.36e-08
4241	0.166	0.000367	3.176e-06	3.201e-08	0.238	0.000652	3.681e-06	2.513e-08
8685	0.357	0.00235	8.347e-06	5.148e-08	0.328	0.00103	6.586e-06	8.667e-08
13101	0.399	0.00139	8.34e-06	4.943e-08	0.38	0.00172	5.816e-06	6.181e-08
16110	0.464	0.00313	1.096e-05	9.648e-08	0.428	0.00139	5.267e-06	9.568e-08
24589	0.525	0.0107	2.46e-05	2.244e-06	0.359	0.0077	1.376e-05	6.558e-07
31891	0.578	0.00484	1.649e-05	1.668e-07	0.532	0.00189	1.287e-05	9.514e-08

(в) Вложени сечение и рекурсивна бисекция.

	Вложени сечения				Рекурсивна бисекция			
n	$R_{relative}$ за r_{tol} 10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2051	0.0417	0.000374	2.168e-06	2.823e-08	0.0693	0.000303	2.692e-06	1.592e-08
4241	0.0981	0.000482	3.485e-06	4.187e-08	0.092	0.000791	3.134e-06	4.693e-08
8685	0.15	0.00137	7.718e-06	1.101e-07	0.192	0.00221	5.094e-06	9.908e-08
13101	0.146	0.00161	7.931e-06	1.29e-07	0.177	0.00282	7.456e-06	1.221e-07
16110	0.216	0.00288	7.564e-06	1.612e-07	0.23	0.00216	1.192e-05	1.41e-07
24589	0.302	0.00735	1.187e-05	4.681e-07	0.31	0.00469	1.789e-05	3.125e-07
31891	0.321	0.00522	1.515e-05	2.454e-07	0.354	0.00437	1.474e-05	3.24e-07

3.4 Заключителни бележки

В тази глава са анализирани числени методи и алгоритми за решаване на системи линейни алгебрични уравнения с плътни матрици получени при дискретизация по метода на крайните елементи на гранични задачи за *дробен* лапласиан описващ *аномална* дифузия в ограничена двумерна област $\Omega \subset \mathbb{R}^2$. Разгледаните блочни методи показват добро бързодействие.

Основно място в представените резултати заема изследването на изчислителната ефективност на йерархичния метод базиран на йерархична полусепарабелна компресия (HSS компресия) и ULV-подобна факторизация. Експерименталният сравнителен анализ се базира на реализацията на HSS компресия и ULV-подобна факторизация в софтуерния пакет STRUMPACK. Анализът показва добро бързодействие на последователния алгоритъм в сравнение с прекия гаусов солвър използващ блочна LU факторизация. В същото време, получените паралелни производителности и ускорения при използване на пакета STRUMPACK са по-малки, което се обяснява с по-сложната йерархична и рекурсивна структура на компресията.

Точността и изчислителната ефективност на HSS компресията съществено зависят от праговете на относителна грешка ε_{rel} и абсолютна грешка ε_{abs}, както и от наличието на подходяща *структура* на матрицата. Праговете се задават от потребителя и при експериментите се наблюдава относителна грешка R_{relative} на численото решение близка до ε_{rel}. За подобряване на *структурата* на матрицата са предложени пет начина за пренареждане на неизвестните, които увеличават съществено ефективността на компресията. Представеният анализ показва, че при повечето експерименти ефективността на HSS компресията е най-добра при пренареждане с рекурсивната бисекция.

Анализът показа, че *структурата* на матрицата получена при дискретизация на дробно дифузионната задача е по-малко подходяща за HSS компресия, в сравнение със задачата разгледана в Глава 2. Това може да се обясни с факта, че *дробният* лапласиан е силно нелокален. За съответната матрица това води до по-бавно намаляване по абсолютна стойност на извъндиагоналните елементи. Предложените пренареждания на неизвестните значително подобряват ефективността на STRUMPACK. Въпреки това паралелното бързодействие на HSS компресията отстъпва на прекия солвър използващ блочна LU факторизация.

Важен резултат от изследванията в тази глава е анализът на изчислителната сложност и паралелната ефективност на отделните части на блочния алгоритъм реализиран в пакета STRUMPACK. Това води до извода, че едно от предимствата на HSS компресия е, че при решаване на серия от системи линейни алгебрични уравнения, при които матрицата не се изменя пониската изчислителна сложност на решаване с факторизирана матрица след HSS компресия и ULV-подобна факторизация $O(nr)$ ще има предимство пред решаването с факторизирана матрица след LU факторизация – $O(n^2)$. Такъв например е случаят на параболична задача, при дискретизацията на която се използва матрица на масата с диагонална концентрация(lumped mass matrix). Това е съществено преимущество в сравнение с реализирания в пакета MKL метод на гаусовата елиминация. Такава задача е разгледана в Глава 4.

Глава 4

Метод на крайните елементи за решаване на двумерна параболична задача за дробна дифузия

В тази глава ще разгледаме параболична задача за *аномална* дифузия описана с *дробен* оператор на Лаплас. В Глава 3 беше разгледана двумерната елиптична (стационарна) гранична задача с дробен лапласиан. Тук ще използваме въведените в предходната глава понятия и означения, свързани с представения в предходната глава метод на крайните елементи (МКЕ) за числено решаване на стационарната задача. Това в частност се отнася за формулите, изчислителните процедури и софтуерни средства за получаване на матрицата на коравина, участваща и в дискретизацията на параболичната задача.

Анализът на числените експерименти в Глава 3 беше насочен към изследване на възможностите за ефективно използване на солвъра базиран на йерархичната полусепарабелна компресия. Показана бе необходимостта от пренареждане на неизвестните. В този контекст, акцент на анализа в тази глава ще бъдат специфики на алгоритмичната реализация на метода на Ойлер за числено решаване на разглежданата параболична задача.

Водещ интерес за изследванията в тази глава представлява решаването на системи с факторизираната матрица при прилагане на йерархичния метод. Тази стъпка, след HSS компресия и ULV-подобна факторизация, има изчислителната сложност $O(nr)$ [17, 86]. За сравнение при използването на LU факторизация (гаусова елиминация), решаването на системи с факторизираната матрица изисква $O(n^2)$ аритметични операции. При стационарната задача та-

зи стъпка се изпълнява само веднъж и почти не влияе на бързодействието на солвърите. Това се променя, когато числениият метод включва решаване на последователност от системи линейни алгебрични уравнения с една и съща матрица. В този случай относителната тежест на времето за решаване на системи с факторизираната матрица се увеличава съществено. Такъв е разглежданият в настоящата глава метод за решаване на параболична задача с *дробна* дифузия.

За дискретизация по пространството ще използваме метода на крайните елементи, предложен в [3]. MatLab програмата публикувана от същите автори в [4] ще бъде използвана за генериране на матрицата на коравина и дясната част. За генериране на матрицата на масата е разработен алгоритъм и софтуерен модул, който използва информацията за геометрията на триангулацията $\mathcal{T} \in \Omega$, включваща координати на възлите и топология на мрежата. За дискретизация по времето ще използваме неявна диференчна схема на Ойлер с постоянна стъпка и диагонална концентрация на матрицата на масата (lumped mass matrix).

В статия [79], Вабишчевич анализира предложения метод за числено решаване на параболична задача за *аномална* дифузия при спектрална дефиниция на *дробния* оператор на Лаплас. За числените експерименти в настоящата глава използваме аналог на тестовия пример от статията на Вабишчевич. В частност, това дава възможност да направим сравнение на числените резултати съответстващи на двете различни дефиниции на дробния лапласиан.

Основните резултати представени в тази глава са включени в следната приета за печат статия:

[75] D. Slavchev and S. Margenov. Performance study of hierarchical semi-separable compression solver for parabolic problems with space-fractional diffusion. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, Cham, in press. Springer International Publishing

4.1 Постановка на задачата

Както отбелязахме в Глава 3, съществуват различни дефиниции на дробен лапласиан. И тук ще използваме интегралното представяне (3.3)

$$(-\Delta)^{\alpha} u(x) = C(d,\alpha)\ \text{P.V.} \int_{\mathbb{R}^n} \frac{u(x)-u(y)}{|x-y|^{d+2\alpha}}, \qquad \text{(3.1 копие)}$$

където $\alpha \in [0,1]$ е степента на *дробния* оператор на Лаплас Δ, d е пространствената размерност на задачата (за разглежданата в тази глава двумерна

задачи $d = 2$), с P.V. е означена главната стойност, а $C(d, \alpha)$ е нормализиращата константа

$$C\left(d, \alpha\right) = \frac{2^{2\alpha}\alpha\Gamma\left(\alpha + \frac{d}{2}\right)}{\pi^{d/2}\Gamma\left(1 - \alpha\right)}.$$

Тук разглеждаме следната параболична задача за неизвестната функция $u(x,t)$, $(x,t) \in \Omega \times [0,T]$

$$\left|\begin{array}{lll} \dfrac{\partial u(x,t)}{\partial t} + (-\Delta)^{\alpha} u(x,t) = f(x,t), & x \in \Omega, & t \in [0,T], \\ u(x,t) = 0, & x \in \Omega^c, & t \in [0,T], \\ u(x,0) = u^0(x), & x \in \Omega. & \end{array}\right.$$

Тук Ω^c е допълнението на ограничената област Ω, а $[0,T]$ е времевият интервал. Хомогенните гранични условия на Дирихле се налагат както при стационарната задача разгледана в предходната глава.

Нека $\mathcal{T} \in \Omega$ е допустима триангулация на изчислителната област. За дискретизация по пространството прилагаме МКЕ с линейни крайни елементи дефинирани върху $\mathcal{T}$. Нека означим с $\mathbb{V}$ крайноелементното пространство от непрекъснати на части линейни функции. Лагранжевият базис $\{\varphi_1, \ldots, \varphi_N\} \in \mathbb{N}$ съответства на вътрешните възли $\{x_1, \ldots, x_N\}$, като $\varphi_i\left(x_j\right) = \delta_j^i$. Така получаваме задачата на Коши

$$M_L \frac{d\mathbf{u}}{dt} + K\mathbf{u} = M_L\mathbf{f}, \quad 0 < t \leq T, \quad \mathbf{u}(0) = \mathbf{u}^0,$$

за неизвестните функции $\mathbf{u} = (u_j(t)) \in \mathbb{R}^N$, $t \in [0,T]$ и дясна част $\mathbf{f} = (f_j(t)) \in \mathbb{R}^N$. Тук $K = K_{ij} \in \mathbb{R}^{N\times N}$ е матрицата на коравина, съответстваща на *дробния* лапласиан. Тя има вида дефиниран в Глава 3 виж уравнения (3.8) и (3.10). С $M_L = \mathrm{diag}\left(m_L^i\right) \in \mathbb{R}^N$ е означена матрицата на масата с диагонална концентрация (lumped mass matrix), където m_L^i е концентрираната маса във възела x_i. Алгоритъмът и програмният модул за изчисляване на матрицата M_L са представени в Приложение Б.

За дискретизация по времето използваме неявния метод на Ойлер, която в общия случай има вида

$$M_L \frac{\mathbf{u}^{j+1} - \mathbf{u}^j}{\tau_j} + K\mathbf{u}^{j+1} = M_L \frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2}, \quad j = 0, \ldots, m-1, \tag{4.1}$$

където m е броят на стъпките по времето, $\sum_{j=0}^{m-1} \tau_j = T$, $t_0 = 0$, $t_{j+1} = t_j + \tau_j$ и $\mathbf{u}^j = \mathbf{u}(t_j)$, $\mathbf{f}^j = \mathbf{f}(t_j)$.

Методът на Ойлер е безусловно устойчив, което е особено важно за задачи с дробна дифузия, при които решението има по-ниска регулярност за по-малки

стойности на дробната степен α. Това е отбелязано също така от Вабишчевич в [79].

В настоящата дисертация ще се ограничим до случая на постоянна стъпка по времето $\tau_j = \tau$. При това условие реализацията на всяка стъпка по времето на (4.1) се свежда до решаване на системата линейни алгебрични уравнения

$$\tilde{K}\mathbf{u}^{j+1} = \tilde{\mathbf{f}}^j, \tag{4.2}$$

където

$$\tilde{K} = \frac{M_L}{\tau} + K, \quad \tilde{\mathbf{f}}^j = M_L \left(\frac{\mathbf{f}^{j+1} + \mathbf{f}^j}{2} + \frac{\mathbf{u}^j}{\tau} \right).$$

Матрицата на коравина и матрицата на масата са симетрични и положително определени, и следователно (4.2) има единствено решение. Матрицата $\tilde{K}$ не зависи от j, т.е. тя е една и съща за всички стъпки по времето. Това означава, че при реализацията на метода на Ойлер факторизацията се изпълнява един път, след което решаваме m системи с факторизираната матрица.

При използване на Гаусова елиминация изчислителната сложност на LU факторизацията е $O\left(n^3\right)$, като след това реализацията m на брой стъпки по времето изисква още $O\left(n^2 m\right)$ аритметични операции.

При йерархичния метод – HSS компресията и ULV-подобната факторизация имат обща изчислителна сложност $O(n^2 r)$, като след това за решаване на системите с факторизираната матрица са необходими още $O(nrm)$ аритметични операции. Така, определящ за ефективността на йерархичния метод е извъндиагоналният ранг r. Известно е, че при матрици с подходяща *структура*, рангът r може да е значително по-малък от броя на неизвестни n (виж например [82]).

Ще припомним, че ефективността на йерархичната полусепарабелна компресия зависи от типа на задачата и от подреждането на неизвестните. В Глава 3 бяха предложени и анализирани няколко метода за пренареждане на неизвестните за стационарната задача с дробна дифузия и съответните системи с матрицата на коравина K. Модифицираната матрица $\tilde{K}$ се различава от K единствено по добавените положителни елементи по диагонала. В същото време HSS компресията не променя диагоналните коефициенти. Това ни дава основание да използваме анализа на методите за пренареждане на неизвестните от предходната Глава 3. Така тук ще се ограничим до анализ на числени резултати при най-ефективното пренареждане – рекурсивната бисекция.

За числените експерименти ще използваме аналог на параболичната задача от статията на Вабишчевич [79], в която се използва спектралната дефиниция на *дробния* лапласиан. Задачата се решава за $(x,t) \in \Omega \times [0,T] = (-1,1)^2 \times (0,0.1)$. Решението се определя от независеща от времето дясна част

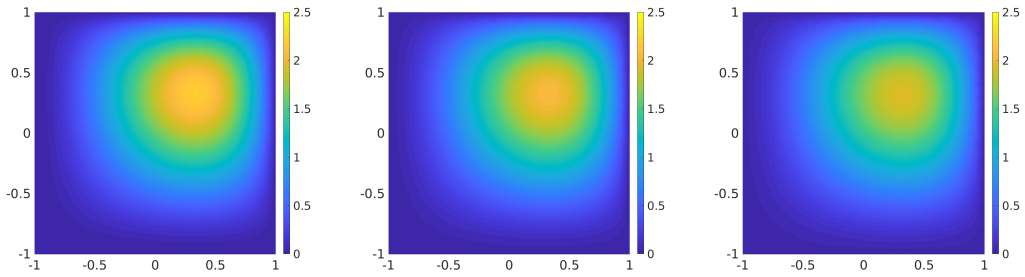
$$f(x) = \frac{(x_1+1)(x_2+1)}{4},$$

и начално условие

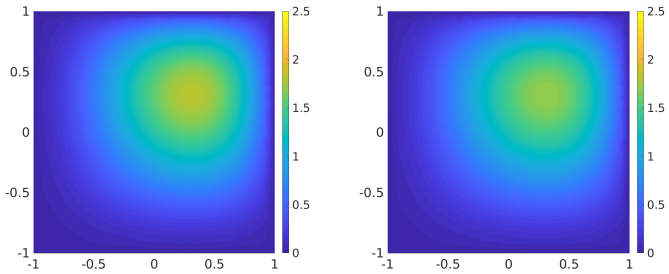
$$u^0(x) = 100 \left(\frac{x_1+1}{2}\right)^2 \left(1 - \frac{x_1+1}{2}\right) \left(\frac{x_2+1}{2}\right)^2 \left(1 - \frac{x_2+1}{2}\right)$$

За дискретизация по пространството са използвани триангулациите от Глава 3, при стъпка по времето $\tau = T/m$, $m = 256$.

На Фигура 4.1 са показани началното условие $u^0(x)$ и числените решения при $\alpha = 0.5$ за $t \in \{0.025, 0.05, 0.075, 0.1\}$. Получените резултати са качествено подобни на резултатите, представени в [79].



(а) $t = 0$ (б) $t = 0.025$ (в) $t = 0.05$



(г) $t = 0.075$ (д) $t = 0.1$

Фигура 4.1: Числени решения на моделната параболична задача с дробна дифузия със степен $\alpha = 0.5$: МКЕ по пространството и неявен метод на Ойлер по времето.

4.2 Анализ на числени експерименти върху компютърни системи с обща памет

Анализираните в тази глава числени резултати са получени върху компютърни системи с обща памет. Както и в предходните глави, този тип експерименти

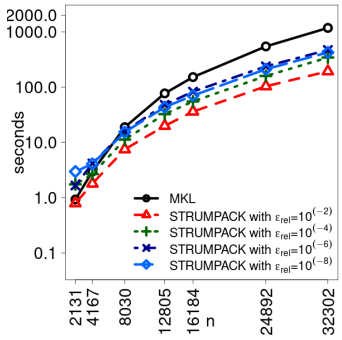
са проведени на суперкомпютър AVITOHOL [1]. Използвани са общо 16 ядра от два Intel Xeon E5-2650v2 8C 2.6GHz CPU процесора интегрирани в един възел. Тези процесори позволяват хипертрединг с по две нишки на ядро, но тъй като това не води до подобряване на ефективността на анализираните методи, такива експерименти не са включени в представения анализ. Програмите за решаване на системите линейни алгебрични уравнения и включените в тях софтуерни пакети са компилирани с Intel Compiler Collection и MKL с версия 2017.2.174. Пакетът STRUMPACK е версия 3.2.

За генериране на матрицата на коравина K и дясната част $\mathbf{f}$ е използвана програма на MatLab, която е публикувана в [4]. За пренареждане на неизвестните по метода на рекурсивната бисекция е използван алгоритъмът описан в Раздел 3.2.5 и разработеният код, представен в Приложение А.4. За изчисляване на матрицата на масата с диагонална концентрация M_L е реализиран алгоритъмът представен в Приложение Б. Получените по този начин данни се записват в текстови файлове. За решаване на разглежданата задача е разработена програма на езика C, която зарежда данните, изпълнява пренареждането и извиква съответните солвъри от използваните софтуерни пакети.

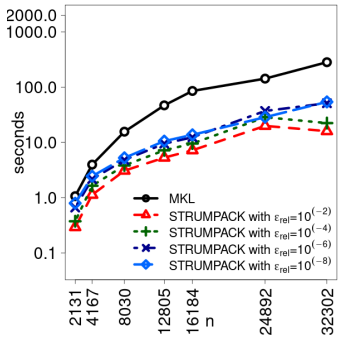
4.2.1 Последователни и паралелни експерименти.

На Фигура 4.2 са представени: времената за изпълнение на йерархичната полусепарабелна компресия и ULV-подобна факторизация при използване на софтуерния пакет STRUMPACK и на LU факторизацията при използване на MKL – (Фигури 4.2а и 4.2г); общото време за решаване на системите с факторизираните матрици на всяка от стъпките по времето в метода на Ойлер – (Фигури 4.2б и 4.2д); и цялото време за решаване на задачата – (Фигури 4.2в и 4.2е). В почти всички случаи резултатите от числените експерименти показват по-добрата ефективност на йерархичния метод, като тенденцията е тя да се повишава с увеличаване на броя на неизвестните n. За най-голямата система ($n = 32\ 302$) времената за решаване на цялата задача с прилагане на HSS компресия от пакета STRUMPACK са между $\sim$2.5 и $\sim$5 пъти по-добри отколкото при използване на LU факторизация от пакета MKL. Тези резултати потвърждават теоретичните очаквания за по-силно подобряване на ефективността на йерархичния солвър при параболичната задача, в сравнение със стационарната задача, разгледана в предходната глава.

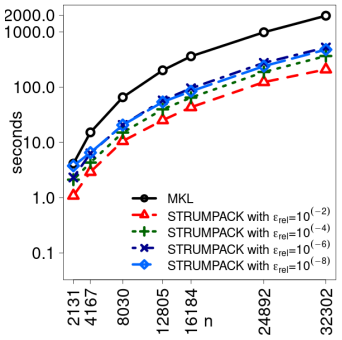
На Фигура 4.3 са показани паралелните ускорения при решаване на параболичната задача с прилагане на солвъра от пакета STRUMPACK, използващ HSS компресия. При паралелните експерименти с 16 нишки получаваме ускорения от $\sim$3 (за $n = 2\ 131$) до $\sim$8 (за $n = 32\ 302$). Ще припомним, че при блочната LU факторизация се достига до почти оптимално ускорение $\sim$15.



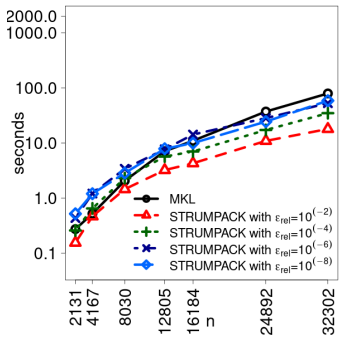
(а) Последователни времена: компресия и факторизация



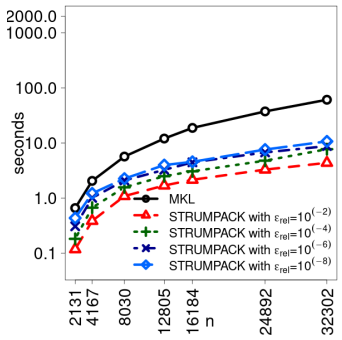
(б) Последователни времена: стъпки по времето



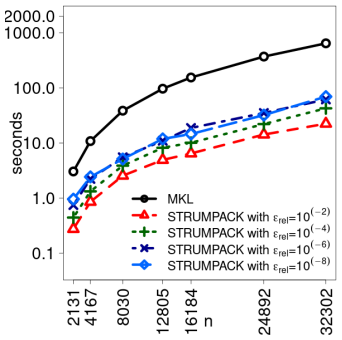
(в) Последователни времена за цялата задача



(г) Паралелни времена при 16 нишки: компресия и факторизация



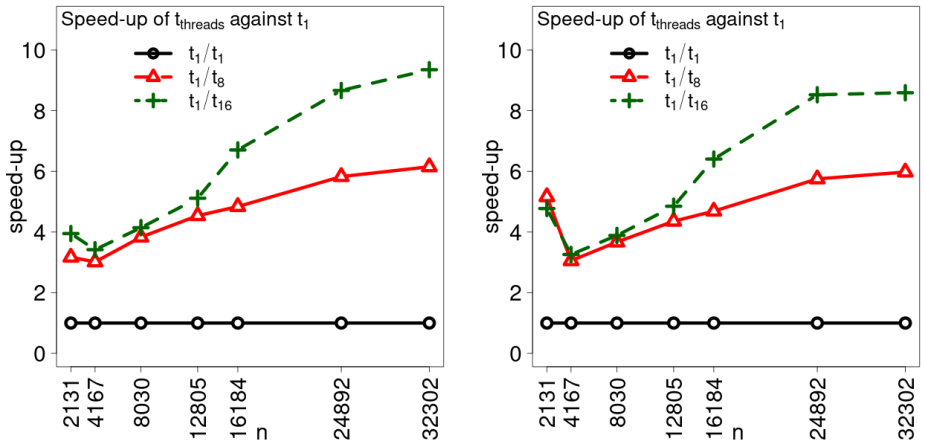
(д) Паралелни времена при 16 нишки: стъпки по времето



(е) Паралелни времена при 16 нишки за цялата задача

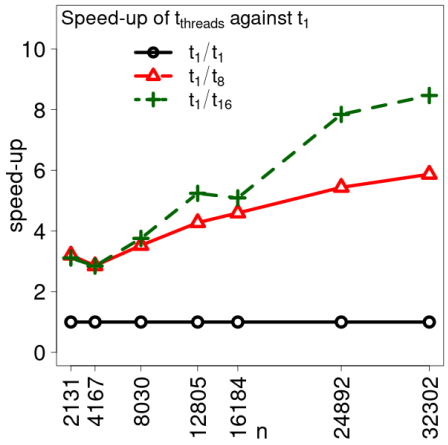
Фигура 4.2: Сравнение между времената за решаване на параболичната задача (4.2) при използване на MKL и STRUMPACK при праг на относителната грешка $\varepsilon_{\text{rel}} \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$.

По-ниското паралелно ускорение на йерархичния метод може да се обясни с по-сложната йерархична и рекурсивна структура на алгоритъма за компресия. Паралелната реализация на решаването на системи с HSS факторизираната матрица също има по-сложна (и по-малко балансирана) структура от блочната LU факторизация.

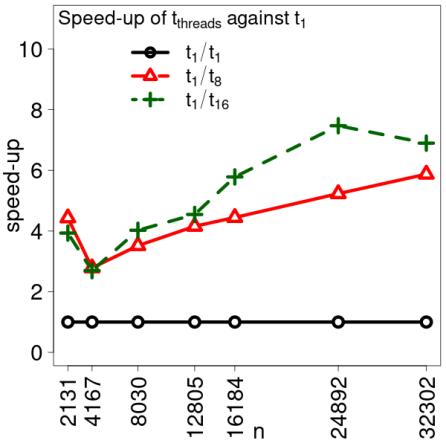


(а) $\varepsilon_{\text{rel}} = 10^{-2}$

(б) $\varepsilon_{\text{rel}} = 10^{-4}$



(в) $\varepsilon_{\text{rel}} = 10^{-6}$



(г) $\varepsilon_{\text{rel}} = 10^{-8}$

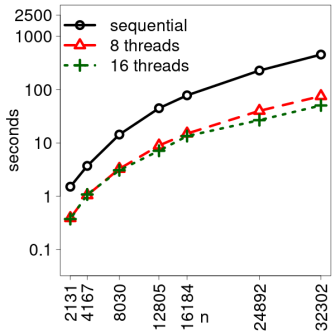
Фигура 4.3: Паралелни ускорения при решаване на параболичната задача с прилагане на HSS базирания солвър за $m = 256$ стъпки по времето.

4.2.2 Сравнение на времената и паралелните ускорения на отделните части на йерархичния метод

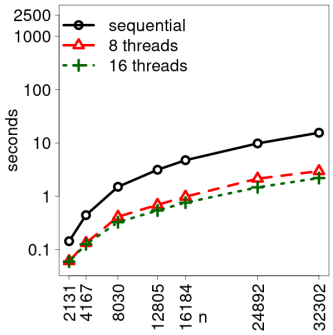
Паралелните времена и ускорения на отделните части от йерархичния метод на базата на HSS компресия за $\varepsilon_{\text{rel}} = 10^{-6}$ са представени на Фигура 4.4. Паралелните ускорение за другите стойности на относителният праг не се различават съществено. Компресията (Фигура 4.4г) и факторизацията (Фигура 4.4д) имат подобни паралелни ускорения, които постепенно се увеличават с увеличаване на размера на задачата, като варират между $\sim$2 и $\sim$6. Изпълнението на метода на Ойлер за дискретизация по времето (включващ решаването на m системи с факторизираната компресирана матрица) (Фигура 4.4е)

има по-ниско паралелно ускорение, вариращо между $\sim$2 и $\sim$6. Определящо за ефективностите на целия алгоритъм обаче е, че времето за изпълнението на $m = 256$ стъпки по времето е по-малко от времето за HSS компресията и ULV-подобната факторизация.

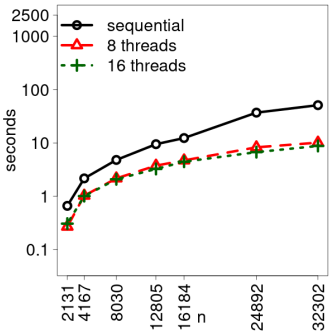
Важно е да отбележим също така, че времето за решаване на системите с факторизираната матрица със STRUMPACK е доста по-добро в сравнение с времето за решаване на тези системи с LU факторизираната матрица с MKL (виж Фигура 4.2б,д). Това потвърждава съответните оценки на изчислителната сложност. Така, на всяка стъпка по времето след HSS компресия и ULV-подобна факторизация се изпълняват $O(nr)$ аритметични операции, докато при LU факторизацията тази сложност е $O(n^2)$.



(а) Компресия

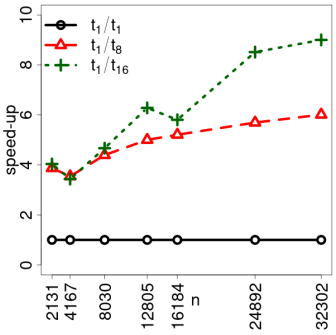


(б) Факторизация

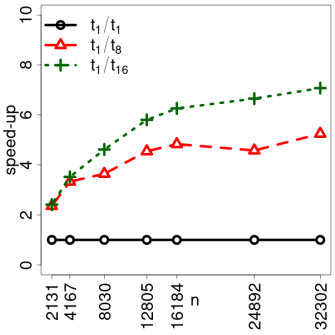


(в) Изпълнение на $m = 256$ стъпки по времето

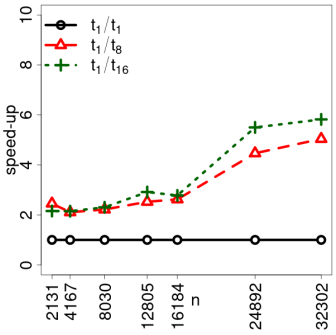
Последователно и паралелни времена



(г) Компресия



(д) Факторизация



(е) Изпълнение на $m = 256$ стъпки по времето

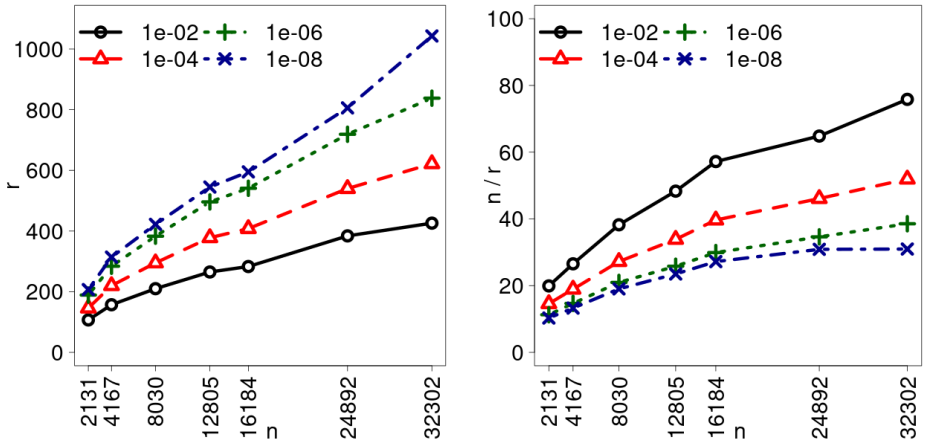
Паралелни ускорения

Фигура 4.4: Времена и паралелни ускорения на стъпките от йерархичния солвър от STRUMPACK при решаване на параболичната задача с праг на относителната грешка $\varepsilon_{rel} = 10^{-6}$.

4.2.3 Извъндиагонален ранг

Изчисленият в процеса на HSS компресията максимален извъндиагонален ранг r е представен на Фигура 4.5а, като на Фигура 4.5б е показано отношението n/r. Както беше отбелязано в предходните глави, рангът е мярка за ефективността на компресията, като също така е определящ в оценките за изчислителната сложност. За разглежданата задача r има значително по-малки стойности от n. Компресията е най-силна, т.е. отношението r/n е най-малко, при най-голяма стойност на прага на относителна грешка. Така, при $\varepsilon_{\text{rel}} = 10^{-2}$ рангът r е между $\sim$20 и $\sim$80 пъти по-малък от n, докато при най-финия праг $\varepsilon_{\text{rel}} = 10^{-8}$ това отношение е между $\sim$10 и $\sim$30. Този анализ

показва, че пренареденатa с рекурсивна бисекция матрица $\tilde{K}$ има подходяща структура за прилагане на HSS компресия. Това се потвърждава от числените експерименти, показващи предимство на йерархичния метод в сравнение с гаусовата елиминация (блочната LU факторизация). Тук е важно да припомним, че компресията е *приблизителна*. Така, по-високата ефективност на компресията, която получаваме при по-големи стойности на прага ε_{rel} е за сметка на по-малка точност на решението. Това води до необходимостта от анализ на относителната грешка на числено решение на системата линейни алгебрични уравнения. Такъв анализ е представен в следващия раздел.



(а) Максимален извънднагонален ранг r (б) Отношение на r и броя на неизвестните n

Фигура 4.5: Визуализация на максималния извънднагонален ранг r и отношение n/r.

4.2.4 Анализ на грешката за HSS-базирания солвър

Матрицата H получена след HSS компресия е приближение на $\tilde{K}$. Както при стационарната задача разгледана в Глава 3, ще анализираме каква е точността на STRUMPACK солвъра, съответстващ на това приближение. За по-добра съпоставимост на резултатите отново ще анализираме относителната грешка (2.5)

$$R_{\text{relative}} = \frac{\left\|x^{\text{Gauss}} - x^{\text{HSS}}\right\|_{l_2}}{\left\|x^{\text{Gauss}}\right\|_{l_2}} = \frac{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}} - x_i^{\text{HSS}})^2}}{\sqrt{\sum_{i=1}^{n}(x_i^{\text{Gauss}})^2}}, \qquad \text{(2.5 копие)}$$

като мярка за точността на метода. В тази формула с x^{Gauss} е означено решението получено по метода на Гаус с използване на блочния LU солвър от

пакета MKL. Приемаме това решение за референтно. Решението получено с помощта на йерархична полусепарабелна компресия и съответния солвър от пакета STRUMPACK е означено с x^{HSS}.

Получените относителни грешки R_{relative} за избраните 4 стойности на времето t са представени в Таблица 4.1, както следва: за $t = 0.025,\ 0.05$ – Подтаблица а) и за $t = 0.075,\ 0.1$ – Подтаблица б. Както и при стационарната задача, относителната грешка е близка по стойност до зададения относителен праг ε_{rel}. Представените числени резултати показват също така, че относителната грешка не нараства съществено с увеличаване на времевия интервал. Това потвърждава устойчивостта на неявния метод на Ойлер.

Таблица 4.1: Относителна грешка на HSS базираният солвър.

(а) $t = 0.025$ и $t = 0.05$

n	Относителна грешка в $t = 0.025$				Относителна грешка в $t = 0.05$			
	$R_{relative}$ за r_{tol}				$R_{relative}$ за r_{tol}			
	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2131	0.00385	$7.52e{-05}$	$2.78e{-07}$	$5.53e{-08}$	0.00659	0.000124	$4.62e{-07}$	$9.72e{-08}$
4167	0.006	0.00013	$6.4e{-07}$	$8.62e{-08}$	0.01	0.000225	$1.03e{-06}$	$1.36e{-07}$
8030	0.0083	0.00023	$1.28e{-06}$	$2.17e{-07}$	0.0146	0.000375	$2.e-06$	$3.49e{-07}$
12805	0.0106	0.00028	$1.68e{-06}$	$4.77e{-07}$	0.0178	0.000484	$2.49e{-06}$	$7.56e{-07}$
16184	0.0126	0.0003	$1.75e{-06}$	$5.16e{-07}$	0.0227	0.00052	$2.77e{-06}$	$7.88e{-07}$
24892	0.0192	0.000393	$2.5e{-06}$	$9.69e{-07}$	0.0349	0.000619	$3.97e{-06}$	$1.49e{-06}$
32302	0.0234	0.000345	$2.48e{-06}$	$1.18e{-06}$	0.0437	0.000537	$3.97e{-06}$	$1.76e{-06}$

(б) $t = 0.075$ и $t = 0.1$

n	Относителна грешка в $t = 0.075$				Относителна грешка в $t = 0.1$			
	$R_{relative}$ за r_{tol}				$R_{relative}$ за r_{tol}			
	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-2}	10^{-4}	10^{-6}	10^{-8}
2131	0.0088	0.000159	$5.98e{-07}$	$1.32e{-07}$	0.0108	0.000188	$7.07e{-07}$	$1.62e{-07}$
4167	0.0135	0.000298	$1.31e{-06}$	$1.707{-07}$	0.0166	0.000361	$1.55e{-06}$	$1.98e{-07}$
8030	0.0206	0.000498	$2.52e{-06}$	$4.55e{-07}$	0.0264	0.000606	$2.93e{-06}$	$5.49e{-07}$
12805	0.0242	0.000653	$3.09e{-06}$	$9.74e{-07}$	0.0301	0.0008	$3.6e{-06}$	$1.16e{-06}$
16184	0.0324	0.000706	$3.6e{-06}$	$1.01e{-06}$	0.0419	0.000875	$4.34e{-06}$	$1.2e-06$
24892	0.0499	0.000807	$5.19e{-06}$	$1.92e{-06}$	0.0646	0.000973	$6.26e{-06}$	$2.3e-06$
32302	0.0636	0.000693	$5.25e{-06}$	$2.21e{-06}$	0.0832	0.000826	$6.41e{-06}$	$2.58e{-06}$

4.3 Заключителни бележки

В тази глава са анализирани методи и алгоритми за числено решаване на параболични задачи с *аномална* дифузия описана с *дробен* лапласиан по пространството в ограничена двумерна област $\Omega \subset \mathbb{R}^2$. За дискретизация е приложен МКЕ в комбинация с неявния метод на Ойлер с постоянна стъпка по времето. Така задачата се свежда до решаване на последователност от системи линейни алгебрични уравнения с една и съща плътна матрица. Числените резултати

потвърждава доброто бързодействие на анализираните блочни методи и алгоритми и техните софтуерни реализации.

Водеща тема в представените резултати е анализът на изчислителната ефективност на метода базиран на йерархична полусепарабелна компресия и ULV-подобна факторизация и неговата паралелна реализация в софтуерния пакет STRUMPACK. Специфика на приложения неявния метод на Ойлер с постоянна стъпка τ е, че числено решаване на параболичната задача се свежда до $m = T/\tau$ системи линейни алгебрични уравнения с една и съща матрица $\tilde{K}$ и променящи се на всяка стъпка по времето десни части. Това означава, че матрицата $\tilde{K}$ се факторизира еднократно. Така акцентът пада върху решаването на m системи с известна факторизирана матрица. Ще припомним, че при стационарната задача тази стъпка на алгоритъма отнема много малка част от общото време, което се дължи на относително по-малката ѝ изчислителна сложност – $O(n^2)$ за гаусовия солвър и $O(nr)$ за йерархичния солвър използващ HSS компресия. Представеният анализ показва, че за разглежданата задача рангът r е съществено по-малък от броя на неизвестните n, което обуславя преимуществото на йерархичния метод. В резултат на това, както последователните, така и паралелните времена за решаване на параболичната задача с използване на солвъра от софтуерния пакет STRUMPACK са съществено по-добри в сравнение с времената при използване на блочна LU факторизация от пакета MKL.

Полученото с йерархичния солвър решение е *приблизително*, което налага анализ на грешката. За целта се разглежда относителната грешка (2.5), където решението получено от прекия гаусов солвър се приема за референтно. За контрол на процеса на компресия, в пакета STRUMPACK се задават на два параметъра – за относителен и за абсолютен праг. В представените експерименти абсолютният праг ε_{abs} е фиксиран на 10^{-8}, като са анализирани числените резултати, получени при вариране на относителния праг $\varepsilon_{\text{rel}} \in \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$. Анализът показва, че относителната грешка R_{relative} е близка до зададения относителен праг, като нараства устойчиво с развитие на процеса по времето. Това потвърждава, че йерархичният метод осигурява добра точност на решението при подходящо избран праг ε_{rel}.

Основен резултат от изследванията в тази глава е анализът на изчислителната сложност и съответните последователни и паралелни времена. Разгледаният вариант на неявен метод на Ойлер свежда задачата до решаване на последователност от системи линейни алгебрични уравнения, при които матрицата не се изменя, като представените резултати показват съществено преимущество на алгоритъма използващ йерархична полусепарабелна компресия и неговата реализация в паралелния пакет STRUMPACK.

Важно е да подчертаем още веднъж, че представеният в тази глава анализ

се отнася за параболични задачи с дробна дифузия, при които за дискретизация по времето прилагаме неявен метод на Ойлер с *постоянна* стъпка τ. Ако стъпката τ_j е неравномерна, т.е. $\tau_{j_1} \neq \tau_{j_2} : \ j_1 \neq j_2 \in [1, m]$, диагоналните коефициенти на матрицата $\tilde{K} = K + 1/\tau_j M_L$ ще се променят. Такава задача с адаптивна стъпка по времето за трансфер на топлина и маса при вакумно замразяване може да се намери в [32]. В този случай, на всяка стъпка по времето ще е необходимо да изпълняваме стъпката факторизация. Това се отнася, както за метода на Гаус използващ блочна LU факторизация, така и за метода на базата на HSS компресия. Съществено обаче е, че при йерархичния метод е достатъчно да направим компресията само веднъж, което се дължи на факта, че измененията на $\tilde{K}$ са само в диагоналните елементи. Това дава основание да очакваме, че йерархичният солвър може да има още по-съществено предимство при адаптивна стъпка по времето, което е перспективно направление за бъдещи изследвания.

Заключение

В дисертацията е анализирана изчислителната ефективност на блочни числени методи и алгоритми за решаване на системи линейни алгебрични уравнения с плътни матрици. Мотивация на това изследване са приложения свързани с числено решаване на елиптични и параболични частни диференциални уравнения. Две такива задачи са използвани в представения сравнителен анализ: а) гранична задача описваща обтичане на крилни профили на Жуковски, дискретизирана с метод на граничните елементи; б) аномална дифузия в ограничена област моделирана с *дробен* лапласиан, където за дискретизация е приложен метод на крайните елементи. И в двата случая непрекъснатите задачи се свеждат до системи линейни алгебрични уравнения с плътни матрици. Показано е, че *структурата* на тези матрици е подходяща за прилагане на йерархичен метод използващ HSS компресия.

Важна част от Глава 2 е сравнителният анализ на изчислителната ефективност на софтуерни пакета имплементиращи блочна LU факторизация, като вариант на гаусова елиминация. При числените експерименти върху компютърни системи с обща памет с процесори Intel Xeon е сравнено бързодействието на разработената от производителя високопроизводителна библиотека Math Kernel Library (MKL) и софтуерния пакет с отворен достъп Parallel Linear Algebra Software for Multicore Architectures (PLASMA). При използване на копроцесорите Intel Xeon Phi 7120P са сравнени MKL и Matrix Algebra on GPU and Multicore Architectures (MAGMA). Числените експерименти показват високата ефективност на разгледаните паралелни библиотеки за реални задачи с голяма размерност. Така например за решаване на система с 40 000 неизвестни изчислителното време се намалява от над един час (за последователните експерименти) до няколко минути при използване на Intel Xeon MIC. Общият извод е, че софтуерният пакет MKL има по-добро бързодействие от анализираните алтернативни паралелни реализации на блочна LU факторизация.

Основен акцент в дисертацията е анализът на възможността за подобряване на изчислителната ефективност на решаването на системи линейни алгебрични уравнения с плътни матрици с помощта на блочен йерархичен метод

използващ HSS компресия. Този метод е реализиран в софтуерния пакет със свободен достъп STRUMPACK. В Глави 2 и 3 е анализирана производителността на йерархичния алгоритъм за системи линейни алгебрични уравнения, получени при прилагане съответно на метод на граничните елементи и метод на крайните елементи за разгледаните елиптични гранични задачи. Анализът показва, че тези плътни матрици имат подходяща *структура* за прилагане на йерархичния метод. Това означава, че в процеса на HSS се получават извъндиагонални блокове с нисък ранг.

В резултат на HSS компресията се получава матрица, която апроксимира изходната. Точността на алгоритъма имплементиран в пакета STRUMPACK се контролира от два прага на грешката – абсолютен и относителен. Те се задават от потребителя. От тези параметри зависи ефективността на метода и точността на получените решения на системите линейни алгебрични уравнения. Анализирани са резултати при абсолютен праг $\varepsilon_{\text{abs}} = 10^{-2}$ и вариране на относителния праг $\varepsilon_{\text{rel}} = 10^{-2}$, 10^{-4}, 10^{-6} и 10^{-8}. За системата получена при дискретизация с метод на граничните елементи е разгледан и случая $\varepsilon_{\text{rel}} = 10^{-12}$.

Последователните експерименти потвърждават оценките на изчислителната сложност на анализираните блочни методи. И за двете задачи йерархичният солвър има по-добро бързодействие от гаусовия солвър от пакета MKL – най-ефективният от анализираните в настоящата работа софтуерни средства използващи LU факторизация. Базираният на HSS компресия солвър има по-сложна йерархична и рекурсивна структура. Това се вижда и от паралелните резултати. Йерархичната полусепарабелна компресия показва паралелни ускорения достигащи до $\sim$8, при използване на 16 нишки. Солвърите с гаусова елиминация, от своя страна достигат до почти оптимално паралелно ускорение от $\sim$15. Независимо от това, при някои експерименти паралелните времена на йерархичния солвър са по-добри от тези на директния солвър.

При прилагане на йерархичния метод се намира приближено решение на системата, като неговата точност зависи от това каква е точността на HSS компресията. На базата на проведените числени експерименти е анализирана относителната грешка R_{relative}, като решението получено с прекия гаусов солвър се приема за референтно. За задачата с дробна дифузия се наблюдават относителни грешки близки до зададения праг ε_{rel}. Това може да бъде прието като добра характеристика на алгоритъма за HSS компресия. За задачата за обтичане на крилни профили на Жуковски относителните грешки за съответните прагове ε_{rel} са по-големи, което обаче се компенсира от по-доброто бързодействие.

Известно е, че качеството на йерархичната полусепарабелна компресия силно зависи от *структурата* на плътната матрица. При дискретизация с

метода на граничните елементи възлите на мрежата са подредени последователно по крилните профили. Оказва се, че тази естествена номерация води до получаване на плътна матрица с подходяща *структура* за прилагане на HSS компресия. При двумерната дробно дифузионна задача, методът на крайните елементи използва номерация на възлите, получена от програмата за триангулация на изчислителната област `initmesh`. *Структурата* на така получената плътна матрица не е подходяща за HSS компресия. За нейното подобряване са предложени и пет метода за пренареждане. Представеният анализ показва преимущества на методите на вложените сечения и на рекурсивната бисекция.

В Глава 4 е изследвана изчислителната ефективност и точност на йерархичния солвър базиран на HSS компресия за параболична задача с *дробна* дифузия по пространството. За дискретизация по времето е използвана неявна диференчна схема на Ойлер с постоянна стъпка. При тази постановка на задачата намирането на числено решение се свежда до решаване на последователност от системи линейни алгебрични уравнения с една и съща матрица. Така на всяка стъпка по времето се решава система с матрицата на прехода, която е факторизирана еднократно. Решаването на такива системи с помощта на HSS компресия има по-ниска изчислителна сложност – $O(nr)$, в сравнение с използваната в метода на Гаус LU факторизация – $O(n^2)$. В допълнение, анализът на числените експерименти показва, че отношението n/r е по-малко. За разгледаната параболична задача времената на йерархичния солвър са подобри, както при последователните, така и при паралелните експерименти. В същото време, благодарение на безусловната устойчивост на неявния метод на Ойлер, относителната грешка на решението се запазва близка до зададения относителен праг ε_{rel}.

Списък на публикациите по дисертацията

Основна част от представените в дисертацията резултати са публикувани в 5 излезли от печат и 2 приети за печат статии, както следва:

Глава 2:

[73] D. Slavchev and S. Margenov. Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around Airfoils. In K. Georgiev, M. Todorov, and I. Georgiev, editors, *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, pages 369–381, Cham, 2019. Springer International Publishing. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30

[72] D. Slavchev and S. Margenov. Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, *Numerical Methods and Applications*, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8

[74] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing

Глава 3:

[70] D. Slavchev. On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2

[71] D. Slavchev. Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5

[76] D. Slavchev, S. Margenov, and I.G. Georgiev. On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using hss compression. In *AIP Conference Proceedings*, volume 2302, page 120008, 2020. doi: 10.1063/5.0034506

Глава 4:

[75] D. Slavchev and S. Margenov. Performance study of hierarchical semi-separable compression solver for parabolic problems with space-fractional diffusion. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, Cham, in press. Springer International Publishing

Апробация на резултатите

Резултати, включени в дисертацията, са докладвани на редица международни конференции и уъркшопи.

Международни конференции:

- Large-Scale Scientific Computations (LSSC), Созопол, 2017, 2019, 2021
- Annual Meeting of the Bulgarian Section of SIAM (BGSIAM), София, 2017, 2018, 2019
- Numerical Methods for Scientific Computations and Advanced Applications (NMSCAA), Хисаря, 2018
- Numerical Methods and Applications (NM&A), Боровец, 2018
- Twelfth On-Line Conference of the Euro-American Consortium for Promoting the Application of Mathematics in Technical and Natural Sciences (AMiTaNS), Албена, 2020

Уъркошопи:

- Две години Авитохол: Иновативни Суперкомпютърни Приложения, Панагюрище, 2017
- Numerical Solution of Fractional Differential Equations and Applications (NSFDE&A), Созопол, 2020

Основни научни и научно-приложни приноси

1. Изследвана е производителността на следните софтуерни пакети за решаване на линейни системи с плътни матрици, с помощта на блочна LU факторизация: за процесори с общо предназначение (CPU) – пакетът на Intel Math Kernel Library (MKL) и библиотеката със свободен достъп Parallel Linear Algebra Software for Multicore Architectures (PLASMA); за ускорители с архитектурата Many Integrated Core (MIC) на Intel – MKL и пакетът със свободен достъп Matrix Algebra on GPU and Multicore Architectures (MAGMA). Резултатите от числените експерименти за системи, получени при дискретизация с метод на граничните елементи за гранична задача за ламинарен поток около крилни профили на Жуковски са в съответствие с асимптотичните оценки на изчислителната сложност. Сравнителният анализ показва по-добро бързодействие и много добра паралелна скалируемост на пакета MKL.

2. Изследвана е изчислителната сложност, паралелната ефективност и относителната грешка на метод на йерархична полусепарабелна компресия (HSS). Числените експерименти са проведени с пакета със свободен достъп STRUctured Matrices PACKage (STRUMPACK), в който е реализиран паралелен солвър на базата HSS компресия и ULV-подобна факторизация. Сравнителният анализ включва два типа плътни матрици, които са получени при дискретизация с: а) метод на граничните елементи на гранична задача за ламинарен поток около крилни профили на Жуковски; б) метод на крайните елементи за двумерна дробно дифузионна гранична задача. Направен е сравнителен анализ на солвърите от пакетите MKL и STRUMPACK. Получена е характеризация в зависимост от прага на относителна грешка при HSS компресията на случаите, в които йерархичният метод има по-добро бързодействие.

3. Показано е, че за задачата за обтичане на профили на Жуковски при дискретизация по метода на граничните елементи, последователната номерация на възлите по границата на профилите води до матрица с подходяща за HSS компресия *структура*. Това не е така за дробно дифузионната гранична задача, дискретизирана с метод на крайните елементи. С цел подобряване на ефективността на йерархичната полусепарабелна компресия са предложени и изследвани пет метода за пренареждане на неизвестните. За три от тях са разработени собствени алгоритми и програмни реализации. Сравнителният анализ показва съществено подобрение на резултатите при прилагане методите на вложените сечения и на рекурсивната бисекция.

4. Разработен е метод, алгоритъм и програмна реализация за числено решаване на параболично уравнение с дробно дифузионен оператор по пространството. За дискретизация по времето е приложен неявен метод на Ойлер с постоянна стъпка по времето и диагонална концентрация на масата. Доказано е, че за тази нестационарна задача, изчислителната сложност на отделните части на алгоритъма създава условия за предимство на йерархичния метод на базата на HSS компресия. Това е потвърдено от проведените числени експерименти. Така за всички размерности на дискретната задача по пространството, както и при всички варианти на праг на относителна грешка, вариантът на програмата използваща солвъра от пакета STRUMPACK има по-добро бързодействие от този използващ MKL.

Декларация за оригиналност

Декларирам, че дисертацията съдържа оригинални резултати, получени при проведени от мен научни изследвания с подкрепата и съдействието на научния ми ръководител.

Резултатите, които са получени, описани и/или публикувани от други учени са коректно и подробно цитирани в библиографията.

Настоящият дисертационен труд не е прилаган за придобиване на научна степен в друго висше училище, университет или научен институт.

Подпис:

Благодарности

На първо място искам да благодаря на научния си ръководител професор Светозар Маргенов за съвместната ни работа, огромната му помощ при работата по тази дисертация и безкрайното му търпение към моето желание да довършвам всичко в последния момент.

Благодарен съм и на моята съпруга Илиана за постоянната подкрепа и търпение към дългите часове прекарани пред компютъра/вложени в тази работа.

Изказвам благодарност и на колегите от секция „Научни пресмятания“ (в азбучен ред) – Явор Вутов, Иван Георгиев, Красимир Георгиев, Силвия Грозданова, Невена Илиева, Никола Костурски, Елена Лилкова, Иван Лирков, Недю Попиванов, Станислав Стойков и Станислав Харизанов за помощта и подкрепата им. Благодарен съм на екипа поддържащ високопроизводителните изчислителни системи в института и най вече на Мария Дурчова и София Ивановска за помощта им при използването на машините.

Благодарен съм и на моите родители Гинка и Георги за подкрепата им и за директната им помощ при намирането на много правописни и стилистични грешки в тази работа.

Благодарен съм и на ръководителите на бакалавърската и магистърската ми работа Весела Пашева и Ваня Марангозова-Мартен, на които дължа интереса си към тази област.

Изказвам благодарността си и на бившите ми работодатели във Visteon, които се съгласиха да работя на половин работен ден и така подкрепиха работата по тази дисертация.

Специална благодарност изказвам на анонимния рецензент на статията [76], който ме насочи към използването на рекурсивна бисекция.

Благодаря и на трите си котки – Жубровка, Мърчо и на последното попълнение Рижко, които енергично се излежаваха около мен по време на работа и особено на Жубровка, която периодично лягаше между мен и клавиатурата и почти участва в работата.

Приложение А

Софтуерна имплементация на пренареждания на неизвестните за HSS компресия

За пренареждания на променливите в Глава 3 се използва MatLab. Използва се кодът на Акоста и др. [4] за генериране на мрежата в областта Ω, матрицата и дясната страна в плътната система линейни алгебрични уравнения. След това се прилага MatLab код който извършва пренареждането. За методите на вложените сечения и рекурсивната бисекция се използва пакетът `meshpart` [34]. В този апендикс е описан кодът, който се използва за всяко едно от разгледаните в Глава 3 пренареждания.

Използва се кодът от статията [4] за генериране на меша и на матрицата K и дясната страна F. След това се използва кодът описан в този апендикс за генериране на пренареждане.

Входните параметри за пренаречданията генерирани от main .m скрипта са:

- [p , ee , t t] е структурата на меша, където p е $2 \times N_{\mathcal{T}}$ вектор с координатите на възлите, ee са ребрата между областите и t t са триъгълните елементи.
- n f е масив с размер N от индексите на възлите намиращи се в $Omega \backslash \overline{\Omega}$. Тогава p(n f) е масив от всички възли намиращи се в $\Omega \setminus \overline{\Omega}$.

Изходният параметър на пренаречданията е:

- nff масив с размер N от пренаредените индекси. Тогава K(nff, nff) е пренаредената матрица, а b(nff) пренаредената дясна част.

A.1 Пренареждане по „Y“ координата – „top“

При пренареждането по „Y“ координата на всички възели в Ω се сортират по тяхната „Y“ координата със следният код:

```
[~, index_internal_points] = ...
    sort(p(2,nf));
nff = nf(index_internal_points);
```

A.2 Пренареждане по спирала – „snake“

Това пренареждане се избира „горният ляв“ възел за първи възел P_1. За следващ възел P_2 се избира възел съседен на P_1, такъв че, векторът $\overrightarrow{P_1P_2}$ да сключва минимален ъгъл по часовниковата стрелка с вектор $\overrightarrow{P_1P_0}$. p_0 е помощна възел, намираща се „горе в ляво“ (т.е. под ъгъл $-45°$) от първият възел P_1. Всеки следващ възел се избира от съседите на предишната, такава че ъгълът $\sphericalangle P_{i-2}P_{i-1}P_i$ да бъде минимален. Ако се стигне до възел, който няма съседни невзети възели, се разглеждат всички останали възели като потенциален следващ възел.

```
% Find the index of the point
% closest to the the upper left
[~,first_point] = max(p(2,nf(:))-p(1,nf(:)));
nff = nf;
no_neighbours = 0;
% Put the next node in it's place
[nff(1), nff(first_point)] = deal(nff(first_point), nff(1));
p1 = [p(1,nff(1))-1, p(2,nff(1))+1];
for i = 2:(length(nff)-1)
    % Find all neighbouring nodes with point nff(i-1)
    neighbour_nodes = tt(1:3,[find(tt(1,:)==nff(i-1)), ...
        find(tt(2,:)==nff(i-1)), find(tt(3,:)==nff(i-1))]);
    % Remove duplicates and nodes outside the solution domain
    neighbour_nodes = intersect(nff(i:length(nff)), ...
        neighbour_nodes(neighbour_nodes~=nff(i-1)));
    p0 = p(:,nff(i-1))';
    % Use the previous node as p1.
    % For the first use a pseudo point at (-1,-1 from it.)
```

```
    if i == 2
        p1 = [p(1,nff(1))-1, p(2,nff(1))+1];
    else
        p1 = p(:,nff(i-2))';
    end

    % If no neighbour nodes exist look up all remaining nodes.
    if length(neighbour_nodes) <= 0
        neighbour_nodes = nff(i:length(nff));
        no_neighbours = 1;
    else
        no_neighbours = 0;
    end
    % find the edge with the lowest angle to the previous edge
    for j = 1:length(neighbour_nodes)
        p2 = p(:,neighbour_nodes(j))';
        temp = atan2((det([p2-p0; p1-p0])), ...
            dot(p1-p0, p2-p0));
        if temp < 0, temp = temp + 2*pi;
        % If there are no neighbours
        % ignore angles below 90 degrees
        % and search all remaining nodes.
        if (temp < pi/2) && (no_neighbours ~= 0)
            temp = temp + 2*pi;
        end
        if j==1 || temp < min_angle
            min_angle=temp;
            min_node_id=find(nff(:)==neighbour_nodes(j));
        end
    end

     % Put the next node in place.
    [nff(i), nff(min_node_id)] = ...
        deal(nff(min_node_id), nff(i));
end
```

А.3 Пренареждане по линии – „stripes“

Това пренареждане започва като „snake“ пренареждането – с избора на „горният ляв“ възел за първи възел p_1. Следващият възел p_2 се избира измежду съседните на p_1 такъв, че ъгълът между оста Oy и $\overrightarrow{p_1p_2}$ да бъде минимален в интервала $[0°, \ldots, 180°]$. Всеки следващ възел се избира по същият начин.

Когато се стигне до възел, който няма подходящ следващ възел, се избира свободният връх „горе в ляво“, подобно на първия връх и се продължава.

```
% Find the index of the point closest to the the upper left
[~,first_point] = max(p(2,nf(:))-p(1,nf(:)));
nff = nf;
no_neighbours = 0;

% Put the next node in it's place
[nff(1), nff(first_point)] = deal(nff(first_point), nff(1));

for i = 2:(length(nff)-1)
    % Find all neighbouring nodes with point nff(i-1)
    neighbour_nodes = tt(1:3,[find(tt(1,:)==nff(i-1)), ...
        find(tt(2,:)==nff(i-1)), find(tt(3,:)==nff(i-1))]);

    % Remove duplicates and nodes outside the solution domain
    neighbour_nodes = intersect(nff(i:length(nff)), ...
        neighbour_nodes(neighbour_nodes~=nff(i-1)));
    min_angle = 2*pi; % Initialize minimum angle

    % find the edge with the lowest angle to the previous edge
    for j = 1:length(neighbour_nodes)
        p2 = p(:,neighbour_nodes(j))';
        temp = atan2(p(1,neighbour_nodes(j))-p(1,nff(i-1)),...
            p(2,neighbour_nodes(j))-p(2,nff(i-1)));

        % Remove the points on the left of the Y axis
        if (temp < 0), continue; end
        % find minimum angle
        if j==1 || temp < min_angle
            min_angle=temp;
            min_node_id=find(nff(:)==neighbour_nodes(j));
        end
    end
    % If no neighbours are available
    % search for the upper left one of the remaining nodes
    if min_angle==2*pi
        [~,min_node_id] = ...
            max(p(2,nff(i:length(nff))) ...
                -p(1,nff(i:length(nff))))

        % because the id is on the [i,length(nff)]
        min_node_id = min_node_id + i -1;
```

```
    end
     % Put the next node in it's place
    [nff(i), nff(min_node_id)] = ...
        deal(nff(min_node_id), nff(i));
end
```

А.4 Пренареждане с метод на вложените сечения и рекурсивна бисекция

За да се използват пренареждания с метод на вложените сечения или рекурсивна бисекция от пакета `meshpart` [34] е нужно да се изгради разредена матрица, в която ненулевите елементи съответстват на ребрата в триангулацията върху Ω. Получената матрица има същата разредена структура, каквато би имала и матрицата при решаване на локална (не-дробна) дифузия. (Под структура на разредената матрица тук разбираме наличието на нулеви и ненулеви елементи в разредената матрица. В останалата част от дисертацията под *структура* на плътната матрица се разбира наличието на извъндиагонални блокове с нисък ранг.)

За целта се използва следният MatLab код:

```
inter_p=p(:,nf);
j=1;
inter_tt=zeros(size(tt));

for i=1:nt
    if ismember(tt(1,i),nf) && ...
        ismember(tt(2,i),nf) && ismember(tt(3,i),nf)

        inter_tt(1,j)=find(nf==tt(1,i));
        inter_tt(2,j)=find(nf==tt(2,i));
        inter_tt(3,j)=find(nf==tt(3,i));
        inter_tt(4,j)=tt(4,i); j=j+1;
    end
end

% Remove the unneeded elements
inter_tt=inter_tt(1:3,1:(j-1));
AAA = sparse(length(nf),length(nf));

for i=1:length(nf)
    AAA(i,i) = 1;
```

```
end

for i=1:length(inter_tt)
    AAA(inter_tt(1,i),inter_tt(2,i)) = 1;
    AAA(inter_tt(2,i),inter_tt(1,i)) = 1;
    AAA(inter_tt(2,i),inter_tt(3,i)) = 1;
    AAA(inter_tt(3,i),inter_tt(2,i)) = 1;
    AAA(inter_tt(3,i),inter_tt(1,i)) = 1;
    AAA(inter_tt(1,i),inter_tt(3,i)) = 1;
    end
```

Изходният параметър е:

- AAA е разредената матрица, в която ненулевите елементи съответстват на ребрата между елементи. Диагоналните елементи също са ненулеви.

Пренареждането с метод на вложените сечение се получава с функция geond от пакета meshpart по следният начин:

```
nd = geond(AAA, transpose(inter_p));
nff=nf(nd);
```

Пренареждането с рекурсивна бисекция се получава с функция geodice от пакета meshpart по следният начин:

```
gd = geodice(AAA, transpose(inter_p), 9999);
[dummy,order]=sort(gd);
nff=nf(order);
```

gd е променлива, която съдържа, в кое парче от графа е подреден всеки връх от рекурсивната бисекция geodice, 9999 е избрано достатъчно голямо число, което задава на колко парчета да се раздели графът. В случая се разделя докато всяко парче има по-малко от 8 елемента (хардкодната стойност в алгоритъма). order е пренареждането което трябва да се приложи върху матрицата K и дясната част B.

Приложение Б

Асемблиране на матрица на масата с диагонална концентрация

Матрицата на масата с диагонална концентрация (lumped mass matrix) е диагонална. Тя се получава, като сумата от извъндиагоналните елементи на матрицата на масата по редове се добави към диагонала.

В дисертацията е използван следният код на MatLab за изчисляване на матрицата на масата с диагонална концентрация:

```
% calculate matrix of mass.
M = MassAssembler2D(p,t.');
M_small = M(nf,nf);
[row, col, v] = find(M_small);
M_global = [row, col, v];
% sum the values into the diagonals
% for a lumped matrix of mass
M_lumped = sparse(diag(full(sum(M_small))));
```

Тук:

- p е $2 \times n_{\tilde{\mathcal{T}}}$ масив от координатите на върховете в триангулацията $\tilde{\mathcal{T}}$, $n_{\tilde{\mathcal{T}}}$ е броят на върховете в $\tilde{\mathcal{T}}$
- t е $N_{\tilde{\mathcal{T}}} \times 3$ масив от индексите на върховете (топология на мрежата), които образуват всеки триъгълен елемент в $\tilde{\mathcal{T}}$.

Функцията MassAssembler2D се използва за изчисляване на матрицата на масата (без диагонална концентрация) със следния MatLab код:

```
function M = MassAssembler2D(p,t)
np = size(p,2);
% number of nodes
nt = size(t,2);
% number of elements
M = sparse(np,np);
% allocate mass matrix
for K = 1:nt
    % loop over elements
    loc2glb = t(1:3,K);     % local-to-global map
    x = p(1,loc2glb);     % node x-coordinates
    y = p(2,loc2glb);     % node y-coordinates
    area = polyarea(x,y); % triangle area
    MK = [2 1 1;1 2 1;1 1 2]/12*area; % element mass matrix
    M(loc2glb,loc2glb) = M(loc2glb,loc2glb) ...
        + MK; % add element masses to M
end
```

Тук:

- area е лицето на съответния елемент (триъгълник) S_e от триангулацията $\tilde{\mathcal{T}}$.
- MK е елементната матрица на масата, която може да се запише във вида:

$$M_e = \frac{S_e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Библиография

[1] AVITOHOL supercomputer. URL `http://www.hpc.acad.bg/system-1/`.

[2] Изчислителна система за обработка на данни, визуализация, разработка и тестване на приложения на ЦВП по информатика и ИКТ. URL `http://ict.acad.bg/?page_id=41`.

[3] G. Acosta and J.P. Borthagaray. A Fractional Laplace Equation: Regularity of Solutions and Finite Element Approximations. In *SIAM Journal on Numerical Analysis*, volume 55, pages 472–495, 2017. doi: 10.1137/15M1033952.

[4] G. Acosta, F.M. Bersetche, and J.P. Borthagaray. A short FE implementation for a 2d homogeneous Dirichlet problem of a fractional Laplacian. In *Computers & Mathematics with Applications*, volume 74, pages 784 – 816, 2017. doi: https://doi.org/10.1016/j.camwa.2017.05.026.

[5] AMD®. AMD Core Math Library User Guide. URL `https://developer.amd.com/wordpress/media/2012/10/acml_userguide.pd`.

[6] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. Improving multifrontal methods by means of block low-rank representations. In *SIAM Journal on Scientific Computing*, volume 37, pages A1451–A1474, 2015. doi: 10.1137/120903476.

[7] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

[8] M. Benzi, D. Bini, D. Kressner, H. Munthe-Kaas, and C. Van Loan. *Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications*. Springer, Cham, Cetraro, Italy, 2015. ISBN 978-3-319-49887-4. doi: 10.1007/978-3-319-49887-4.

[9] J. Bertoin. *Lévi Processes*. Cambridge University Press, October 1998. ISBN 9780521646321.

[10] K. Binder, Ch. Bennemann, J. Baschnagel, and W. Paul. Anomalous diffusion of polymers in supercooled melts near the glass transition. In A. Pękalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 124–139, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.

[11] K. Bogdan, K. Burdzy, and Z.-Q. Chen. Censored stable processes. In *Probability Theory and Related Fields*, volume 127, pages 89–152, Sep 2003. doi: 10.1007/s00440-003-0275-1.

[12] C.A. Brebbia, J.C.F. Telles, and L.C. Wrobel. *Boundary element techniques: theory and applications in engineering*. Springer-Verlag, 1984. ISBN 9783540124849. URL `https://books.google.bg/books?id=HpwQAQAAMAAJ`.

[13] C. Brown, A. Abdelfattah, S. Tomov, and J. Dongarra. Design, optimization, and benchmarking of dense linear algebra algorithms on amd gpus. In *2020 IEEE High Performance Extreme Computing Virtual Conference*. IEEE, 2020-09 2020.

[14] A. Buades, B. Coll, and J.M. Morel. Image Denoising Methods. A New Nonlocal Principle. In *SIAM Review*, volume 52, pages 113–147, 2010. doi: 10.1137/090773908.

[15] A. Buttari, J. Langou, J. Kurzak, and J. Dongarra. A class of parallel tiled linear algebra algorithms for multicore architectures. In *Parallel Computing*, volume 35, pages 38 – 53, 2009. doi: https://doi.org/10.1016/j.parco.2008.10.002.

[16] P. Carr, H. Geman, D.B. Madan, and M. Yor. The Fine Structure of Asset Returns: An Empirical Investigation. In *The Journal of Business*, volume 75, pages 305–332. The University of Chicago Press, 2002. URL `http://www.jstor.org/stable/10.1086/338705`.

[17] S. Chandrasekaran, M. Gu, and T. Pals. A fast *ulv* decomposition solver for hierarchically semiseparable representations. In *SIAM J. Matrix Anal. Appl.*, volume 28, page 603–622, USA, August 2006. Society for Industrial and Applied Mathematics. doi: 10.1137/S0895479803436652.

[18] S. Chaturapruek, J. Breslau, D. Yazdi, T. Kolokolnikov, and S.G. McCalla. Crime modeling with Lèvy flights. In *SIAM Journal on Applied Mathematics*,

volume 73, pages 1703–1720. Society for Industrial and Applied Mathematics, 2021/12/06/ 2013. URL http://www.jstor.org/stable/24510700.

[19] H. Cheng, Z. Gimbutas, P.G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. In *SIAM J. Sci. Comput.*, volume 26, page 1389–1404, USA, April 2005. Society for Industrial and Applied Mathematics. doi: 10.1137/030602678.

[20] G. Chávez, G. Turkiyyah, S. Zampini, H. Ltaief, and D. Keyes. Accelerated Cyclic Reduction: A distributed-memory fast solver for structured linear systems. In *Parallel Computing*, volume 74, pages 65–83, 2018. doi: https://doi.org/10.1016/j.parco.2017.12.001. Parallel Matrix Algorithms and Applications (PMAA'16).

[21] R. Cont and P. Tankov. *Financial Modelling with Jump Processes*. Chapman and Hall/CRC, first edition, Dec 2003. doi: 10.1201/9780203485217.

[22] J.H. Cushman and T. R. Ginn. Nonlocal dispersion in media with continuously evolving scales of heterogeneity. In *Transport in Porous Media*, volume 13, pages 123–138, Oct 1993. doi: 10.1007/BF00613273.

[23] G. D'Angelo and S. Rampone. Diagnosis of aerospace structure defects by a hpc implemented soft computing algorithm. In *2014 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 408–412, 2014. doi: 10.1109/MetroAeroSpace.2014.6865959.

[24] S. Donfack, J. Dongarra, M. Faverge, M. Gates, J. Kurzak, P. Luszczek, and I. Yamazaki. A survey of recent developments in parallel implementations of Gaussian elimination. In *Concurrency and Computation: Practice and Experience*, volume 27, pages 1292–1309, 2015. doi: 10.1002/cpe.3306.

[25] J. Dongarra, M. Abalenkovs, A. Abdelfattah, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, I. Yamazaki, and A. YarKhan. Parallel programming models for dense linear algebra on heterogeneous systems. In *Supercomputing Frontiers and Innovations*, volume 2, 2016. URL http://superfri.org/superfri/article/view/90.

[26] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, P. Wu, I. Yamazaki, A. Yarkhan, M. Abalenkovs, N. Bagherpour, S. Hammarling, J. Šístek, D. Stevens, M. Zounon, and S.D. Relton. Plasma: Parallel linear algebra software for multicore using openmp. In *ACM Trans. Math. Softw.*, volume 45, New York, NY, USA, May 2019. Association for Computing Machinery. doi: 10.1145/3264491.

[27] A.C. Eringen. *Nonlocal Continuum Field Theories*. Springer, New York, NY, first edition. ISBN 978-0-387-95275-8,978-1-4419-2931-0,978-0-387-22643-9. doi: 10.1007/b97697.

[28] G. Fasshauer. Lecture notes in Numerical Linear Algebra and Computational Mathematics, 2006. URL `http://www.math.iit.edu/~fass/477577_Chapter_7.pdf`.

[29] M. Ganzha, I. Lirkov, and M. Paprzycki. Performance analysis of hybrid parallel solver for 3D Stokes equation on Intel Xeon computer system. In *AIP Conference Proceedings*, volume 2164, page 120003, 2019. doi: 10.1063/1.5130863.

[30] P. Gatto and J.S. Hesthaven. Numerical Approximation of the Fractional Laplacian via *hp*-finite Elements, with an Application to Image Denoising. In *Journal of Scientific Computing*, volume 65, pages 249–270, Oct 2015. doi: 10.1007/s10915-014-9959-1.

[31] A. George. Nested dissection of a regular finite element mesh. In *SIAM Journal on Numerical Analysis*, volume 10, pages 345–363, 1973. doi: 10.1137/0710032.

[32] K. Georgiev, N. Kosturski, S. Margenov, and J. Starý. On adaptive time stepping for large-scale parabolic problems: Computer simulation of heat and mass transfer in vacuum freeze-drying. In *Journal of Computational and Applied Mathematics*, volume 226, pages 268–274, 2009. doi: https://doi.org/10.1016/j.cam.2008.08.020. URL `https://www.sciencedirect.com/science/article/pii/S0377042708004068`. Special Issue: Large scale scientific computations.

[33] P Ghysels, X.S. Li, F.-H. Rouet, S. Williams, and A. Napov. An Efficient Multicore Implementation of a Novel HSS-Structured Multifrontal Solver Using Randomized Sampling. In *SIAM Journal on Scientific Computing*, volume 38, pages S358–S384, 2016. doi: 10.1137/15M1010117.

[34] J.R. Gilbert, Y. Li, and S.-H. Teng. Mesh Partitioning Toolbox – meshpart. Zenodo, Apr 2020. doi: 10.5281/zenodo.3746723.

[35] G. Gilboa and S. Osher. Nonlocal Operators with Applications to Image Processing. In *Multiscale Modeling & Simulation*, volume 7, pages 1005–1028, 2009. doi: 10.1137/070698592.

[36] A. Golbabai and K. Sayevand. Analytical modelling of fractional advection–dispersion equation defined in a bounded space domain. In

Mathematical and Computer Modelling, volume 53, pages 1708–1718, 2011. doi: https://doi.org/10.1016/j.mcm.2010.12.046.

[37] C. Gorman, G. Chavez, P. Ghysels, T. Mary, F.-H. Rouet, and X. Li. Matrix-free construction of hss representation using adaptive randomized sampling. In *ArXiv*, volume abs/1810.04125, 2018.

[38] G. Grubb. Fractional Laplacians on domains, a development of Hörmander's theory of μ-transmission pseudodifferential operators. In *Advances in Mathematics*, volume 268, pages 478–528, 2015. doi: https://doi.org/10.1016/j.aim.2014.09.018.

[39] W. Hackbusch. A Sparse Matrix Arithmetic Based on $\mathcal{H}$-Matrices. Part I: Introduction to $\mathcal{H}$-Matrices. In *Computing*, volume 62, pages 89–108, Apr 1999. doi: 10.1007/s006070050015.

[40] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-matrices. In H.-J. Bungartz, R.H.W. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-642-59709-1. doi: 10.1007/978-3-642-59709-1_2.

[41] A. Haidar, J. Kurzak, and P. Luszczek. An improved parallel singular value algorithm and its implementation for multicore hardware. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 90:1–90:12, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2378-9. doi: 10.1145/2503210.2503292.

[42] S. Harizanov, R. Lazarov, and S. Margenov. A survey on numerical methods for spectral space-fractional diffusion problems:. In *Fractional Calculus and Applied Analysis*, volume 23, pages 1605–1646, 2020. doi: doi:10.1515/fca-2020-0080.

[43] S. Harizanov, S. Margenov, and N. Popivanov. Spectral Fractional Laplacian with Inhomogeneous Dirichlet Data: Questions, Problems, Solutions. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 123–138, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5.

[44] Intel®. Math Kernel Library. URL `https://software.intel.com/en-us/intel-mkl`.

[45] A. Karaivanova, V. Alexandrov, T. Gurov, and S. Ivanovska. On the Monte Carlo Matrix Computations on Intel MIC Architecture. In *Cybernetics*

and Information Technologies, volume 17, pages 49–59, 2017. doi: 10.1515-2017-0054.

[46] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 1991. ISBN 0-534-13014-3.

[47] J. Klafter and I.M. Sokolov. Anomalous diffusion spreads its wings. In *Physics World*, volume 18, pages 29–32. IOP Publishing, aug 2005. doi: 10.1088/2058-7058/18/8/33.

[48] N. Kosturski, S. Margenov, and Y. Vutov. Performance analysis of mg preconditioning on intel xeon phi: Towards scalability for extreme scale problems with fractional laplacians. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 304–312, Cham, 2018. Springer International Publishing. ISBN 978-3-319-73441-5.

[49] T.A.M. Langlands, B.I. Henry, and S.L. Wearne. Fractional Cable Equation Models for Anomalous Electrodiffusion in Nerve Cells: Finite Domain Solutions. In *SIAM Journal on Applied Mathematics*, volume 71, pages 1168–1203, 2011. doi: 10.1137/090775920.

[50] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized Nested Dissection. In *SIAM Journal on Numerical Analysis*, volume 16, pages 346–358, 1979. doi: 10.1137/0716027.

[51] I. Lirkov. Performance Analysis of a Scalable Algorithm for 3D Linear Transforms on Supercomputer with Intel Processors/Co-Processors. In *Cybernetics and Information Technologies*, volume 20, pages 94 – 104, Berlin, 01 Dec. 2020. Sciendo. doi: https://doi.org/10.2478/cait-2020-0064.

[52] I. Lirkov, S. Harizanov, M. Paprzycki, and M. Ganzha. Performance analysis of parallel high-resolution image restoration algorithms on Intel supercomputer. In *Concurrency and Computation: Practice and Experience*, volume 33, page e5996, 2021. doi: 10.1002/cpe.5996. e5996 CPE-20-0320.R1.

[53] L. Litov, P. Petkov, M. Rangelov, N. Ilieva, E. Lilkova, N. Todorova, E. Krachmarova, K. Malinova, A. Gospodinov, R. Hristova, I. Ivanov, and G. Nacheva. Molecular Mechanism of the Anti-Inflammatory Action of Heparin. In *Int J Mol Sci*, volume 22, Oct 2021. doi: 10.3390/ijms221910730.

[54] Y. Lou, X. Zhang, S. Osher, and A. Bertozzi. Image Recovery via Nonlocal Operators. In *Journal of Scientific Computing*, volume 42, pages 185–197, Feb 2010. doi: 10.1007/s10915-009-9320-2.

[55] Z. Mao, N. Chimitt, and S.H. Chan. Accelerating atmospheric turbulence simulation via learned phase-to-space transform. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14759–14768, 2021.

[56] P.G. Martinsson. A Fast Randomized Algorithm for Computing a Hierarchically Semiseparable Representation of a Matrix. In *SIAM Journal on Matrix Analysis and Applications*, volume 32, pages 1251–1274, 2011. doi: 10.1137/100786617.

[57] M. McCool, J. Reinders, and A. Robison. *Structured Parallel Programming: Patterns for Efficient Computation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012. ISBN 9780123914439.

[58] R. Metzler and J. Klafter. The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics. In *Journal of Physics A: Mathematical and General*, volume 37, pages R161–R208. IOP Publishing, jul 2004. doi: 10.1088/0305-4470/37/31/r01.

[59] G.E. Moore. Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. In *IEEE Solid-State Circuits Society Newsletter*, volume 11, pages 33–35, 2006. doi: 10.1109/N-SSC.2006.4785860.

[60] S. Moretti. In Silico Experiments in Scientific Papers on Molecular Biology. In *Science & Technology Studies*, volume 24, page 23–42, Jan. 2011. doi: 10.23987/sts.55262.

[61] R. Musina and A.I. Nazarov. On fractional laplacians. In *Communications in Partial Differential Equations*, volume 39, pages 1780–1790. Taylor & Francis, 2014. doi: 10.1080/03605302.2013.864304.

[62] A. Napov and X.S. Li. An algebraic multifrontal preconditioner that exploits the low-rank property. In *Numerical Linear Algebra with Applications*, volume 23, pages 61–82, 2016. doi: https://doi.org/10.1002/nla.2006.

[63] V. Pasheva and R. Lazarov. Boundary element method for 2D problems of ideal fluid flows with free boundaries. In *Advances in Water Resources*, volume 12, pages 37–45, 1989. doi: 10.1016/0309-1708(89)90014-6.

[64] E. Rebrova, G. Chávez, Y. Liu, P. Ghysels, and X.S. Li. A Study of Clustering Techniques and Hierarchical Matrix Formats for Kernel Ridge Regression. In *2018 IEEE International Parallel and Distributed Processing Symposium*

Workshops (IPDPSW), pages 883–892, 2018. doi: 10.1109/IPDPSW.2018.00140.

[65] L. Rosasco, M. Belkin, and E. De Vito. On learning with integral operators. In *Journal of Machine Learning Research*, volume 11, pages 905–934, 2010. URL `http://jmlr.org/papers/v11/rosasco10a.html`.

[66] F.-H. Rouet, X.S. Li, P. Ghysels, and A. Napov. A Distributed-Memory Package for Dense Hierarchically Semi-Separable Matrix Computations Using Randomization. In *ACM Trans. Math. Softw.*, volume 42, pages 27:1–27:35, New York, NY, USA, June 2016. ACM. doi: 10.1145/2930660.

[67] S.A. Silling. Reformulation of elasticity theory for discontinuities and long-range forces. In *Journal of the Mechanics and Physics of Solids*, volume 48, pages 175–209, 2000. doi: https://doi.org/10.1016/S0022-5096(99)00029-0.

[68] H.D. Simon and S.-H. Teng. How Good is Recursive Bisection? In *SIAM Journal on Scientific Computing*, volume 18, pages 1436–1445, 1997. doi: 10.1137/S1064827593255135.

[69] D. Slavchev. Parallelization of boundary element method for laplasian equation. Master's thesis, Technical University - Sofia, 2014.

[70] D. Slavchev. On the impact of reordering in a hierarchical semi-separable compression solver for fractional diffusion problems. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, pages 373–381, Cham, 2020. Springer International Publishing. ISBN 978-3-030-41032-2.

[71] D. Slavchev. Performance Analysis of Hierarchical Semi-separable Compression Solver for Fractional Diffusion Problems. In I. Georgiev, H. Kostadinov, and E. Lilkova, editors, *Advanced Computing in Industrial Mathematics*, pages 333–344, Cham, 2021. Springer International Publishing. ISBN 978-3-030-71616-5.

[72] D. Slavchev and S. Margenov. Analysis of Hierarchical Compression Parallel Solver for BEM Problems on Intel Xeon CPUs. In G. Nikolov, N. Kolkovska, and K. Georgiev, editors, *Numerical Methods and Applications*, pages 466–473, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10692-8.

[73] D. Slavchev and S. Margenov. Performance Analysis of Intel Xeon Phi MICs and Intel Xeon CPUs for Solving Dense Systems of Linear Algebraic Equations: Case Study of Boundary Element Method for Flow Around Airfoils. In K. Georgiev, M. Todorov, and I. Georgiev, editors, *Advanced Computing in Industrial Mathematics: BGSIAM 2017*, pages 369–381, Cham,

2019. Springer International Publishing. ISBN 978-3-319-97277-0. doi: 10.1007/978-3-319-97277-0_30.

[74] D. Slavchev and S. Margenov. Performance analysis of a parallel hierarchical semi-separable compression solver in shared and distributed memory environment for bem discretization of flow around airfoils. In *Advanced Computing in Industrial Mathematics*, Cham, in press. Springer International Publishing.

[75] D. Slavchev and S. Margenov. Performance study of hierarchical semi-separable compression solver for parabolic problems with space-fractional diffusion. In I. Lirkov and S. Margenov, editors, *Large-Scale Scientific Computing*, Cham, in press. Springer International Publishing.

[76] D. Slavchev, S. Margenov, and I.G. Georgiev. On the application of recursive bisection and nested dissection reorderings for solving fractional diffusion problems using hss compression. In *AIP Conference Proceedings*, volume 2302, page 120008, 2020. doi: 10.1063/5.0034506.

[77] S. Stoykov, E. Atanassov, and S. Margenov. Efficient Sparse Matrix-Matrix Multiplication for Computing Periodic Responses by Shooting Method on Intel Xeon Phi. In *Application of Mathematics in Technical and Natural Sciences*, volume 1773, page 110012. AIP, 2016. doi: 0.1063/1.4965016.

[78] H. Taitelbaum. Diagnosis using photon diffusion: From brain oxygenation to the fat of the atlantic salmon. In A. Pękalski and K. Sznajd-Weron, editors, *Anomalous Diffusion From Basics to Applications*, pages 160–174, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-49259-7.

[79] P.N. Vabishchevich. Splitting schemes for non-stationary problems with a rational approximation for fractional powers of the operator. In *Applied Numerical Mathematics*, volume 165, pages 414–430, 2021. doi: https://doi.org/10.1016/j.apnum.2021.03.006.

[80] Enrico Valdinoci. From the long jump random walk to the fractional laplacian, 2009.

[81] M.I. Vishik and G.I. Eskin. Equations in Convolutions in a Bounded Region. In *Russian Mathematical Surveys*, volume 20, pages 85–151. IOP Publishing, jun 1965. doi: 10.1070/rm1965v020n03abeh001184.

[82] S. Wang, X.S. Li, F. Rouet, J. Xia, and M.V. De Hoop. A parallel geometric multifrontal solver using hierarchically semiseparable structure. In *ACM*

Trans. Math. Softw., volume 42, New York, NY, USA, May 2016. Association for Computing Machinery. doi: 10.1145/2830569.

[83] R.C. Whaley. Atlas (automatically tuned linear algebra software). In D. Padua, editor, *Encyclopedia of Parallel Computing*, pages 95–101, Boston, MA, 2011. Springer US. ISBN 978-0-387-09766-4. doi: 10.1007/978-0-387-09766-4_85.

[84] R.C. Whaley. *ATLAS Installation Guide*. 2016. URL `http://math-atlas.sourceforge.net/atlas_install/`.

[85] J. Xia. Efficient structured multifrontal factorization for general large sparse matrices. In *SIAM Journal on Scientific Computing*, volume 35, pages A832–A860, 2013. doi: 10.1137/120867032.

[86] J. Xia. Randomized Sparse Direct Solvers. In *SIAM Journal on Matrix Analysis and Applications*, volume 34, pages 197–227, 2013. doi: 10.1137/12087116X.

[87] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. In *Numerical Lin. Alg. with Applic.*, volume 17, pages 953–976, 2010. doi: 10.1002/nla.691.

[88] Б. Боянов. *Лекции по Числени методи*. Дарба, 1998. ISBN 9549012611.

[89] П. Боянова. *Оптимални многонивови методи за некомформни крайни елементи*. Дисертация, ИИКТ-БАН, 2011.

[90] С. Маргенов. *Числени методи за системи с разредени матрици*. Институт по паралелна обработка на информацията, Българска академия на науките, 2007. ISBN 9780198520115.