Софийски Университет „Св. Климент Охридски“
Факултет по Математика и Информатика

Гергана Иванова Бенчева, фак.№ 10250

ДИПЛОМНА РАБОТА

на тема: ***Бързи директни методи за числено решаване на елиптични уравнения с разделящи се променливи***

специалност: Математика
специализация: Числени методи и алгоритми
научен ръководител: ст.н.с. IIст. Панайот Василевски
ръководител на катедра: акад. Борислав Боянов

София, 1998 г.

Съдържание

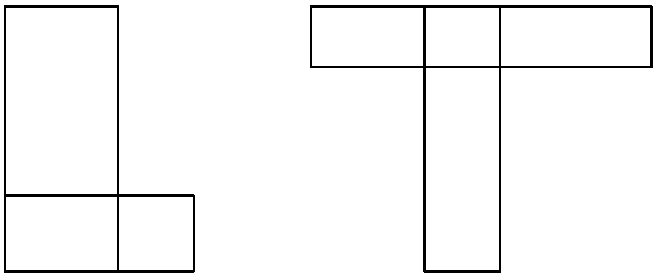
Увод

Много задачи на класическата математическа физика, представляващи математически модели на различни процеси в природата, се свеждат до гранични задачи за диференциални уравнения. Поради факта, че няма универсален начин за тяхното решаване, а и не всеки път може да се посочи аналитичен запис на точното решение, доста често се прибягва до употребата на числени методи. Все по-широкото навлизане в практиката на съвременни изчислителни машини с висока производителност и бързодействие обуславя все по-интензивното използване на дискретни методи при решаване на диференциалните уравнения.

Методът на крайните разлики (МКР) и методът на крайните елементи (МКЕ) са най-често прилаганите методи за дискретизация на диференциални уравнения и водят до решаване на системи алгебрични уравнения. Матриците на такива системи са обикновено от голяма размерност, но с разредена структура, т.е. броят на ненулевите елементи е пропорционален на броя на неизвестните. За малки размерности, в резултат на по-груба дискретизация на областта, такива системи могат да се решават с директни методи, като метод на Гаус, разлагане на триъгълни множители (LU-факторизация на матрици). При дискретизация на елиптични диференциални уравнения върху правоъгълни мрежи, в случай, че коефициентите на диференциалния оператор са константи или позволяват разделяне на променливите, за решаване на получената система могат да се използват варианти на дискретния метод за разделяне на променливите и на обобщения марш-алгоритъм. Това са някои от т.н. бързи директни методи, които всъщност са високоефективни методи за решаване на елиптични задачи с разделящи се променливи върху правоъгълна област.

Освен ограниченото самостоятелно значение, посочените директни методи имат и приложение за конструиране на преобусловители за решаване на елиптични задачи от втори ред с плавно променящи се коефициенти. При задачи, за които може да се използва метода на разделяне на областта на подобласти, като уравнение на Лаплас в L-образна (фиг. 0.1 (а)) или Т-образна (фиг. 0.1 (б)) област, при предположение, че в подобластите задачата е с разделящи се променливи, също могат да се използват посочените по-горе преки методи при построяване на преобусловители за задачите в съставните области.

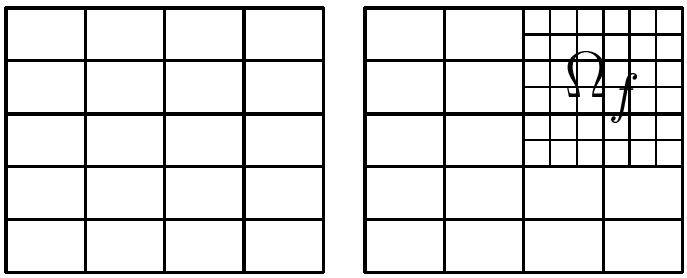
Решаването на елиптични задачи с разделящи се променливи в правоъгълни области има подобно приложение при задачи с локално сгъстяване на



(а) L-образна област (б) T-образна област

Фигура 0.1: Разделяне на областта на подобласти

мрежата, както е показано на фиг. 0.2 (а) и фиг. 0.2 (б). Тук преобусловителят използва пряк метод върху регулярна груба мрежа (фиг. 0.2 (а)), както и



(а) Груба мрежа (б) Фина мрежа - Ω_f

Фигура 0.2: Локално сгъстяване на мрежата

върху регулярна фина мрежа Ω_f (фиг. 0.2 (б)).

Предвид тези приложения на директните методи е важно те да са бързи, т.е. да изискват по-малък брой аритметични операции за изпълнението им.

Много от бързите директни методи са разработени за решаване на уравнението на дифузия в правоъгълна област, а първите бързи директни методи за решаване на уравнения от типа на уравнението на Поасон са предложени в средата на 60-те години.

При дискретизация на елиптични уравнения с разделящи се променливи в правоъгълна област чрез МКР по петточков шаблон „кръст“ или чрез на части линейни триъгълни крайни елементи, се получават блочно тридиагонални матрици. В такъв случай бързите директни методи могат да бъдат съществено по-бързи от итеративните методи.

В настоящата дипломна работа са разгледани стандартния дискретен алгоритъм за разделяне на променливите (SV) и бърз алгоритъм за разделяне на променливите (FASV), както и стандартен и обобщен марш алгоритъм за решаване на двумерна елиптична гранична задача на Дирихле от втори ред

с разделящи се променливи в правоъгълна област (глава 1), дискретизирана върху равномерна $n \times m$ мрежа.

Бързият алгоритъм за разделяне на променливите (FASV) се основава на техниката на четно-нечетна блочна елиминация в комбинация с дискретния метод за разделяне на променливите (SV). В общия случай на задачи с разделящи се променливи, за прилагане на SV се изискват всички собствени стойности и собствени вектори на една от двете тридиагонални матрици, с чиято помощ се конструира матрицата на изходната система. При FASV са необходими всички собствени стойности и по три компоненти от всички собствени вектори на една от споменатите матрици, както и на определени нейни подматрици.

При FASV общата изчислителна сложност $\mathcal{O}\left(m^2 n\right)$ на алгоритъма SV, се редуцира до $\mathcal{O}\left(nm \log m + n^2\right)$ аритметични операции.

В глава 2 на дипломната работа е изложено описанието на SV и FASV и описанието на техниката за непълно решаване на задачи с разредени десни части, която е в основата на FASV. Дадена е също оценка за общия брой операции, необходими за изпълнението на тези алгоритми. Целта на дипломната работа е да се направи реализация на MATLAB на тези алгоритми, за решаване на двумерна елиптична гранична задача на Дирихле с разделящи се променливи за правоъгълна област - коефициентите на диференциалния оператор са функции и граничните условия на Дирихле са в общ вид.

В глава 3 е конструирана устойчива версия на обобщения марш алгоритъм като е приложен бързия алгоритъм за разделяне на променливите.

Резултатите от проведените числени експерименти са показани в глава 4, а към дипломната работа е приложен и текста на програмата със съответните коментари.

Глава 1

Постановка на задачата

В тази глава е формулирана изходната диференциална задача, показано е накратко как от нея може да се получи дискретна задача, използвайки диференчна апроксимация по петточков шаблон „кръст“ върху равномерна мрежа. В зависимост от начина на подреждане на неизвестните, по хоризонтални или по вертикални линии, диференчната задача е записана в матричен вид по два различни начина като система линейни алгебрични уравнения.

1.1 Непрекъсната задача

Елиптичните задачи описват редица важни процеси; други по-сложни задачи се свеждат до решаване на елиптични задачи на всяка стъпка след линеаризация. При много физични задачи се среща уравнението на Поасон (Poisson) ($-\Delta u = f$), което го прави важен частен случай на елиптичните уравнения.

Разглежда се хомогенна задача на Дирихле (Dirichlet) за елиптично уравнение от втори ред с разделящи се променливи от вида:

$$\left| \begin{array}{rcll} -\sum\limits_{s=1}^{2} \dfrac{\partial}{\partial x_s}\left(a_s(x_s)\dfrac{\partial u}{\partial x_s}\right) & = & f(x), & x \in \Omega \\ u & = & 0, & \text{върху } \partial\Omega \end{array} \right. , \qquad (1.1.1)$$

където $x = (x_1, x_2)$ и $\Omega = (0,1)^2$. Коефициентите $a_s(x_s)$ са достатъчно гладки функции, такива че $0 < c_1 \leq a_s(x_s) \leq c_2$, $s = 1, 2$ ($c_1 > 0$, $c_2 > 0$ са константи), дясната страна $f(x)$ и търсената функция $u(x)$ са непрекъснати в $\bar{\Omega} = \Omega \bigcup \partial\Omega$.

1.2 Дискретизация

Задача (1.1.1) се дискретизира, като се използва диференчна апроксимация с грешка $\mathcal{O}\left(|h|^2\right)$ (където $|h|^2 = h_1^2 + h_2^2$), по петточков шаблон „кръст“ върху равномерна мрежа (Самарский [4]).

Нека вътрешните възли на мрежата са

$$(x_1^i, x_2^j) = (i\,h_1, j\,h_2),\ i = 1, \ldots, n,\ j = 1, \ldots, m$$

$$h_1 = \tfrac{1}{n+1},\ h_2 = \tfrac{1}{m+1}.$$

Всеки от диференциалните оператори

$$\frac{\partial}{\partial x_s}\left(a_s(x_s)\,\frac{\partial u}{\partial x_s}\right),\ s = 1, 2$$

се апроксимира с диференчен оператор от вида

$$\frac{1}{h_s}\left(b_{s,+1}\,\frac{u^{+1_s} - u}{h_s} - b_s\,\frac{u - u^{-1_s}}{h_s}\right),\ s = 1, 2\,, \tag{1.2.2}$$

където

$$b_s = a_s(x_s - \tfrac{1}{2}\,h_s)\,,$$
$$u^{\alpha_1} = u(x_1 + \alpha\,h_1, x_2),\ u^{\alpha_2} = u(x_1, x_2 + \alpha\,h_2)\,,\ \alpha = 1, -1\,,$$
$$b_{s,+1} = a_s((x_s - \tfrac{1}{2}\,h_s) + h_s) = a_s(x_s + \tfrac{1}{2}\,h_s)\,.$$

За апроксимация на дясната страна $f(x)$ се взима мрежова функция $\varphi(x)$ такава, че $\varphi(x) - f(x) = \mathcal{O}\,(|h|^2)$. Понеже $f(x)$ е непрекъсната, може да се вземе $\varphi(x) = f(x)$ за $x \in \{(x_1^i, x_2^j), i = 1, \ldots, n,\ j = 1, \ldots, m\,\}$.

Като се въведе означението $f_{i,j} = f(x_1^i, x_2^j)$, на диференциалната задача (1.1.1) се съпоставя следната диференчна задача на Дирихле

$$\left|\begin{array}{rcl}
-\frac{1}{h_1^2}\left(a_{1,i-\frac{1}{2}}\,y_{i-1,j} - (a_{1,i+\frac{1}{2}} + a_{1,i-\frac{1}{2}})\,y_{i,j} + a_{1,i+\frac{1}{2}}\,y_{i+1,j}\right) & & \\
-\frac{1}{h_2^2}\left(a_{2,j-\frac{1}{2}}\,y_{i,j-1} - (a_{2,j+\frac{1}{2}} + a_{2,j-\frac{1}{2}})\,y_{i,j} + a_{2,j+\frac{1}{2}}\,y_{i,j+1}\right) & & \\
\qquad = & f_{i,j}\,, & i = 1, \ldots, n, j = 1, \ldots, m, \\
y_{0,j} = y_{n+1,j} & = & 0, \qquad j = 1, \ldots, m, \\
y_{i,0} = y_{i,m+1} & = & 0, \qquad i = 1, \ldots, n
\end{array}\right. , \tag{1.2.3}$$

имаща грешка на апроксимация $\mathcal{O}\,(|h|^2)$.

Нека граничните условия на Дирихле за задача (1.1.1) са ненулеви, т.е. нека те имат вида:

$$u = \mu(x),\ \text{ върху }\ \partial\Omega\,.$$

Съответната диференчна задача на Дирихле е:

$$\left|\begin{array}{rcl}
-\frac{1}{h_1^2}\left(a_{1,i-\frac{1}{2}}\,y_{i-1,j} - (a_{1,i+\frac{1}{2}} + a_{1,i-\frac{1}{2}})\,y_{i,j} + a_{1,i+\frac{1}{2}}\,y_{i+1,j}\right) & & \\
-\frac{1}{h_2^2}\left(a_{2,j-\frac{1}{2}}\,y_{i,j-1} - (a_{2,j+\frac{1}{2}} + a_{2,j-\frac{1}{2}})\,y_{i,j} + a_{2,j+\frac{1}{2}}\,y_{i,j+1}\right) & & \\
\qquad = & f_{i,j}\,, & i = 1, \ldots, n, j = 1, \ldots, m, \\
y_{0,j} & = & \mu(0, x_2^j), \quad j = 1, \ldots, m, \\
y_{n+1,j} & = & \mu(1, x_2^j), \quad j = 1, \ldots, m, \\
y_{i,0} & = & \mu(x_1^i, 0), \quad i = 1, \ldots, n, \\
y_{i,m+1} & = & \mu(x_1^i, 1), \quad i = 1, \ldots, n
\end{array}\right. . \tag{1.2.4}$$

1.3 Еквивалентна система алгебрични уравнения

Нека $T = \{t_{i,j}\}_{i,j=1}^{n}$ и $B = \{b_{i,j}\}_{i,j=1}^{m}$ са матричните представяния на диференчните оператори (1.2.2) съответно за $s = 1$ и $s = 2$, т.е.

$$T = \frac{1}{h_1^2}\begin{pmatrix} a_{1,1+\frac{1}{2}} + a_{1,1-\frac{1}{2}} & -a_{1,1+\frac{1}{2}} & 0 & 0 \\ -a_{1,2-\frac{1}{2}} & a_{1,2+\frac{1}{2}} + a_{1,2-\frac{1}{2}} & -a_{1,2+\frac{1}{2}} & 0 \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ 0 & -a_{1,n-1-\frac{1}{2}} & a_{1,n-1+\frac{1}{2}} + a_{1,n-1-\frac{1}{2}} & a_{1,n-1+\frac{1}{2}} \\ 0 & 0 & -a_{1,n-\frac{1}{2}} & a_{1,n+\frac{1}{2}} + a_{1,n-\frac{1}{2}} \end{pmatrix},$$

$$B = \frac{1}{h_2^2}\begin{pmatrix} a_{2,1+\frac{1}{2}} + a_{2,1-\frac{1}{2}} & -a_{2,1+\frac{1}{2}} & 0 & 0 \\ -a_{2,2-\frac{1}{2}} & a_{2,2+\frac{1}{2}} + a_{2,2-\frac{1}{2}} & -a_{2,2+\frac{1}{2}} & 0 \\ \ddots & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ 0 & -a_{2,m-1-\frac{1}{2}} & a_{2,m-1+\frac{1}{2}} + a_{2,m-1-\frac{1}{2}} & a_{2,m-1+\frac{1}{2}} \\ 0 & 0 & -a_{2,m-\frac{1}{2}} & a_{2,m+\frac{1}{2}} + a_{2,m-\frac{1}{2}} \end{pmatrix}.$$

T и B са тридиагонални, симетрични и положително определени матрици.

Дефиниция 1.3.1 *Нека C е матрица с размерност $m_1 \times n_1$ и елементи $c_{i,j}$, а D е матрица с размерност $m_2 \times n_2$. Матрицата*

$$C \otimes D = \begin{pmatrix} c_{1,1}\,D & c_{1,2}\,D & \dots & c_{1,n_1}\,D \\ c_{2,1}\,D & c_{2,2}\,D & \dots & c_{2,n_1}\,D \\ \dots & \dots & \dots & \dots \\ c_{m_1,1}\,D & c_{m_1,2}\,D & \dots & c_{m_1,n_1}\,D \end{pmatrix}$$

е с размерност $m_1\,m_2 \times n_1\,n_2$ и се нарича тензорно произведение на матриците C и D.

Ако се използва подреждане на неизвестните по хоризонтални линии, т.е. по линии $y = const$, системата (1.2.3) се представя чрез тензорно произведение на матрици по следния начин:

$$A\mathbf{x} = \mathbf{f}\,, \tag{1.3.5}$$

където

$$A = B \otimes I_n + I_m \otimes T$$

$$= \begin{pmatrix} T + b_{1,1}\, I_n & b_{1,2}\, I_n & 0 & \dots & 0 \\ b_{2,1}\, I_n & T + b_{2,2}\, I_n & b_{2,3}\, I_n & \dots & 0 \\ \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots \\ 0 & \dots & b_{m-1,m-2}\, I_n & T + b_{m-1,m-1}\, I_n & b_{m-1,m}\, I_n \\ 0 & \dots & 0 & b_{m,m-1}\, I_n & T + b_{m,m}\, I_n \end{pmatrix},$$

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}, \mathbf{f} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_m \end{pmatrix},$$

$$\mathbf{x}_j = \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{n,j} \end{pmatrix}, \mathbf{f}_j = \begin{pmatrix} f_{1,j} \\ f_{2,j} \\ \vdots \\ f_{n,j} \end{pmatrix} \in \mathbf{R}^n, j = 1, \dots, m.$$

I_n и I_m са единични матрици съответно от ред n и m, а $x_{i,j}$ е стойността на търсената функция във възел (i, j) на мрежата от точка 1.2.

Ако се използва вертикално лексикографско подреждане на неизвестните, т.е. по линии $x = const$, системата (1.2.3) се записва във вида:

$$A'\mathbf{x}' = \mathbf{f}', \tag{1.3.6}$$

където

$$A' = T \otimes I_m + I_n \otimes B$$

$$= \begin{pmatrix} B + t_{1,1}\, I_m & t_{1,2}\, I_m & 0 & \dots & 0 \\ t_{2,1}\, I_m & B + t_{2,2}\, I_m & t_{2,3}\, I_m & \dots & 0 \\ \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots \\ 0 & \dots & t_{n-1,n-2}\, I_m & B + t_{n-1,n-1}\, I_m & t_{n-1,n}\, I_m \\ 0 & \dots & 0 & t_{n,n-1}\, I_m & B + t_{n,n}\, I_m \end{pmatrix},$$

$$\mathbf{x}' = \begin{pmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_n \end{pmatrix}, \mathbf{f}' = \begin{pmatrix} \mathbf{f}'_1 \\ \mathbf{f}'_2 \\ \vdots \\ \mathbf{f}'_n \end{pmatrix},$$

$$\mathbf{x}'_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,m} \end{pmatrix}, \mathbf{f}'_i = \begin{pmatrix} f_{i,1} \\ f_{i,2} \\ \vdots \\ f_{i,m} \end{pmatrix} \in \mathbf{R}^m, i = 1, \ldots, n.$$

В случая на нехомогенни гранични условия по аналогичен начин от системата (1.2.4) се получават системите

$$A\mathbf{x} = \tilde{\mathbf{f}} \tag{1.3.7}$$

и

$$A'\mathbf{x}' = \tilde{\mathbf{f}}', \tag{1.3.8}$$

където

$$\tilde{\mathbf{f}} = \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ \tilde{\mathbf{f}}_2 \\ \vdots \\ \tilde{\mathbf{f}}_m \end{pmatrix}, \tilde{\mathbf{f}}_j = \begin{pmatrix} \tilde{f}_{1,j} \\ \tilde{f}_{2,j} \\ \vdots \\ \tilde{f}_{n,j} \end{pmatrix} \in \mathbf{R}^n, j = 1, \ldots, m,$$

$$\tilde{\mathbf{f}}' = \begin{pmatrix} \tilde{\mathbf{f}}'_1 \\ \tilde{\mathbf{f}}'_2 \\ \vdots \\ \tilde{\mathbf{f}}'_n \end{pmatrix}, \tilde{\mathbf{f}}'_i = \begin{pmatrix} \tilde{f}_{i,1} \\ \tilde{f}_{i,2} \\ \vdots \\ \tilde{f}_{i,m} \end{pmatrix} \in \mathbf{R}^m, i = 1, \ldots, n,$$

а матриците A и A' и векторите $\mathbf{x}$ и $\mathbf{x}'$ имат вида от съответното представяне (1.3.5) или (1.3.6).

Компонентите $\tilde{f}_{i,j}$ на векторите $\tilde{\mathbf{f}}$ и $\tilde{\mathbf{f}}'$ са получени от $f_{i,j}$ след налагане на граничните условия и са определени с равенствата:

$$\begin{aligned}
\tilde{f}_{1,j} &= f_{1,j} + \tfrac{1}{h_1^2}\, a_{1,1-\frac{1}{2}}\, y_{0,j}, && j = 2, \ldots, m-1, \\
\tilde{f}_{n,j} &= f_{n,j} + \tfrac{1}{h_1^2}\, a_{1,n+\frac{1}{2}}\, y_{n+1,j}, && j = 2, \ldots, m-1, \\
\tilde{f}_{i,1} &= f_{i,1} + \tfrac{1}{h_2^2}\, a_{2,1-\frac{1}{2}}\, y_{i,1}, && i = 2, \ldots, n-1, \\
\tilde{f}_{i,m} &= f_{i,m} + \tfrac{1}{h_2^2}\, a_{2,m+\frac{1}{2}}\, y_{i,m+1}, && i = 2, \ldots, n-1,
\end{aligned}$$

$$\begin{aligned}
\tilde{f}_{1,1} &= f_{1,1} + \tfrac{1}{h_1^2}\, a_{1,1-\frac{1}{2}}\, y_{0,1} + \tfrac{1}{h_2^2}\, a_{2,1-\frac{1}{2}}\, y_{1,0} \\
\tilde{f}_{1,m} &= f_{1,m} + \tfrac{1}{h_1^2}\, a_{1,1-\frac{1}{2}}\, y_{0,m} + \tfrac{1}{h_2^2}\, a_{2,m+\frac{1}{2}}\, y_{1,m+1} \\
\tilde{f}_{n,1} &= f_{n,1} + \tfrac{1}{h_1^2}\, a_{1,n+\frac{1}{2}}\, y_{n+1,1} + \tfrac{1}{h_2^2}\, a_{2,1-\frac{1}{2}}\, y_{n,0} \\
\tilde{f}_{n,m} &= f_{n,m} + \tfrac{1}{h_1^2}\, a_{1,n+\frac{1}{2}}\, y_{n+1,m} + \tfrac{1}{h_2^2}\, a_{2,m+\frac{1}{2}}\, y_{n,m+1}
\end{aligned}$$

$$\tilde{f}_{i,j} = f_{i,j},\ i = 2, \ldots, n-1,\ j = 2, \ldots, m-1.$$

Същата дискретна задача може да бъде получена, като за дискретизация на диференциалната задача (1.1.1), се използват на части линейни крайни елементи върху равномерна триъгълна мрежа.

Глава 2

Метод за разделяне на променливите

В точка 2.1 на настоящата глава е разгледан стандартния метод за разделяне на променливите (SV) за решаване на системи линейни алгебрични уравнения от вида (1.3.5) или (1.3.6). В случая на дискретизация на изходната задача върху равномерна $n \times m$ мрежа, за $m = \mathcal{O}(n)$, метода SV има изчислителна сложност $\mathcal{O}(n^3)$ аритметични операции.

Техниката за непълно решаване на задачи с разредени десни части (точка 2.2), предложена независимо от Бенегас (Benegas), Прошкуровски (Proskurowski) и Кузнецов (Kuznetsov), е в основата на бързия алгоритъм за разделяне на променливите (FASV). В точка 2.3 е описан FASV и е показано как за дискретизация върху същата мрежа, изчислителната сложност се редуцира до $\mathcal{O}(n^2 \log n)$ аритметични операции.

2.1 Дискретен метод за разделяне на променливите

При описанието на дискретния метод за разделяне на променливите, се разглежда диференчната задача (1.2.3), записана в матричния вид (1.3.6).

С $\{\mathbf{q}_k, \lambda_k\}_{k=1}^m$ са означени ортонормираните собствени вектори и съответните им собствени стойности на матрицата B, т.е. $B\,\mathbf{q}_k = \lambda_k\,\mathbf{q}_k$, $k = 1, \ldots, m$. Тъй като B е симетрична и положително определена, то $\lambda_k > 0, k = 1, \ldots, m$ и

$$\mathbf{q}_k^T \cdot \mathbf{q}_r = \delta_{k,r},\ k, r = 1, \ldots, m,$$

където $\delta_{k,r}$ е символа на Кронекер (Kronecker).

Векторите $\mathbf{x}'_i$ и $\mathbf{f}'_i$ за $i = 1, \ldots, n$ се разлагат в базиса $\{\mathbf{q}_k\}_{k=1}^m$:

$$\begin{array}{rcl} \mathbf{x}'_i & = & \sum\limits_{k=1}^{m} \eta_{i,k}\, \mathbf{q}_k,\ i = 1, \ldots, n \\ \mathbf{f}'_i & = & \sum\limits_{k=1}^{m} \beta_{i,k}\, \mathbf{q}_k,\ i = 1, \ldots, n \end{array} , \qquad (2.1.1)$$

като коефициентите на Фурие (Fourier) $\beta_{i,k}$ за $\mathbf{f}'_i$ се определят чрез

$$\beta_{i,k} = \mathbf{f}_i'^T \cdot \mathbf{q}_k,\ i = 1, \ldots, n,\ k = 1, \ldots, m.$$

Решаването на системата (1.3.6) се свежда до намиране на коефициентите на Фурие $\eta_{i,k}$ за неизвестните вектори $\mathbf{x}'_i$.

След заместване на равенствата (2.1.1) в системата (1.3.6) и след опростяване (използвано е, че $B\,\mathbf{q}_k = \lambda_k\,\mathbf{q}_k$, $I_m\,\mathbf{q}_k = \mathbf{q}_k$, $k = 1, \ldots, m$) се получава:

$$\left|\begin{array}{rcl} \sum\limits_{k=1}^{m} \left(\left(\lambda_k + t_{1,1}\right)\eta_{1,k} + t_{1,2}\,\eta_{2,k}\right)\,\mathbf{q}_k & = & \sum\limits_{k=1}^{m} \beta_{1,k}\,\mathbf{q}_k \\ \ldots & & \\ \sum\limits_{k=1}^{m} \left(t_{i,i-1}\,\eta_{i-1,k} + \left(\lambda_k + t_{i,i}\right)\eta_{i,k} + t_{i,i+1}\,\eta_{i+1,k}\right)\,\mathbf{q}_k & = & \sum\limits_{k=1}^{m} \beta_{i,k}\,\mathbf{q}_k \\ \ldots & & \\ \sum\limits_{k=1}^{m} \left(t_{n,n-1}\,\eta_{n-1,k} + \left(\lambda_k + t_{n,n}\right)\eta_{n,k}\right)\,\mathbf{q}_k & = & \sum\limits_{k=1}^{m} \beta_{n,k}\,\mathbf{q}_k \end{array}\right| . \qquad (2.1.2)$$

Тъй като системата вектори $\{\mathbf{q}_k\}_{k=1}^m$ е линейно независима и образува базис, то коефициентите пред съответните вектори от двете страни на уравненията в (2.1.2) трябва да са равни. Следователно за всяко $k = 1, \ldots, m$ се получава система от вида:

$$\left|\begin{array}{rcl} \left(\lambda_k + t_{1,1}\right)\eta_{1,k} + t_{1,2}\,\eta_{2,k} & = & \beta_{1,k} \\ \ldots & & \\ t_{i,i-1}\,\eta_{i-1,k} + \left(\lambda_k + t_{i,i}\right)\eta_{i,k} + t_{i,i+1}\,\eta_{i+1,k} & = & \beta_{i,k} \\ \ldots & & \\ t_{n,n-1}\,\eta_{n-1,k} + \left(\lambda_k + t_{n,n}\right)\eta_{n,k} & = & \beta_{n,k} \end{array}\right|$$

или в матричен вид

$$\left(\lambda_k\, I_m + T\right)\eta_k = \beta_k,\ k = 1, \ldots, m\,,$$

където

$$\eta_k \ = \ \begin{pmatrix} \eta_{1,k} \\ \eta_{2,k} \\ \vdots \\ \eta_{n,k} \end{pmatrix}, \beta_k \ = \ \begin{pmatrix} \beta_{1,k} \\ \beta_{2,k} \\ \vdots \\ \eta_{n,k} \end{pmatrix}, k = 1, \ldots, m\,.$$

С други думи, за да се намерят неизвестните $\eta_{i,k},\ i = 1, \ldots, n,\ k = 1, \ldots, m$, трябва да се решат m независими една от друга тридиагонални системи, с матрици с размерност $n \times n$. Всяка една от тях може да се реши чрез $LD^{-1}U$ факторизация, която в случая на тридиагонална матрица е добре известният метод на прогонката.

Дискретният алгоритъм за разделяне на променливите може да се опише по следния начин:

Алгоритъм 1 (Разделяне на променливите - SV)

Стъпка 0 *Намиране* на собствените вектори и собствените стойности $\{\mathbf{q}_k,\ \lambda_k\}_{k=1}^{m}$ на тридиагоналната матрица B.

Стъпка 1 *Пресмятане* на коефициентите на Фурие $\beta_{i,k}$ за векторите $\mathbf{f}_i'$, използвайки

$$\beta_{i,k} = \mathbf{f}_i'^{T} \cdot \mathbf{q}_k,\ i = 1, \ldots, n,\ k = 1, \ldots, m.$$

Стъпка 2 *Решаване* на m системи с тридиагонални матрици от ред n от вида

$$(\lambda_k\, I_n + T)\, \eta_k = \beta_k,\ k = 1, \ldots, m\,,$$

с цел да се получат коефициентите на Фурие $\eta_{i,k}$.

Стъпка 3 *Възстановяване* на решението на системата (1.3.6) (или (1.3.5)) въз основа на коефициентите на Фурие $\{\eta_{i,k}\}_{i=1\,k=1}^{n\quad m}$ на векторите $\mathbf{x}_i$. Има две възможности:

вертикално възстановяване: $\mathbf{x}_i' = \sum_{k=1}^{m} \eta_{i,k}\, \mathbf{q}_k,\ i = 1, \ldots, n$

хоризонтално възстановяване: $\mathbf{x}_j = \sum_{k=1}^{m} q_{j,k}\eta_k,\ j = 1, \ldots, m$

Методът на разделяне на променливите (**Алгоритъм 1**) изисква:

За стъпка 0 — $\mathcal{O}\,(m^3)$ аритметични операции за намиране на двойките собствени вектори и собствени стойности.

За стъпка 1 — $2\,m^2\,n$ аритметични операции.

За стъпка 2 — $m(5n - 4)$ аритметични операции за решаване на m тридиагонални системи, като допълнително са необходими

$$m[n \text{ деления } + 3(n-1) \text{ други операции}]$$

за точна $LD^{-1}U$ факторизация на матриците $\lambda_k\, I_n + T,\ k = 1, \ldots, m$.

За стъпка 3 — $2\,m^2\,n$ аритметични операции.

Този резултат може да се обобщи в следната

Теорема 2.1.1 *Алгоритъмът за разделяне на променливите за решаване на система с блочнотридиагонална матрица с размерност* $mn \times mn$ *изисква*

$$4m^2n + m(5n - 4)$$

аритметични операции в частта за решаване,

$$m[n \text{ деления } + 3(n-1) \text{ други операции}]$$

за факторизация на тридиагонални матрици и $\mathcal{O}(m^3)$ *аритметични операции за намиране на необходимите собствени вектори и собствени стойности.*

В случая $m = \mathcal{O}(n)$, който се среща често на практика, за изпълнение на алгоритъма са необходими $\mathcal{O}(n^3)$ аритметични операции.

2.2 Непълно решаване на задачи с разредени десни части

В тази точка се описва т.н. техника за непълно решаване на задачи от вида (1.3.5) $A\mathbf{x} = \mathbf{f}$ с разредена дясна част.

Предполага се, че дясната част $\mathbf{f}$ има само d ($d \ll m$) ненулеви блочни компоненти $\mathbf{f}_j$ и те са зададени с индексите $j_1, j_2, \ldots, j_d$, т.е.

$$\mathbf{f}_j = 0 \text{ за } j \neq j_1, j_2, \ldots, j_d$$

Тогава всеки вектор $\mathbf{f}'_i, i = 1, \ldots, n$ от пренаредения вектор дясна част има по d ненулеви скаларни компоненти $f_{i,j_s}, s = 1, \ldots, d$.

Допуска се също, че се търсят r ($r \ll m$) блочни компоненти $\mathbf{x}_j$ на решението $\mathbf{x}$ и нека те са

$$\mathbf{x}_{j'_1}, \mathbf{x}_{j'_2}, \ldots, \mathbf{x}_{j'_r}.$$

Следователно за $i = 1, \ldots, n$ трябва да се пресметнат само част от компонентите на $\mathbf{x}'_i$, а именно $j'_1, j'_2, \ldots, j'_r$-тите компоненти на всички $\mathbf{x}'_i$, $i = 1, \ldots, n$.

За да се намерят необходимите компоненти на решението се прилага алгоритъма за разделяне на променливите (**Алгоритъм 1**) по следния начин:

Алгоритъм 2 (За непълно решаване на задачи с разредени десни части - SRHS)

Стъпка 0 *Намиране* на всички собствени стойности и на d компоненти от всички собствени вектори на тридиагоналната матрица B.

Стъпка 1 *Изчисляване* на коефициентите на Фурие $\beta_{i,k}$ на $\mathbf{f}'_i$ от уравненията

$$\beta_{i,k} = \mathbf{q}_k^T \cdot \mathbf{f}'_i = \sum_{s=1}^{d} q_{j_s,k} f_{i,j_s},\ i = 1, \ldots, n,\ k = 1, \ldots, m.$$

Стъпка 2 *Решаване* на m системи линейни уравнения с тридиагонални матрици от ред n от вида

$$(\lambda_k\, I_n + T)\, \eta_k = \beta_k,\ k = 1, \ldots, m\,.$$

Стъпка 3 *Възстановяване* на r компоненти на решението по редове, използвайки формулите $\mathbf{x}_j = \sum_{k=1}^{m} q_{j,k}\eta_k,\ j = 1, \ldots, m.$

Алгоритъм 2 (за непълно решаване на задачи с разредени десни части) изисква следния брой аритметични операции:

За стъпка 0 — $\mathcal{O}\,(d\, m^2)$ аритметични операции за намиране на d компоненти на всички собствени вектори и $9m^2$ аритметични операции за намиране на всички собствени стойности.

За стъпка 1 — $2\, d\, m\, n$ аритметични операции (само d компоненти на $\mathbf{f}'_i$ са ненулеви).

За стъпка 2 — $m(5n-4)$ аритметични операции за решаване на m тридиагонални системи и допълнително

$$m[n \text{ деления } + 3(n-1) \text{ други операции }]$$

за точна $LD^{-1}U$ факторизация на матриците $\lambda_k\, I_n + T,\ k = 1, \ldots, m.$

За стъпка 3 — $2\, r\, m\, n$ аритметични операции, защото се търсят само r компоненти на решението.

Този резултат се обобщава в следната

Теорема 2.2.1 *Алгоритъмът за разделяне на променливите за решаване на задача (1.3.5) с разредена дясна част, например за която* $\mathbf{f}_j = 0$ *за* $j \neq j_1, j_2, \ldots, j_d$ *и когато се търсят само част от компонентите на решението, именно* $\mathbf{x}_j$ *за* $j = j'_1\,, j'_2\,, \ldots, j'_r$ *изисква*

$$m[2(r+d)n + (5n-4)]$$

аритметични операции за решаване,

$$m[n \text{ деления } + 3(n-1) \text{ други операции }]$$

за факторизация на тридиагоналните матрици $\lambda_k\, I_n + T$ *и* $\mathcal{O}\,(dm^2) + 9m^2$ *аритметични операции за намиране на всички собствени стойности и на* d *компоненти на всички собствени вектори.*

2.3 Бърз алгоритъм за разделяне на променливите (FASV)

2.3.1 Основна идея

Подобно на други директни методи като Гаусова елиминация, FASV се състои от прав и обратен ход.

За удобство при изложението се предполага, че

$$m = 2^l - 1.$$

На всяка стъпка блоковете на матрицата A се групират по подходящ начин, а именно:

$$A = \begin{pmatrix} A^{(k,1)} & A^{(k)}_{1,2} & 0 & \dots & 0 \\ A^{(k)}_{2,1} & T + b_{2^k,2^k} I_n & A^{(k)}_{2,3} & \dots & 0 \\ 0 & A^{(k)}_{3,2} & A^{(k,2)} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A^{(k)}_{2^{l-k+1}-1,2^{l-k+1}-2} & A^{(k,2^{l-k})} \end{pmatrix},$$

където матриците $A^{(k,s)}$ се определят с формулите

$$A^{(k,s)} = I_{2^k-1} \otimes T + B^{(s)}_k \otimes I_n,\ s = 1, 2, \dots, 2^{l-k}.$$

Матриците $B^{(s)}_k$ са подматрици на матрицата B и имат вида:

$$B^{(s)}_k = \begin{pmatrix} b_{s_k+1,s_k+1} & b_{s_k+1,s_k+2} & 0 & \dots & 0 \\ b_{s_k+2,s_k+1} & b_{s_k+2,s_k+2} & b_{s_k+2,s_k+3} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & b_{s_k+2^k-1,s_k+2^k-2} & b_{s_k+2^k-1,s_k+2^k-1} \end{pmatrix},$$

където $s_k = (s-1)2^k$.

С други думи, всяка от матриците $A^{(k,s)}$ и $B^{(s)}_k$, $1 \le k \le l$, $1 \le s \le 2^{l-k}$ има $2^k - 1$ блока от ред n.

Целта е на всяка стъпка на правия и обратния ход да може да се прилага алгоритъма за непълно решаване на задачи с разредени десни части.

На всяка стъпка на правия ход се конструират вектори $\mathbf{x}^{(k)}$ от решенията на системи с матрици $A^{(k,s)}$ и дясна част, съответния блок на вектор $\mathbf{f}^{(k)}$, с една ненулева блочна компонента. Блоковете на $\mathbf{x}^{(k)}$, съответстващи на матриците $T + b_{s2^k,s2^k} I_n$ се нулират. Векторите $\mathbf{f}^{(k)}$ за всяка следваща стъпка се определят чрез формулата

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} - A\mathbf{x}^{(k)},\ (\mathbf{f}^{(1)} = \mathbf{f})$$

и имат разреден вид.

По този начин, при нарастване на k, броят на нулевите компоненти на конструираните вектори $\mathbf{x}^{(k)}$ намалява, а този на дясната част $\mathbf{f}^{(k)}$ - се увеличава.

Показва се, че решението на изходната задача може да се разложи във вида:

$$\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(2)} + \ldots + \mathbf{x}^{(l)}.$$

При обратния ход отново се решават системи с матриците $A^{(k,s)}$, но десните части имат по две ненулеви блочни компоненти и се конструират, като се използва разлагането на $\mathbf{x}$ и начина на образуване на $\mathbf{f}^{(k)}$. Освен това те се решават непълно, като се търси само по една блочна компонента на решението.

Характерното е, че решението на изходната задача (1.3.5) се възстановява постепенно - най-напред една компонента ($\mathbf{x}_{2^{l-1}}$), след това - две ($\mathbf{x}_{2^{l-2}}$ и $\mathbf{x}_{3.2^{l-2}}$), след това - четири ($\mathbf{x}_{2^{l-3}}$, $\mathbf{x}_{3.2^{l-3}}$, $\mathbf{x}_{5.2^{l-3}}$ и $\mathbf{x}_{7.2^{l-3}}$) и т.н., докато се възстановят всички компоненти на решението.

В следващите две точки се описват подробно правия и обратния ход на алгоритъма FASV.

2.3.2 FASV-прав ход

Правият ход на алгоритъма FASV се състои от няколко стъпки.

На първата стъпка се полага

$$\mathbf{f}^{(1)} = \mathbf{f}.$$

За да се осигури разредена блочна структура на дясната част на разглежданата система за всяка следваща стъпка k, $\mathbf{f}^{(k+1)}$ (за $k = 1, \ldots, l-1$) се определя като вектор от остатъци:

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} - A\mathbf{x}^{(k)}. \tag{2.3.3}$$

Векторът $\mathbf{x}^{(k)}$, за всяко $k = 1, \ldots, l-1$, се конструира по следния начин: Решават се задачите

$$A^{(k,s)}\,\mathbf{x}^{(k,s)} = \mathbf{f}^{(k,s)},\ s = 1, 2, \ldots, 2^{l-k},$$

където

$$\mathbf{x}^{(k,s)} = \begin{pmatrix} \mathbf{x}^{(k)}_{s_k+1} \\ \mathbf{x}^{(k)}_{s_k+2} \\ \vdots \\ \mathbf{x}^{(k)}_{s_k+2^k-1} \end{pmatrix} \text{ и } \mathbf{f}^{(k,s)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f}^{(k)}_{s_k+2^{k-1}} \\ \mathbf{0} \end{pmatrix} \begin{matrix} \} & 2^{k-1}-1 & \text{блока} \\ \} & 1 & \text{блок} \\ \} & 2^{k-1}-1 & \text{блока} \end{matrix}$$

Освен това се полага $\mathbf{x}^{(k)}_{s2^k} = 0$ за $s = 1, 2, \ldots, 2^{l-k}-1$ и $\mathbf{x}^{(k)}$ и $\mathbf{f}^{(k)}$ са всъщност векторите

$$\mathbf{x}^{(k)} = \begin{pmatrix} \mathbf{x}^{(k,1)} \\ \mathbf{0} \\ \mathbf{x}^{(k,2)} \\ \mathbf{0} \\ \vdots \\ \mathbf{x}^{(k,2^{l-k})} \end{pmatrix} \begin{matrix} \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ & \vdots & \\ \} & 2^k-1 & \text{блока} \end{matrix}, \quad \mathbf{f}^{(k)} = \begin{pmatrix} \mathbf{f}^{(k,1)} \\ \mathbf{f}^{(k)}_{2^k} \\ \mathbf{f}^{(k,2)} \\ \mathbf{f}^{(k)}_{2.2^k} \\ \vdots \\ \mathbf{f}^{(k,2^{l-k})} \end{pmatrix} \begin{matrix} \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ & \vdots & \\ \} & 2^k-1 & \text{блока} \end{matrix} \tag{2.3.4}$$

Векторите $\mathbf{x}^{(k)}$ и $\mathbf{f}^{(k)}$ от (2.3.4) се заместват в (2.3.3) и след преобразуване, за вектора $\mathbf{f}^{(k+1)}$ се получава

$$\mathbf{f}^{(k+1)} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f}^{(k+1)}_{1.2^k} \\ \mathbf{0} \\ \mathbf{f}^{(k+1)}_{2.2^k} \\ \vdots \\ \mathbf{f}^{(k+1)}_{(2^{l-k}-1).2^k} \\ \mathbf{0} \end{pmatrix} \begin{matrix} \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ & \vdots & \\ \} & 1 & \text{блок} \\ \} & 2^k-1 & \text{блока} \end{matrix} .$$

По-точно векторът $\mathbf{f}^{(k+1)}$ може да има ненулеви блочни компоненти $\mathbf{f}^{(k+1)}_i$ само за $i = s2^k,\ s = 1, 2, \ldots, 2^{l-k}-1$ и те се дават с формулата:

$$\mathbf{f}^{(k+1)}_{s2^k} = \mathbf{f}^{(k)}_{s2^k} - A^{(k)}_{2s,2s-1}\mathbf{x}^{(k,s)} - A^{(k)}_{2s,2s+1}\mathbf{x}^{(k,s+1)} . \tag{2.3.5}$$

Въвежда се означението

$$\mathbf{f}^{(k+1,s')} = \begin{pmatrix} \mathbf{0} \\ \mathbf{f}^{(k+1)}_{s2^k} \\ \mathbf{0} \end{pmatrix} \begin{matrix} \} & 2^k-1 & \text{блока} \\ \} & 1 & \text{блок} \\ \} & 2^k-1 & \text{блока} \end{matrix}$$

където $s = 2s'-1$, $s' = 1, 2, \ldots, 2^{l-k-1}$ и $\mathbf{f}^{(k+1)}$ получава същия вид като вида на $\mathbf{f}^{(k)}$ в (2.3.4). По този начин броят на ненулевите компоненти на $\mathbf{f}^{(k)}$ на всяка стъпка намалява наполовина.

Пример: При $l = 4$, $m = 2^4 - 1 = 15$, векторите $\mathbf{f}^{(k)}$, $k = 1, \ldots 4$ ще имат следните ненулеви блокове:

$\mathbf{f}^{(1)}$:	$\mathbf{f}^{(1)}_1$	$\mathbf{f}^{(1)}_2$	$\mathbf{f}^{(1)}_3$	$\mathbf{f}^{(1)}_4$	$\mathbf{f}^{(1)}_5$	$\mathbf{f}^{(1)}_6$	$\mathbf{f}^{(1)}_7$	$\mathbf{f}^{(1)}_8$	$\mathbf{f}^{(1)}_9$	$\mathbf{f}^{(1)}_{10}$	$\mathbf{f}^{(1)}_{11}$	$\mathbf{f}^{(1)}_{12}$	$\mathbf{f}^{(1)}_{13}$	$\mathbf{f}^{(1)}_{14}$	$\mathbf{f}^{(1)}_{15}$
$\mathbf{f}^{(2)}$:		$\mathbf{f}^{(2)}_2$		$\mathbf{f}^{(2)}_4$		$\mathbf{f}^{(2)}_6$		$\mathbf{f}^{(2)}_8$		$\mathbf{f}^{(2)}_{10}$		$\mathbf{f}^{(2)}_{12}$		$\mathbf{f}^{(2)}_{14}$	
$\mathbf{f}^{(3)}$:				$\mathbf{f}^{(3)}_4$				$\mathbf{f}^{(3)}_8$				$\mathbf{f}^{(3)}_{12}$			
$\mathbf{f}^{(4)}$:								$\mathbf{f}^{(4)}_8$							

За да се опростят равенствата (2.3.5) е необходимо да се знае как изглеждат матриците $A_{i,j}^{(k)}$. За една матрица $A^{(k,s)}$ има четири матрици $A_{i,j}^{(k)}$:

$$\begin{array}{ccccc} & \vdots & \vdots & \vdots & \\ \ldots & A^{(k,s)} & A_{2s-1,2s}^{(k)} & 0 & \ldots \\ \ldots & A_{2s,2s-1}^{(k)} & T + b_{s2^k,s2^k} I_n & A_{2s,2s+1}^{(k)} & \ldots \\ \ldots & 0 & A_{2s+1,2s}^{(k)} & A^{(k,s+1)} & \ldots \\ & \vdots & \vdots & \vdots & \end{array},$$

където

$$A_{2s,2s-1}^{(k)} = (0, \ldots, 0, b_{s_k+2^k,s_k+2^k-1} I_n),\ A_{2s,2s+1}^{(k)} = (b_{s_k+2^k,s_k+2^k+1} I_n, 0, \ldots, 0),$$

$$A_{2s-1,2s}^{(k)} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_{s2^k-1,s2^k} I_n \end{pmatrix},\ A_{2s+1,2s}^{(k)} = \begin{pmatrix} b_{s2^k+1,s2^k} I_n \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

След опростяване на произведенията $A_{2s,2s-1}^{(k)} \mathbf{x}^{(k,s)}$ и $A_{2s,2s+1}^{(k)} \mathbf{x}^{(k,s+1)}$, за ненулевите блокове на $\mathbf{f}^{(k+1)}$ се получава

$$\begin{aligned} \mathbf{f}_{s2^k}^{(k+1)} &= \mathbf{f}_{s2^k}^{(k)} - b_{s2^k,s2^k-1} \mathbf{x}_{2^k-1}^{(k,s)} - b_{s2^k,s2^k+1} \mathbf{x}_1^{(k,s+1)} \\ &= \mathbf{f}_{s2^k}^{(k)} - b_{s2^k,s2^k-1} \mathbf{x}_{s2^k-1}^{(k)} - b_{s2^k,s2^k+1} \mathbf{x}_{s2^k+1}^{(k)}. \end{aligned} \tag{2.3.6}$$

Т.е. за пресмятане на ненулевите компоненти на $\mathbf{f}^{(k+1)}$ са необходими по две блочни компоненти на всяко $\mathbf{x}^{(k,s)}$ именно $\mathbf{x}_1^{(k,s)}$ и $\mathbf{x}_{2^k-1}^{(k,s)}$. При възстановяване на решението на обратния ход е необходима още компонентата $\mathbf{x}_{2^k-1}^{(k,s)}$ (вж.2.3.3).

Следователно задачите

$$A^{(k,s)} \mathbf{x}^{(k,s)} = \mathbf{f}^{(k,s)},\ s = 1, 2, \ldots, 2^{l-k}$$

имат разредена дясна част с една ненулева блочна компонента ($\mathbf{f}_{2^k-1}^{(k,s)}$) и могат да се решават непълно като се търсят три блочни компоненти на решението.

Може да се приложи **Алгоритъм 2** с брой ненулеви блокове в дясната част - $d = 1$, брой на търсените блочни компоненти на решението - $r = 3$ и размерност на блоковете $B_k^{(s)}$ - $m = 2^k - 1$.

Забележка: Матриците $A^{(k,s)}$ позволяват разделяне на променливите, защото са от същия вид като изходната матрица A, но B е заменена с нейна главна подматрица $B_k^{(s)}$.

2.3.3 FASV-обратен ход

При описанието на обратния ход на FASV ще са необходими следните факти:

1) Решението на системата $A\mathbf{x} = \mathbf{f}$ може да се разложи като сума на векторите $\mathbf{x}^{(k)}$, $k = 1, \ldots, l$:

$$\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(2)} + \ldots + \mathbf{x}^{(l)}. \tag{2.3.7}$$

За доказателство на това равенство се използва тъждеството

$$\mathbf{f}^{(1)} - \mathbf{f}^{(l+1)} = \sum_{k=1}^{l} (\mathbf{f}^{(k)} - \mathbf{f}^{(k+1)}) \,.$$

В него $\mathbf{f}^{(k+1)}$ се замества с равното му от (2.3.3) и след преобразуване се получава, че

$$\mathbf{f}^{(1)} - \mathbf{f}^{(l+1)} = \sum_{k=1}^{l} A\mathbf{x}^{(k)} = A\left(\sum_{k=1}^{l} \mathbf{x}^{(k)}\right).$$

От тук като се вземе предвид, че $\mathbf{f}^{(1)} = \mathbf{f}$ и че при $k = l$ от конструкцията

$$\mathbf{f}^{(l+1)} \equiv \mathbf{f}^{(l)} - A\mathbf{x}^{(l)} \equiv \mathbf{f}^{(l,1)} - A^{(l,1)}\mathbf{x}^{(l,1)} = 0 \,,$$

следва исканото равенство.

2) Според конструкцията

$$\mathbf{x}^{(k)}_{s2^k} = 0 \text{ за } s = 1, 2, \ldots, 2^{l-k} - 1.$$

Това равенство означава, че при нарастване на k, броят на нулевите компоненти на $\mathbf{x}^{(k)}$ намалява наполовина, като част от нулевите компоненти евентуално стават ненулеви, а другата част остават нули.

Пример: При $l = 4$, $m = 2^4 - 1 = 15$, векторите $\mathbf{x}^{(k)}$ имат нулеви компоненти на следните места:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathbf{x}^{(1)}$:		0		0		0		0		0		0		0	
$\mathbf{x}^{(2)}$:				0				0				0			
$\mathbf{x}^{(3)}$:								0							
$\mathbf{x}^{(4)}$:															

С други думи $\mathbf{x}^{(k)}_{s2^{k_0}} = 0$ за $k \leq k_0$. Като се вземе предвид и равенството (2.3.7) се получава, че

$$\mathbf{x}_{s2^{k_0}} = \left(\sum_{k=1}^{l} \mathbf{x}^{(k)}\right)_{s2^{k_0}} = \sum_{k>k_0} \mathbf{x}^{(k)}_{s2^{k_0}} \,. \tag{2.3.8}$$

При обратния ход на FASV целта е за $k = l, l-1, \ldots, 1$ да се определят

$$\mathbf{x}_{(2s-1)2^{k-1}} \text{ за } s = 1, 2, \ldots, 2^{l-k}.$$

Така на всяка стъпка броят на търсените компоненти на $\mathbf{x}$ се увеличава, докато се възстанови целия вектор $\mathbf{x}$.

Пример: При $l = 4$, $m = 2^4 - 1 = 15$, на всяка стъпка k се търсят $\mathbf{x}_{(2s-1)2^{k-1}}$ за $s = 1, 2, \ldots, 2^{4-k}$. Решението се възстановява по следния начин:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$k=4:$								$\mathbf{x}_8$							
$k=3:$				$\mathbf{x}_4$				$*$				$\mathbf{x}_{12}$			
$k=2:$		$\mathbf{x}_2$		$*$		$\mathbf{x}_6$		$*$		$\mathbf{x}_{10}$		$*$		$\mathbf{x}_{14}$	
$k=1:$	$\mathbf{x}_1$	$*$	$\mathbf{x}_3$	$*$	$\mathbf{x}_5$	$*$	$\mathbf{x}_7$	$*$	$\mathbf{x}_9$	$*$	$\mathbf{x}_{11}$	$*$	$\mathbf{x}_{13}$	$*$	$\mathbf{x}_{15}$

като $*$ означава, че съответната компонента вече е намерена.

На първата стъпка на обратния ход, т.е. когато $k = l$, $(2^{(l-k)} = 1,\ s = 1)$, трябва да се намери една блочна компонента на решението, а именно

$$\mathbf{x}_{(2s-1)2^{k-1}} \equiv \mathbf{x}_{2^{l-1}}.$$

От друга страна, от равенството (2.3.8) за $k_0 = l - 1$ и $s = 1$ следва, че $\mathbf{x}_{2^{l-1}} = \mathbf{x}^{(l)}_{2^{l-1}}$. Това означава, че средната блочна компонента на решението $\mathbf{x}$ на системата $A\mathbf{x} = \mathbf{f}$ се намира чрез непълно решаване (с $r = 1$) на системата

$$A\mathbf{x}^{(l)} = \mathbf{f}^{(l)}, \tag{2.3.9}$$

където $A \equiv A^{(l,1)}$, $\mathbf{x}^{(l)} = \mathbf{x}^{(l,1)}$ и $\mathbf{f}^{(l)} = \mathbf{f}^{(l,1)}$. Дясната част $\mathbf{f}^{(l)}$ е намерена на последната стъпка на правия ход и има разредена структура с една ненулева блочна компонента, т.е. $d = 1$. (Това следва от уравнение (2.3.6)).

За следващите стъпки $k = l - 1, \ldots, 1$, изхождайки от равенството (2.3.3)

$$\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} - A\mathbf{x}^{(k)},\ (\mathbf{f}^{(1)} \equiv \mathbf{f}),$$

се конструират системи, чието непълно решаване води до намиране на останалите компоненти на $\mathbf{x}$.

Тъй като

$$A\mathbf{x}^{(i)} = \mathbf{f}^{(i)} - \mathbf{f}^{(i+1)},\ i = 1, \ldots, l,$$

то

$$\begin{aligned} A\sum_{i=k}^{l}\mathbf{x}^{(i)} &= \sum_{i=k}^{l}A\mathbf{x}^{(i)} = \sum_{i=k}^{l}\left(\mathbf{f}^{(i)} - \mathbf{f}^{(i+1)}\right) \\ &= \mathbf{f}^{(k)} - \mathbf{f}^{(k+1)} + \mathbf{f}^{(k+1)} - \mathbf{f}^{(k+2)} + \ldots + \mathbf{f}^{(l-1)} - \mathbf{f}^{(l)} + \mathbf{f}^{(l)} - \mathbf{f}^{(l+1)} \\ &= \mathbf{f}^{(k)} - \mathbf{f}^{(l+1)} = \mathbf{f}^{(k)}. \end{aligned}$$

т.е.

$$A\sum_{i=k}^{l}\mathbf{x}^{(i)} = \mathbf{f}^{(k)}. \tag{2.3.10}$$

Ако матрицата A се представи в блочен вид чрез блоковете $A^{(k,s)}$ (както е показано в точка 2.3.1) и за

$$z^{(k)} = \sum_{i=k}^{l}\mathbf{x}^{(i)}$$

и $\mathbf{f}^{(k)}$ се използва представянето

$$\mathbf{z}^{(k)} = \begin{pmatrix} \mathbf{z}^{(k,1)} \\ \mathbf{z}^{(k)}_{2^k} \\ \vdots \\ \mathbf{z}^{(k,2^{l-k})} \end{pmatrix}, \mathbf{f}^{(k)} = \begin{pmatrix} \mathbf{f}^{(k,1)} \\ \mathbf{f}^{(k)}_{2^k} \\ \vdots \\ \mathbf{f}^{(k,2^{l-k})} \end{pmatrix},$$

матричното уравнение (2.3.10) може да се запише като система от блочни уравнения по следния начин:

$$\left|\begin{array}{lcl} A^{(k,1)}\mathbf{z}^{(k,1)} + A^{(k)}_{1,2}\mathbf{z}^{(k)}_{2^k} & = & \mathbf{f}^{(k,1)} \\ A^{(k)}_{2,1}\mathbf{z}^{(k,1)} + (T + b_{2^k,2^k}I_n)\mathbf{z}^{(k)}_{2^k} + A^{(k)}_{2,3}\mathbf{z}^{(k,2)} & = & \mathbf{f}^{(k)}_{2^k} \\ \ldots & & \\ A^{(k)}_{2s-1,2s-2}\mathbf{z}^{(k)}_{(s-1)2^k} + A^{(k,s)}\mathbf{z}^{(k,s)} + A^{(k)}_{2s-1,2s}\mathbf{z}^{(k)}_{s2^k} & = & \mathbf{f}^{(k,s)} \\ A^{(k)}_{2s,2s-1}\mathbf{z}^{(k,s)} + (T + b_{s.2^k,s.2^k}I_n)\mathbf{z}^{(k)}_{s2^k} + A^{(k)}_{2s,2s+1}\mathbf{z}^{(k,s+1)} & = & \mathbf{f}^{(k)}_{s2^k} \\ \ldots & & \\ A^{(k)}_{2^{l-k+1}-1,2^{l-k+1}-2}\mathbf{z}^{(k)}_{(2^{l-k}-1)2^k} + A^{(k,2^{l-k})}\mathbf{z}^{(k,2^{l-k})} & = & \mathbf{f}^{(k,2^{l-k})} \end{array}\right. .$$

Като се вземат от нея подсистемите с матрици $A^{(k,s)}$ или, което е едно и също, за някое дадено $s \geq 1$ и $s_k = (s-1)2^k$, (s_k+1)-то, (s_k+2)-то, ... , $(s_k + 2^k - 1)$-то уравнение от тъждеството

$$\mathbf{f}^{(k)} = A \sum_{k' \geq k} \mathbf{x}^{(k')},$$

се получава

$$\mathbf{f}^{(k,s)} = A^{(k,s)}\mathbf{z}^{(k,s)} + A^{(k)}_{2s-1,2s-2}\mathbf{z}^{(k)}_{(s-1)2^k} + A^{(k)}_{2s-1,2s}\mathbf{z}^{(k)}_{s2^k}$$

$$\Updownarrow$$

$$\begin{aligned} \mathbf{f}^{(k,s)} = & \quad A^{(k,s)} \left(\sum_{k' \geq k} \mathbf{x}^{(k')}_{s_k+r} \right)_{r=1}^{2^k-1} \\ & + A^{(k)}_{2s-1,2s-2} \left(\sum_{k' \geq k} \mathbf{x}^{(k')} \right)_{(s-1)2^k} \\ & + A^{(k)}_{2s-1,2s} \left(\sum_{k' \geq k} \mathbf{x}^{(k')} \right)_{s2^k} \end{aligned} . \qquad (2.3.11)$$

Тъй като $\mathbf{x}^{(k)}_{s_k} = 0$ и $\mathbf{x}^{(k)}_{s2^k} = 0$ и от (2.3.8) следва

$$\sum_{k'>k} \mathbf{x}^{(k')}_{s_k} = \mathbf{x}_{s_k}, \sum_{k'>k} \mathbf{x}^{(k')}_{s2^k} = \mathbf{x}_{s2^k},$$

то

$$\sum_{k' \geq k} \mathbf{x}^{(k')}_{s_k} = \mathbf{x}_{s_k} \text{ и } \sum_{k' \geq k} \mathbf{x}^{(k')}_{s2^k} = \mathbf{x}_{s2^k}.$$

Полага се

$$\mathbf{y}_r^{(k)} = \sum_{k'>k} \mathbf{x}_{s_k+r}^{(k')}$$

за $r = 1, \ldots, 2^k - 1$ и (2.3.11) добива вида

$$A^{(k,s)}\mathbf{y}^{(k)} = \mathbf{f}(k,s) - A^{(k)}_{2s-1,2s-2}\mathbf{x}_{s_k} - A^{(k)}_{2s-1,2s}\mathbf{x}_{s2^k} \,. \tag{2.3.12}$$

При $k > 1$ блоковете $A^{(k)}_{2s-1,2s-2}$ и $A^{(k)}_{2s-1,2s}$ са

$$A^{(k)}_{2s-1,2s-2} = A^{(k)}_{2(s-1)+1,2(s-1)} = \begin{pmatrix} b_{(s-1)2^k+1,(s-1)2^k} I_n \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

$$A^{(k)}_{2s-1,2s} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_{s2^k-1,s2^k} I_n \end{pmatrix},$$

а при $k = 1$ са

$$\begin{aligned} A^{(k)}_{2s-1,2s-2} &= b_{(s-1)2^k+1,(s-1)2^k} I_n \,, \\ A^{(k)}_{2s-1,2s} &= b_{s2^k-1,s2^k} I_n \,. \end{aligned}$$

Следователно (2.3.12) може да се запише във вида

$$A^{(k,s)}\mathbf{y}^{(k)} = \mathbf{f}^{(k,s)} + \hat{\mathbf{f}}^{(k,s)}, \tag{2.3.13}$$

където

$$\hat{\mathbf{f}}^{(k,s)} = \begin{cases} \begin{pmatrix} -b_{s_k+1,s_k}\mathbf{x}_{s_k} \\ 0 \\ \vdots \\ 0 \\ -b_{s2^k-1,s2^k}\mathbf{x}_{s2^k} \end{pmatrix}, & k \neq 1 \\ -b_{s_k+1,s_k}\mathbf{x}_{s_k} - b_{s2^k-1,s2^k}\mathbf{x}_{s2^k}, & k = 1 \end{cases}.$$

На всяка стъпка на обратния ход се решава непълно система от вида (2.3.13), като се търси само

$$\begin{aligned} \mathbf{y}^{(k)}_{2^{k-1}} &= \sum_{k'\geq k} \mathbf{x}^{(k')}_{(s-1)2^k+2^{k-1}} = \sum_{k'\geq k} \mathbf{x}^{(k')}_{(2s-2+1)2^{k-1}} \\ &= \sum_{k'\geq k} \mathbf{x}^{(k')}_{(2s-1)2^{k-1}} = \mathbf{x}_{(2s-1)2^{k-1}} \,. \end{aligned}$$

От друга страна, тъй като $\mathbf{x}^{(k,s)}$ е решение на системата

$$A^{(k,s)}\mathbf{x}^{(k,s)} = \mathbf{f}^{(k,s)} \,,$$

то $\mathbf{y}^{(k)}$ може да се разложи по следния начин

$$\mathbf{y}^{(k)} = \hat{\mathbf{y}}^{(k)} + \mathbf{x}^{(k,s)},$$

където $\hat{\mathbf{y}}^{(k)}$ е решение на системата

$$A^{(k,s)}\hat{\mathbf{y}}^{(k)} = \hat{\mathbf{f}}^{(k,s)}.$$

От тук следва, че

$$\mathbf{y}^{(k)}_{2^{k-1}} = \hat{\mathbf{y}}^{(k)}_{2^{k-1}} + \mathbf{x}^{(k,s)}_{2^{k-1}},$$

като $\mathbf{x}^{(k,s)}_{2^{k-1}}$ вече е намерен при правия ход.

Следователно задачата се редуцира до непълно решаване на задачи с разредени десни части, чиито матрици $A^{(k,s)}$ позволяват разделяне на променливите.

След всяка стъпка k на обратния ход са изчислени компонентите $\mathbf{x}_{s2^{k-1}}$, необходими за образуване на съответните десни части на стъпка $k-1$.

Индукцията за намиране блоковете на $\mathbf{x}$ е следната:

При $k = l$ се намира $\mathbf{x}_{2^{l-1}} = \mathbf{x}^{(l)}_{2^{l-1}}$.

Предполага се по индукция, че за някое k, $1 \leq k \leq l-1$ са намерени векторите $\mathbf{x}_{s2^{k-1}}$, $s = 1, \ldots, 2^{l-k} - 1$ на предишните стъпки $l, \ldots, k+1$.

След това на k-тата стъпка се намират само векторите $\mathbf{x}_{(2s-1)2^{k-1}}$ за $s = 1, \ldots, 2^{l-k}$ и заедно с блоковете, намерени на предишните стъпки, известните компоненти стават $\mathbf{x}_{s2^{k-1}}$ за $s = 1, \ldots, 2^{l-k+1} - 1$.

Сумират се тъждествата

$$\mathbf{f}^{(j+1)} = \mathbf{f}^{(j)} - A\mathbf{x}^{(j)}$$

за $j = k, k+1, \ldots, l$, полученото равенство

$$\mathbf{f}^{(k)} = A\left(\sum_{j \geq k} \mathbf{x}^{(j)}\right)$$

се представя във вида (2.3.12) и след като от дясната страна се изключат известните вектори $\mathbf{x}_{(s-1)2^k}$ и $\mathbf{x}_{s2^k}$ се получава системата, която трябва да се реши непълно на k-тата стъпка на обратния ход.

Забележка: За $k = 1$ получената система има една блочна компонента и се решава пълно.

С това описанието на обратния ход на алгоритъм FASV е завършено.

2.3.4 FASV-сложност на алгоритъма

Бързият алгоритъм за разделяне на променливите (FASV), въз основа на изложението в предишните две точки, може да бъде описан накратко по следния начин:

Алгоритъм 3 (FASV)

Стъпка 1 Прав ход:

Полага се $\mathbf{f}^{(1)} = \mathbf{f}$

for $k = 1$ to $l - 1$

for $s = 1$ to 2^{l-k} *се решават*

$$A^{(k,s)}\mathbf{x}^{(k,s)} = \begin{pmatrix} 0 \\ \mathbf{f}_{2^{k-1}}^{(k,s)} \\ 0 \end{pmatrix} \tag{2.3.14}$$

непълно, като се намират само

$$\mathbf{x}_1^{(k,s)},\ \mathbf{x}_{2^{k-1}}^{(k,s)},\ \mathbf{x}_{2^k-1}^{(k,s)}$$

end{цикъл по s}

for $s = 1$ to $2^{l-k} - 1$ *се изчисляват*

$$\mathbf{f}_{s2^k}^{(k+1)} = \mathbf{f}_{s2^k}^{(k)} - b_{s2^k,s2^k-1}\mathbf{x}_{2^k-1}^{(k,s)} - b_{s2^k,s2^k+1}\mathbf{x}_1^{(k,s+1)} \tag{2.3.15}$$

end{цикъл по s}

end{цикъл по k}

Стъпка 2 Обратен ход:

Решава се

$$A\mathbf{x}^{(l)} = \begin{pmatrix} 0 \\ \mathbf{f}_{2^{l-1}}^{(l)} \\ 0 \end{pmatrix} \tag{2.3.16}$$

непълно, само за $\mathbf{x}_{2^{l-1}}^{(l)} = \mathbf{x}_{2^{l-1}}$

for $k = l - 1$ downto 1

for $s = 1$ to 2^{l-k} непълно *се решават*

$$A^{(k,s)}\hat{\mathbf{y}}^{(k)} = \hat{\mathbf{f}}^{(k,s)} \tag{2.3.17}$$

само за $\hat{\mathbf{y}}_{2^{k-1}}^{(k)}$

Полага се

$$\mathbf{x}_{(2s-1)2^{k-1}} = \hat{\mathbf{y}}_{2^{k-1}}^{(k)} + \mathbf{x}_{2^{k-1}}^{(k,s)} \tag{2.3.18}$$

end{цикъл по s}

end{цикъл по k}

END {FASV}

Броят аритметични операции, необходими за изпълнение на правия и обратния ход и на целия алгоритъм FASV, се определя с помощта на **Теорема** 2.2.1 за сложността на алгоритъма за непълно решаване на задачи с разредени десни части и се дава от следните теореми:

Теорема 2.3.1 *Броят на аритметичните операции, необходими при правия ход на алгоритъма FASV е от следния порядък*

$$\sum_{k=1}^{l-1} 2^{l-k}(4\,n + (13\,n - 4)\,n_k) \sim 13\,n^2(l-1) - 9\,n^2$$

в частта за решаване и

$$\sum_{k=1}^{l-1} 2^{l-k}(n_k\ \textit{npделения} + 3\,n_k(n-1)\textit{други операции}) \sim n^2\,(l-1)\textit{деления} + 3\,n^2\,(l-1)\textit{други операции}$$

за факторизация на матриците $T + \lambda_k I_n$, *като* $n_k = 2^k - 1$.

Доказателство. Множителят 2^{l-k} дава броя на решаваните системи от вида (2.3.14). $4n$ идва от пресмятането на дясната страна $\mathbf{f}^{(k+1)}$ за всяка следваща стъпка посредством (2.3.15). Частта $(13n-4)\,n_k$ дава броя аритметични операции за непълно решаване на (2.3.14), намирайки само $\mathbf{x}_1^{(k,s)}$, $\mathbf{x}_{2^{k-1}}^{(k,s)}$ и $\mathbf{x}_{2^k-1}^{(k,s)}$, използвайки метода на разделяне на променливите въз основа на собствените вектори и собствените стойности на симетричните и положително определени матрици $B_k^{(s)}$ от ред $n_k = 2^k - 1$. Тази част се получава, като се приложи **Теорема** 2.2.1 при $r = 3$ (броя търсени компоненти на $\mathbf{x}^{(k,s)}$) и $d = 1$ (броя ненулеви компоненти на дясната част $\mathbf{f}^{(k,s)}$). ∎

Теорема 2.3.2 *Обратният ход на FASV изисква*

$$\sum_{k=1}^{l-1} 2^{l-k}(2\,n + (11\,n - 4)\,n_k) + 9\,n^2 - 4\,n \sim 11\,n^2(l-1)$$

аритметични операции.

Доказателство. Множителят 2^{l-k} е заради броя на задачите (2.3.17), $2\,n$ идва заради изчисляването на дясната страна на (2.3.17). Частта $(11\,n-4)n_k$ идва от непълното решаване на (2.3.17) и е пресметната с помощта на **Теорема** 2.2.1 за $r = 1$ (брой търсени компоненти на решението) и $d = 2$ (брой ненулеви компоненти на дясната част).

Последният израз $9\,n^2 - 4\,n$ дава цената за непълното решаване на задача (2.3.16) за $k = l$ със $d = 1$ и $r = 1$. ∎

Въз основа на тези две теореми, общата цена на FASV се дава с теоремата:

Теорема 2.3.3 *Задачата* $A\mathbf{x} = \mathbf{f}$ *може да бъде решена чрез бързия алгоритъм за разделяне на променливите (FASV) за*

$$24\,n^2(l-1) - 9\,n^2$$

операции в частта за решаване и

$$n^2(l-1)\ \textit{деления}\ + 3\,n^2(l-1)\ \textit{други операции}$$

за факторизация на всички матрици $T + \lambda_k I_n$ *в произведения от вида* $L_k D_k^{-1} U_k$, *където* L_k, U_k *са долнотриъгълни и горнотриъгълни с единици по главния диагонал, а* D_k *е диагонална матрица.*

За по-голяма точност, към предишната цена трябва да се включи и броят на операциите, необходими за пресмятане на всички собствени стойности и 1-та, 2^{k-1}-та, и $(2^k - 1)$-та компонента на всички собствени вектори на тридиагоналните матрици $B_k^{(s)}$ от ред n_k $(n_k = 2^k - 1)$ за $s = 1, 2, \ldots, 2^{l-k}$ и за всички $k = 1, \ldots, l$. Тези данни могат да бъдат намерени итеративно с машинна точност за

$$\sum_{k=1}^{l} 2^{l-k} \mathcal{O}\left(n_k^2\right) = \mathcal{O}\left(n^2\right)$$

аритметични операции.

За прилагане на алгоритъм FASV е необходима следната машинна памет:

- n^2 думи за дясната част $\mathbf{f}$;
- векторът на решението $\mathbf{x}$ заема същата памет както вектора $\mathbf{f}$;
- за всички собствени стойности и трите необходими компоненти на всеки от собствените вектори на матриците $B_s^{(k)}$ са необходими

$$4 \sum_{1 \le k \le l} 2^{l-k} n_k \sim 4nl \text{ думи;}$$

- за да се запазят всички матрици L_k , D_k^{-1} , U_k от разлагането на матриците $T + \lambda I$ са необходими два вектора с дължина n за всяка от тях или общо

$$2n \sum_{1 \le k \le l} 2^{l-k} n_k \sim 2n^2 l \text{ думи.}$$

Общо за съхраняване на необходимите данни при прилагането на алгоритъм FASV са нужни $2n^2 l + 4nl + 2n^2$ машинни думи.

Глава 3

Устойчиви марш алгоритми

В тази глава е описан марш алгоритъм за решаване на системи уравнения с разделящи се променливи от вида (1.3.5) $A\mathbf{x} = \mathbf{f}$, където $A = I_m \otimes T + B \otimes I_n$. Матриците $T = \text{tridiag}\,(t_{i,j})_{n\times n}$ и $B = \text{tridiag}\,(b_{i,j})_{m\times m}$ са тридиагонални, симетрични и положително полуопределени, като една от тях (T или B) е положително определена.

След това е описан устойчив марш алгоритъм, получен чрез ограничаване дължината на марш стъпките и използване на техниката за непълно решаване на задачи с разредени десни части. Получена е оценка за максималния брой аритметични операции, необходими за изпълнение на устойчивия марш алгоритъм.

3.1 Решаване на системи чрез блочна факторизация на матрици

Разглежда се система линейни алгебрични уравнения от вида

$$A\mathbf{x} = \mathbf{f},$$

където

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix},\ \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix},\ \mathbf{f} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}.$$

Векторите $\mathbf{x}_i$ и $\mathbf{f}_i$ са с размерности, съответстващи на размерността на $A_{i,i}$ за $i = 1, 2$.

Матрицата A се представя във вида

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1} & 0 \\ A_{2,1} & S \end{pmatrix} \begin{pmatrix} I & A_{1,1}^{-1} A_{1,2} \\ 0 & I \end{pmatrix},$$

където $S = A_{2,2} - A_{2,1} A_{1,1}^{-1} A_{1,2}$ е допълнението на Шур (Schur).

Трябва да се реши системата:

$$\begin{pmatrix} A_{1,1} & 0 \\ A_{2,1} & S \end{pmatrix} \begin{pmatrix} I & A_{1,1}^{-1} A_{1,2} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}.$$

Полага се

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} I & A_{1,1}^{-1} A_{1,2} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix}.$$

Тогава

$$\left| \begin{array}{rcl} A_{1,1}\mathbf{y}_1 & = & \mathbf{f}_1 \\ A_{2,1}\mathbf{y}_1 + S\mathbf{y}_2 & = & \mathbf{f}_2 \end{array} \right. .$$

Тази система се решава относно $\mathbf{y}$ и се получава

$$\left| \begin{array}{rcl} \mathbf{y}_1 & = & A_{1,1}^{-1}\mathbf{f}_1 \\ \mathbf{y}_2 & = & S^{-1}(\mathbf{f}_2 - A_{2,1}\, A_{1,1}^{-1}\mathbf{f}_1) \end{array} \right. . \tag{3.1.1}$$

При намерени $\mathbf{y}_1$ и $\mathbf{y}_2$, $\mathbf{x}_1$ и $\mathbf{x}_2$ се намират от системата

$$\left| \begin{array}{rcl} \mathbf{x}_1 + A_{1,1}^{-1}A_{1,2}\mathbf{x}_2 & = & \mathbf{y}_1 \\ \mathbf{x}_2 & = & \mathbf{y}_2 \end{array} \right. ,$$

откъдето

$$\left| \begin{array}{rcl} \mathbf{x}_2 & = & \mathbf{y}_2 \\ \mathbf{x}_1 & = & \mathbf{y}_1 - A_{1,1}^{-1}A_{1,2}\mathbf{y}_2 \end{array} \right. \tag{3.1.2}$$

като $\mathbf{y}_1$ и $\mathbf{y}_2$ се заместват от (3.1.1) в (3.1.2).

3.2 Основна идея

При стандартния марш алгоритъм, изходната матрица се пренарежда и записва в 2×2 блочна форма така, че единият от блоковете по главния диагонал е блочна горнотриъгълна матрица, а другият е нулев блок. С дясната страна се извършват съответни преобразувания, за да се стигне до еквивалентна система.

При решаването на получената система чрез блочна факторизация, е необходимо да се решат две системи с горнотриъгълна матрица и една система (с допълнението на Шур), чието решаване се свежда до непълно решаване на задача с разредена дясна част.

При обобщения марш алгоритъм, изходната система се разделя на блокове и се пренарежда така, че новите блокове с нечетни номера се записват в началото, а тези с четните - в края на новата матрица. При записване на новата матрица в блочна 2×2 форма

$$\begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{pmatrix},$$

блокът $\tilde{A}_{1,1}$ ще е блочнодиагонален. Получената система се решава чрез блочна факторизация. Системата с блок $\tilde{A}_{1,1}$ се разпада на системи с по-малка размерност, които се решават със стандартен марш алгоритъм. Системата, в която участва блокът $\tilde{A}_{2,2}$ се свежда до непълно решаване на система с разредена дясна част.

3.3 Стандартен марш алгоритъм - МА

При стандартния марш алгоритъм се извършва пренареждане на блочните компоненти на матрицата A и на дясната страна $\mathbf{f}$ на системата (1.3.5), като първото блочно уравнение се записва на последно място. Пренареденатa матрица се записва в 2×2 блочна форма и се получава следната еквивалентна задача:

$$\begin{pmatrix} U & G \\ C & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}' \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathbf{f}' \\ \mathbf{f}_1 \end{pmatrix}, \tag{3.3.3}$$

където U е горнотриъгълната матрица

$$U = \begin{pmatrix} b_{2,1}I_n & T + b_{2,2}I_n & b_{2,3}I_n & \dots & 0 \\ 0 & b_{3,2}I_n & T + b_{3,3}I_n & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & b_{m-1,m-2}I_n & T + b_{m-1,m-1}I_n \\ 0 & \dots & \dots & 0 & b_{m,m-1}I_n \end{pmatrix},$$

а матриците G и C са

$$G = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ b_{m-1,m}I_n \\ T + b_{m,m}I_n \end{pmatrix}, \; C = (T + b_{1,1}I_n, b_{1,2}I_n, 0, \dots, 0)\,.$$

Векторите $\mathbf{x}'$ и $\mathbf{f}'$ са

$$\mathbf{x}' = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{m-1} \end{pmatrix}, \mathbf{f}' = \begin{pmatrix} \mathbf{f}_2 \\ \mathbf{f}_3 \\ \vdots \\ \mathbf{f}_m \end{pmatrix}.$$

Пренареденатa матрица допуска следната блочна факторизация

$$\begin{pmatrix} U & G \\ C & 0 \end{pmatrix} = \begin{pmatrix} U & 0 \\ C & -CU^{-1}G \end{pmatrix} \begin{pmatrix} I & U^{-1}G \\ 0 & I \end{pmatrix}.$$

Тогава системата (3.3.3) добива вида:

$$\begin{pmatrix} U & 0 \\ C & -CU^{-1}G \end{pmatrix} \begin{pmatrix} I & U^{-1}G \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}' \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathbf{f}' \\ \mathbf{f}_1 \end{pmatrix}. \tag{3.3.4}$$

За решаването на тази система се постъпва по стандартния начин за решаване на системи чрез блочна факторизация (вж. т.3.1).

Полага се

$$\mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \begin{pmatrix} I & U^{-1}G \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}' \\ \mathbf{x}_m \end{pmatrix}$$

и за $\mathbf{y}$ се получава следната система

$$\left|\begin{array}{rcl} U\mathbf{y}_1 & = & \mathbf{f}' \\ C\mathbf{y}_1 - CU^{-1}G\mathbf{y}_2 & = & \mathbf{f}_1 \end{array}\right. . \tag{3.3.5}$$

При намерени $\mathbf{y}_1$, $\mathbf{y}_2$ за $\mathbf{x}_1$, $\mathbf{x}_2$ се стига до системата

$$\left|\begin{array}{rcl} \mathbf{x}' + U^{-1}G\mathbf{x}_m & = & \mathbf{y}_1 \\ \mathbf{x}_m & = & \mathbf{y}_2 \end{array}\right. . \tag{3.3.6}$$

От (3.3.5) следва, че $\mathbf{y}_1 = U^{-1}\mathbf{f}'$, откъдето за $\mathbf{y}_2$ се получава системата

$$-CU^{-1}G\mathbf{y}_2 = \mathbf{f}_1 - CU^{-1}\mathbf{f}' .$$

Но от (3.3.6) $\mathbf{y}_2 = \mathbf{x}_m$, откъдето

$$-CU^{-1}G\mathbf{x}_m = \mathbf{f}_1 - CU^{-1}\mathbf{f}' .$$

От (3.3.5) и (3.3.6) се намира последователно:

$$\begin{array}{l} U\mathbf{y}_1 = \mathbf{f}' \iff U(\mathbf{x}' + U^{-1}G\mathbf{x}_m) = \mathbf{f}' \\ \\ \quad \Rightarrow \quad U\mathbf{x}' + G\mathbf{x}_m = \mathbf{f}' \\ \quad \Rightarrow \quad U\mathbf{x}' = -G\mathbf{x}_m + \mathbf{f}' . \end{array}$$

За да се реши системата (3.3.3) трябва последователно да се решат системите:

$$\left|\begin{array}{l} U\mathbf{y}_1 = \mathbf{f}' \\ -CU^{-1}G\mathbf{x}_m = \mathbf{f}_1 - C\mathbf{y}_1 = \mathbf{f}_1 - CU^{-1}\mathbf{f}' \\ U\mathbf{x}' = -G\mathbf{x}_m + \mathbf{f}' \end{array}\right. . \tag{3.3.7}$$

Тъй като U е горнотриъгълна матрица, за решаването на системите с нея (от (3.3.7)) се използва проста рекурентна връзка.

Разглежда се системата

$$U\begin{pmatrix} \underline{\xi}_1 \\ \underline{\xi}_2 \\ \vdots \\ \underline{\xi}_{m-1} \end{pmatrix} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{m-1} \end{pmatrix}$$

или по компонентно

$$\begin{array}{rcl} b_{2,1}\underline{\xi}_1 + (b_{2,2}I_n + T)\underline{\xi}_2 + b_{2,3}\underline{\xi}_3 & = & \mathbf{g}_1 \\ \ddots & & \vdots \\ b_{m-1,m-2}\underline{\xi}_{m-2} + (b_{m-1,m-1}I_n + T)\underline{\xi}_{m-1} & = & \mathbf{g}_{m-2} \\ b_{m,m-1}\underline{\xi}_{m-1} & = & \mathbf{g}_{m-1} \end{array} .$$

Решението се получава чрез стандартния обратен ход:

$$\underline{\xi}_{m-1} = \frac{1}{b_{m,m-1}}\mathbf{g}_{m-1};$$
$$\text{for } i = m-1 \text{ down to } 2$$
$$\underline{\xi}_{i-1} = \frac{1}{b_{i,i-1}}\left(\mathbf{g}_{i-1} - b_{i,i}\underline{\xi}_i - T\underline{\xi}_i - b_{i,i+1}\underline{\xi}_{i+1}\right)$$
$$end\,.$$

Решаването на редуцираната система

$$-CU^{-1}G\mathbf{x}_m = \tilde{\mathbf{f}}_1 \equiv \mathbf{f}_1 - CU^{-1}\begin{pmatrix} \mathbf{f}_2 \\ \mathbf{f}_3 \\ \vdots \\ \mathbf{f}_m \end{pmatrix} \tag{3.3.8}$$

е еквивалентно на непълно решаване на система с изходната матрица A и разредена дясна част.

За компонентата $\tilde{\mathbf{x}}_m$ на решението на системата

$$A\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{3.3.9}$$

е в сила $\tilde{\mathbf{x}}_m = \mathbf{x}_m$, тъй като ако се извърши същата процедура с (3.3.9) както с изходната система, т.е. ако (3.3.9) се запише във вида (3.3.3) и се изпълни блочна елиминация, се завършва със същата система (3.3.8).

Системата (3.3.9) може да бъде решена чрез техниката за непълно решаване на система с разредена дясна част (вж. точка 2.2) с $r = 1$ и $d = 1$, т.е. като се търси само една блочна компонента ($\tilde{\mathbf{x}}_m$) на решението и дясната страна има една ненулева компонента ($\tilde{\mathbf{f}}_1$).

Оставащата част $\mathbf{x}'$ от решението $\mathbf{x}$ се получава отново чрез проста рекурсия решавайки система с U.

3.4 Обобщен марш алгоритъм - GMA

В Банк и Росе (Bank and Rose) [6] и Банк (Bank) [7] е показано, че за големи m марш алгоритъма е неустойчив. По-точно, рекурентната връзка, използвана за решаването на системи с горнотриъгълната матрица U, е неустойчива за големи m. По тази причина, стандартният марш алгоритъм е от практически интерес само ако дължината на рекурсията е малка, т.е. за $m \ll n$.

Когато $m = n$ или от същия порядък, за задачата (1.3.5) $A\mathbf{x} = \mathbf{f}$ може да се използва обобщения марш алгоритъм (GMA).

Нека $m = n$ и $n + 1 = p(k + 1)$ за някои цели k и p. Извършва се блочно четно-нечетно пренареждане на матрицата A и векторите $\mathbf{x}$ и $\mathbf{f}$ по следния начин:

Стълбовете в изходния блочен вид на A се групират в ленти, всяка от които е с по $k+1$ стълба ($p-1$ ленти с по $k+1$ стълба и 1 лента с k стълба). Първите k стълба от всяка лента образуват нечетните стълбове L_{2r-1}, а $k+1$-вите стълбове - четните стълбове L_{2r} на това групиране. Спрямо така направената блочна структура на матрицата, стълбовете с нечетни номера (L_{2r-1}) се подреждат в нарастващ ред преди стълбовете с четните (също подредени в нарастващ ред). Тази процедура се прави последователно със стълбовете и с редовете на матрицата A:

$$A = \begin{pmatrix} \vdots & \vdots & \vdots & & \vdots \\ L_1 & L_2 & L_3 & \dots & L_{2p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ \underbrace{}_{k} & \underbrace{}_{1} & \underbrace{}_{k} & & \underbrace{}_{k} \\ \text{стълба} & \text{стълб} & \text{стълба} & & \text{стълба} \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} S_1 & & & & \} & k \text{ реда} \\ S_2 & & & & \} & 1 \text{ ред} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ L_1 & L_3 & L_{2p-1} & L_2 & & L_{2p-2} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ S_{2p-2} & & & & \} & 1 \text{ ред} \\ S_{2p-1} & & & & \} & k \text{ реда} \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{array}{c} \\ S_1 \\ S_3 \\ S_5 \\ \vdots \\ S_{2p-1} \\ \\ S_2 \\ \vdots \\ S_{2p-2} \end{array} \begin{array}{c} \begin{array}{cccccccc} L_1 & L_3 & \dots & L_{2p-1} & & L_2 & \dots & L_{2p-2} \end{array} \\ \begin{pmatrix} A_1^{(k)} & & & 0 & \vdots & & & \\ & & & & & & & \\ & & \ddots & & \vdots & & \tilde{A}_{1,2} & \\ & & & & & & & \\ 0 & & & A_p^{(k)} & \vdots & & & \\ \dots & \dots & \dots & \dots & & \dots & & \dots \\ & & & & \vdots & & & \\ & & & & & & & \\ & \tilde{A}_{2,1} & & & \vdots & & \tilde{A}_{2,2} & \\ & & & & & & & \end{pmatrix} \end{array}.$$

Матрицата на системата придобива вида:

$$\tilde{A} = \begin{pmatrix} A_1^{(k)} & & 0 & \vdots & \\ & \ddots & & \vdots & \tilde{A}_{1,2} \\ 0 & & A_p^{(k)} & \vdots & \\ \cdots & \cdots & \cdots & & \\ & \tilde{A}_{2,1} & & & \tilde{A}_{2,2} \end{pmatrix} = \begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{pmatrix},$$

където съответните блокове са дефинирани чрез

$$\begin{aligned} \tilde{A}_{1,1} &= \text{blockdiag}\,(A_s^{(k)})_{s=1}^{p}, \\ A_s^{(k)} &= I_k \otimes T + B_s^{(k)} \otimes I_n, \\ B_s^{(k)} &= \text{tridiag}\,(b_{k_s+i,k_s+i-1}\,, b_{k_s+i,k_s+i}\,, b_{k_s+i,k_s+i+1})_{i=1}^{k}\,, \\ \tilde{A}_{2,2} &= \text{blockdiag}\,(T + b_{k_{s+1},k_{s+1}}\, I_n),\ k_s = (s-1)(k+1), s = 1, \ldots, p\,. \end{aligned}$$

Компонентите на вектора на решението $\mathbf{x}$ и дясната част $\mathbf{f}$ се групират и пренареждат съобразно с матрицата A и позволяват следните блочни форми:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix},\ \mathbf{x}^{(1)} = \begin{pmatrix} \mathbf{x}_1^{(1)} \\ \vdots \\ \mathbf{x}_p^{(1)} \end{pmatrix},\ \mathbf{x}_s^{(1)} = \begin{pmatrix} \mathbf{x}_{(s-1)(k+1)+1} \\ \vdots \\ \mathbf{x}_{(s-1)(k+1)+k} \end{pmatrix},\ s = 1, \ldots, p\,,$$

$$\mathbf{f} = \begin{pmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \end{pmatrix},\ \mathbf{f}^{(1)} = \begin{pmatrix} \mathbf{f}_1^{(1)} \\ \vdots \\ \mathbf{f}_p^{(1)} \end{pmatrix},\ \mathbf{f}_s^{(1)} = \begin{pmatrix} \mathbf{f}_{(s-1)(k+1)+1} \\ \vdots \\ \mathbf{f}_{(s-1)(k+1)+k} \end{pmatrix},\ s = 1, \ldots, p\,;$$

$$\mathbf{x}^{(2)} = \begin{pmatrix} \mathbf{x}_{k+1} \\ \vdots \\ \mathbf{x}_{s(k+1)} \\ \vdots \\ \mathbf{x}_{(p-1)(k+1)} \end{pmatrix},\ \mathbf{f}^{(2)} = \begin{pmatrix} \mathbf{f}_{k+1} \\ \vdots \\ \mathbf{f}_{s(k+1)} \\ \vdots \\ \mathbf{f}_{(p-1)(k+1)} \end{pmatrix}.$$

Тогава задачата придобива вида

$$\begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \end{pmatrix}.$$

Чрез блочна факторизация се получава

$$\begin{pmatrix} \tilde{A}_{1,1} & 0 \\ \tilde{A}_{2,1} & S \end{pmatrix} \begin{pmatrix} I & \tilde{A}_{1,1}^{-1}\,\tilde{A}_{1,2} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \end{pmatrix},$$

където $S = \tilde{A}_{2,2} - \tilde{A}_{2,1}\,\tilde{A}_{1,1}^{-1}\,\tilde{A}_{1,2}$.

След въвеждане на ново неизвестно

$$\begin{pmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \end{pmatrix} = \begin{pmatrix} I & \tilde{A}_{1,1}^{-1}\,\tilde{A}_{1,2} \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix}$$

и извършване на преобразувания, аналогични на тези в т. 3.1 и т. 3.3 задачата се свежда до решаване на системите

$$\left|\begin{array}{l} \tilde{A}_{1,1}\mathbf{y}^{(1)} = \mathbf{f}^{(1)} \\ \tilde{A}_{2,2} - \tilde{A}_{2,1}\,\tilde{A}_{1,1}^{-1}\,\tilde{A}_{1,2}\mathbf{x}^{(2)} = \mathbf{f}^{(2)} - \tilde{A}_{2,1}\,\tilde{A}_{1,1}^{-1}\,\mathbf{f}^{(1)} \\ \tilde{A}_{1,1}\mathbf{x}^{(1)} = \mathbf{f}^{(1)} - \tilde{A}_{1,2}\mathbf{x}^{(2)} \end{array}\right. . \tag{3.4.10}$$

Тъй като матрицата $\tilde{A}_{1,1}$ е блочнодиагонална, решаването на система с нея се свежда до решаване на p системи с матрици $A_s^{(k)}$, $s = 1, \ldots, p$. Системите с матрици блоковете $A_s^{(k)}$ на $\tilde{A}_{1,1}$ се решават използвайки стандартния марш алгоритъм.

Дължината на рекурентната връзка, необходима за решаване на системи с горнотриъгълни блокове е $k - 1 = \frac{n+1}{p} - 2$ и може да се контролира чрез избиране на достатъчно голямо p.

Решаването на редуцираната система

$$\tilde{A}_{2,2} - \tilde{A}_{2,1}\,\tilde{A}_{1,1}^{-1}\,\tilde{A}_{1,2}\mathbf{x}^{(2)} = \tilde{\mathbf{f}}^{(2)} \equiv \mathbf{f}^{(2)} - \tilde{A}_{2,1}\,\tilde{A}_{1,1}^{-1}\,\mathbf{f}^{(1)} \tag{3.4.11}$$

е еквивалентно на непълното решаване на система с първоначалната матрица A и с разредена дясна част. Затова втората стъпка на блочната факторизация е решаването на системата

$$A\hat{\mathbf{x}} = \hat{\mathbf{f}}, \text{ където } \hat{\mathbf{f}}_i = \begin{cases} \tilde{\mathbf{f}}_s^{(2)}, & i = s(k+1) \\ 0, & i \neq s(k+1) \end{cases} , \tag{3.4.12}$$

търсейки само $\hat{\mathbf{x}}_{s(k+1)} = \mathbf{x}_{s(k+1)}$, $s = 1, \ldots, p-1$.

Тази задача може да се реши с FASV за $r = p-1$ (брой търсени компоненти на решението) и $d = p - 1$ (брой ненулеви компоненти на дясната част).

При намерени $\mathbf{x}^{(2)}$ и $\mathbf{y}^{(1)}$, $\mathbf{x}^{(1)}$ се намира решавайки съответната система от (3.4.10) с матрица $\tilde{A}_{1,1}$, т.е. чрез решаване на p системи с матрици $A_s^{(k)}$, $s = 1, \ldots, p$.

3.5 Сложност на алгоритмите MA и GMA

Въз основа на казаното в т.3.3, стандартният марш алгоритъм може да бъде описан накратко по следния начин:

Алгоритъм 4 (Марш алгоритъм - MA)

Стъпка 1 *Пренареждане* на блоковете на матрицата A и записване на системата във вида

$$\begin{pmatrix} U & G \\ C & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}' \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathbf{f}' \\ \mathbf{f}_1 \end{pmatrix} .$$

Стъпка 2 *Решаване* на системата

$$U\mathbf{y}_1 = \mathbf{f}'$$

чрез стандартния обратен ход.

Стъпка 3 *Непълно решаване* на системата

$$A\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

намирайки само $\tilde{\mathbf{x}}_m$ (т.е. с $r = 1$ и $d = 1$) с цел да се получи решението $\mathbf{x}_m$ на системата

$$-CU^{-1}\, G\mathbf{x}_m = \mathbf{f}_1 - C\mathbf{y}_1 = \mathbf{f}_1 - CU^{-1}\mathbf{f}'\,.$$

Стъпка 4 *Решаване* на

$$U\mathbf{x}' = -G\mathbf{x}_m + \mathbf{f}'$$

чрез стандартната рекурентна връзка на обратния ход.

За рекурентната връзка, необходима за решаване на системи с горнотриъгълната матрица U, са необходими $\mathcal{O}(nm)$ аритметични операции. Съгласно **Теорема** 2.2.1 за непълно решаване на системата в **Стъпка 3** се изисква същия порядък аритметични операции.

Така се обосновава верността на следната

Теорема 3.5.1 *Марш алгоритъмът, в комбинация с техниката за разделяне на променливите за непълно решаване на редуцираната система, изисква оптимален брой аритметични операции за решаване на задачи* $A\mathbf{x} = \mathbf{f}$ *с разделящи се променливи, а именно* $\mathcal{O}(nm)$ *операции.*

Съгласно точка 3.4, обобщеният марш алгоритъм накратко може да бъде записан във вида:

Алгоритъм 5 (Обобщен марш алгоритъм - GMA)

Стъпка 1 *Пренареждане* на матрицата и записване на системата в 2×2 блочна форма

$$\begin{pmatrix} \tilde{A}_{1,1} & \tilde{A}_{1,2} \\ \tilde{A}_{2,1} & \tilde{A}_{2,2} \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{pmatrix} = \begin{pmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \end{pmatrix},$$

$$\tilde{A}_{1,1} = \text{blockdiag}\,(A_s^{(k)})_{s=1}^{p}\,.$$

Стъпка 2 for $s = 1$ to p *решаване* на системата

$$A_s^{(k)} \mathbf{y}_s^{(1)} = \mathbf{f}_s^{(1)}$$

end

Стъпка 3 *Непълно решаване* на системата

$$A\hat{\mathbf{x}} = \hat{\mathbf{f}}, \text{ където } \hat{\mathbf{f}}_i = \begin{cases} \tilde{\mathbf{f}}_s^{(2)}, & i = s(k+1) \\ 0, & i \neq s(k+1) \end{cases},$$

намирайки само $\hat{\mathbf{x}}_{s(k+1)} = \mathbf{x}_{s(k+1)},\ s = 1, \ldots, p-1,$

$$\tilde{\mathbf{f}}^{(2)} \equiv \mathbf{f}^{(2)} - \tilde{A}_{2,1}\, \tilde{A}_{1,1}^{-1}\, \mathbf{f}^{(1)},$$

с цел намиране на компонентата $\mathbf{x}^{(2)}$.

Стъпка 4 *Намиране* на компонентата $\mathbf{x}^{(1)}$ на решението чрез

for $s = 1$ to p *решаване* на системата

$$A_s^{(k)} \mathbf{x}_s^{(1)} = \tilde{\mathbf{f}}_s^{(1)}$$

end

$$\tilde{\mathbf{f}}^{(1)} = \mathbf{f}^{(1)} - \tilde{A}_{1,2} \mathbf{x}^{(2)}.$$

Предполага се за определеност, че $p = 2^l$. Тогава цената за непълно решаване на системата в **Стъпка 3** чрез алгоритъма FASV се дава със следното твърдение:

Теорема 3.5.2 *Втората стъпка на блочната гаусова елиминация въз основа на непълното решаване на задача (3.4.12), използвайки алгоритъм FASV, изисква*

$$24n^2 l - 9n^2$$

операции.

Доказателство. Това е директно следствие от Теорема 2.3.3, тъй като за непълно решаване на (3.4.12) трябва да се извършат l стъпки на FASV. ∎

Забележка: Обобщеният марш алгоритъм във вида представен в Банк [7] изисква за втората стъпка на блочната гаусова елиминация $28n^2 l$ операции. Това показва, че алгоритъмът представен тук има асимптотически малко по-малък брой операции.

Като се има предвид, че при всяка от стъпки 2 и 4 на **Алгоритъм 5** се прилага p пъти стандартния марш алгоритъм, броят аритметични операции за тяхното изпълнение е $\mathcal{O}(npk)$. Следователно общият брой аритметични операции, необходими за изпълнението на обобщения марш алгоритъм е

$$\mathcal{O}(npk) + 24n^2 l - 9n^2 = \mathcal{O}\left(n^2\right) + 24n^2 \log_2(p) - 9n^2.$$

Глава 4

Числени експерименти

Алгоритмите SV и FASV, описани в глава 2, са реализирани с MATLAB. Числените експерименти за двата примера, формулирани в точка 4.1 по-долу, са направени на машини SUN SPARC station 20 и SUN Enterprise 3000. В тази глава са анализирани получените резултати и в края на главата е направен извод за целесъобразността от използването на MATLAB за решаване на задачи от разглеждания вид.

4.1 Формулировка на примерите за числени експерименти

Пример 1. Приложение за случая $a_1(x_1) = a_2(x_2) \equiv 1$.

Разглежда се задачата на Дирихле за уравнението на Поасон :

$$\left| \begin{array}{rcll} -\Delta u(x_1,x_2) & = & f(x_1,x_2), & (x_1,x_2) \in \Omega = (0,1)\times(0,1) \\ u(x_1,x_2) & = & 0, & (x_1,x_2) \in \partial\Omega \end{array} \right. .$$

За решение u се избира

$$u(x_1,x_2) = \sin(\pi x_1)\sin(\pi x_2)\,,$$

откъдето следва, че дясната част $f(x_1,x_2)$ е

$$f(x_1,x_2) = 2\pi^2 \sin(\pi x_1)\sin(\pi x_2)\,.$$

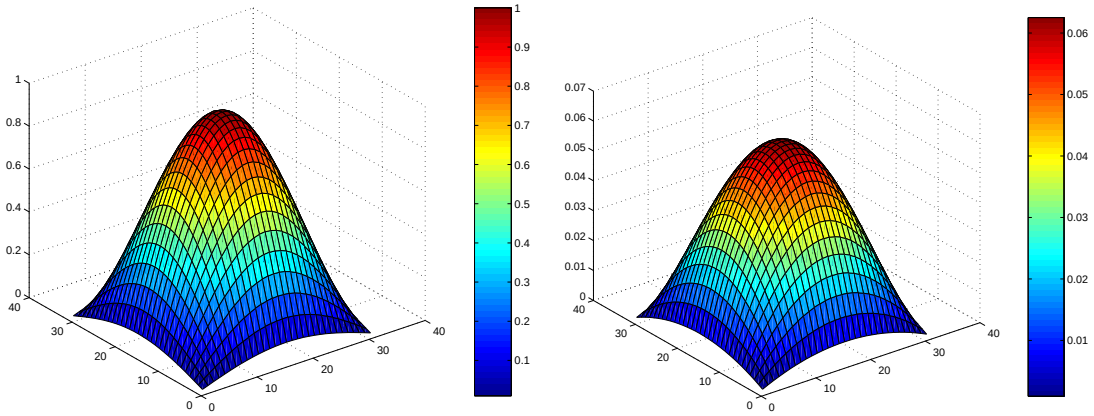
Пример 2. Приложение за случая $a_1(x_1) \neq 1$ и $a_2(x_2) \neq 1$.

Разглежда се задача (1.1.1), с коефициенти

$$a_1(x_1) = 1 + x_1^2, \qquad a_2(x_2) = e^{-x_2},$$

а за решението u се взима функцията

$$u(x_1,x_2) = (1-x_1)x_1x_2(1-x_2).$$



(а) пример 1 (б) пример 2

Фигура 4.1: Точно решение

Следователно дясната страна $f(x_1, x_2)$ е

$$f(x_1, x_2) = 2x_2(1 - x_2)(3x_1^2 - x_1 + 1) + e^{-x_2}x_1(1 - x_1)(3 - 2x_2)\,.$$

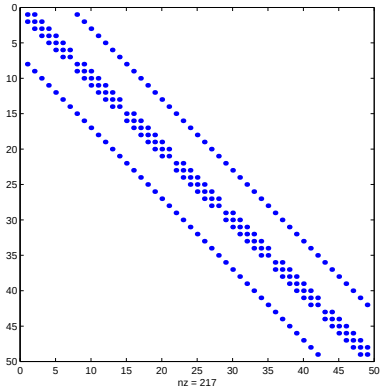
На фиг. 4.1 е показана графика на точното решение за **Пример 1** и **Пример 2**.

И в двата случая дискретизацията е направена върху равномерна мрежа $n \times n$ ($m = n$, $h_1 = h_2 = h$) с помощта на диференчна апроксимация по пет точков шаблон „кръст“.

Възлите на мрежата са

$$(x_1^i, x_2^j) = (i\,h, j\,h),\ i, j = 1, \ldots, n,\ h = \frac{1}{n+1}.$$

Получената система е от n^2 уравнения за n^2 неизвестни и за различните стойности на n, нейната матрица има един и същи разреден вид. На фиг.4.2 е показана структурата на матрицата A в случая $n = 7$.



Фигура 4.2: Матрица на системата

4.2 Сравнителен анализ на изчислителната сложност

Тъй като методите SV и FASV са директни, то получената грешка при решаване на системата $A\mathbf{x} = \mathbf{f}$ трябва да е грешката на диференчната схема, т.е. $\mathcal{O}\left(|h|^2\right)$ (вж. глава 1). С други думи, при намаляване на стъпката два пъти, грешката трябва да намалява четири пъти.

Времето за изпълнение на методите SV и FASV, за разглеждания случай $n = m$, зависи от броя операции, дадени в таблица 4.1, т.е. от асимптотиката на изчислителната сложност, посочена в съответните теореми в глава 2.

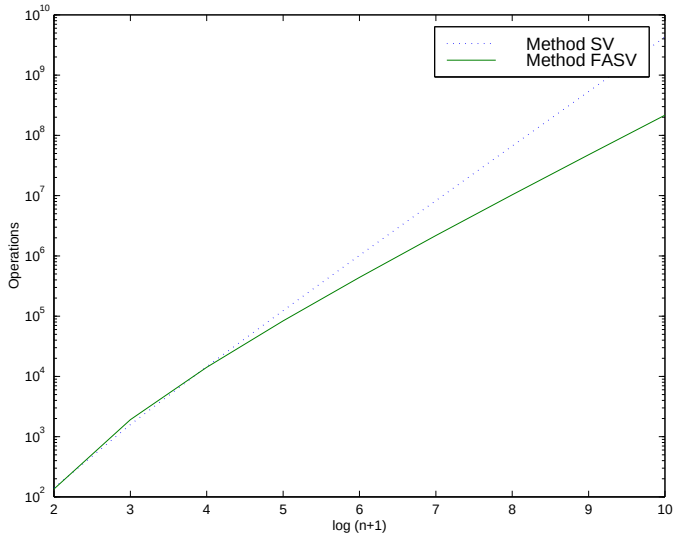
Таблица 4.1: Методи за разделяне на променливите

Метод	Брой операции
SV	$4n^3 + 5n^2$
FASV	$24n^2\left(\log_2\left(n+1\right) - 1\right) - 9n^2$

Графиката на данните от таблица 4.1, представена с фиг. 4.3, показва, че за малки стойности на l ($n = 2^l - 1$), т.е. за $l < 4$, броят операции за изпълнение на FASV е по-голям от този за изпълнение на SV, а при $l \geq 4$ FASV става с по-малък брой операции. Това означава, че за малки l, времето за изпълнение на FASV трябва да е по-голямо от времето за изпълнение на SV.

4.3 Резултати от числените експерименти

Всяка от задачите, формулирани като **Пример 1** и **Пример 2** в точка 4.1, е решена както чрез алгоритъм SV, така и чрез алгоритъм FASV за седем различни стойности на $n = 2^l - 1$ (за $l = 4, \ldots, 10$).



Фигура 4.3: Сравнение между методите SV и FASV

Резултатите от проведените числени експерименти за всеки от примерите са представени в таблици със следните колони:

n — брой възли на мрежата по всяко от направленията x_1 и x_2.

h — стъпка на мрежата по всяко от направленията.

L_2-грешка — L_2 норма на вектора на грешката.

макс. грешка — максималният по абсолютна стойност елемент на вектора на грешката.

cpu — cpu време за изпълнение на съответния алгоритъм, като в него не е включено времето за дискретизация на задачата, за намиране на собствените вектори и стойности и за факторизация на матриците.

cpu_eig — cpu време за намиране на необходимите собствени вектори и собствени стойности.

Резултатите за **Пример 1** са дадени в таблица 4.2, а тези за **Пример 2** — в таблица 4.3.

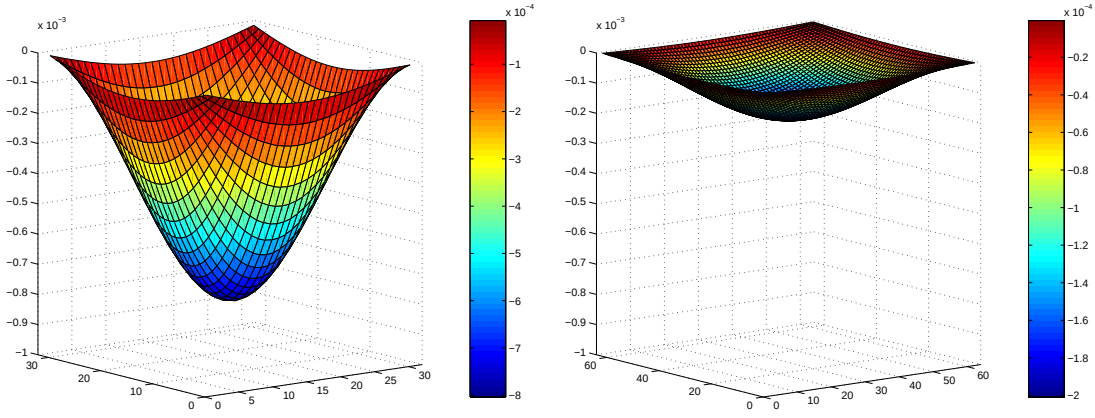
На фиг.4.4 и фиг.4.5 съответно за **Пример 1** и **Пример 2** е показано как намалява грешката при намаляване на стъпката два пъти, именно както L_2-грешката, така и максималната грешка намаляват четири пъти (това се вижда и от таблиците с резултатите).

Времето за изпълнение на SV първоначално (за $n \leq 511$) се увеличава квадратично и едва при $n = 1023$ започва да се променя според функцията

$$\alpha_1 n^3 + \alpha_2 n^2.$$

Таблица 4.2: Резултати за пример 1

n	h	cpu	cpu_eig	L_2-грешка	Макс. грешка
Метод SV					
15	1/16	0.07	0.00	1.6095e-03	3.2190e-03
31	1/32	0.27	0.01	4.0179e-04	8.0358e-04
63	1/64	1.11	0.06	1.0041e-04	2.0082e-04
127	1/128	4.95	0.43	2.5100e-05	5.0201e-05
255	1/256	24.22	3.44	6.2750e-06	1.2550e-05
511	1/512	149.93	29.01	1.5687e-06	3.1375e-06
1023	1/1024	1134.90	324.73	3.9222e-07	7.8443e-07
Метод FASV					
15	1/16	0.30	0.00	1.6095e-03	3.2190e-03
31	1/32	1.20	0.03	4.0179e-04	8.0358e-04
63	1/64	5.79	0.11	1.0041e-04	2.0082e-04
127	1/128	25.96	0.96	2.5100e-05	5.0201e-05
255	1/256	125.19	5.68	6.2750e-06	1.2550e-05
511	1/512	605.40	46.34	1.5687e-06	3.1375e-06
1023	1/1024	3155.70	485.46	3.9220e-07	7.8443e-07

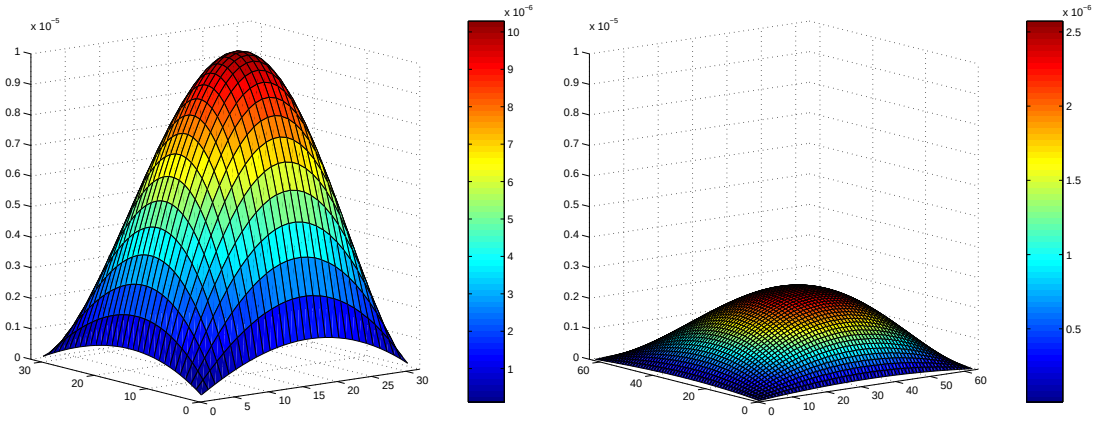


(а) $n = 31$ (б) $n = 63$

Фигура 4.4: Грешка за пример 1

Таблица 4.3: Резултати за пример 2

n	h	cpu	cpu_eig	L_2-грешка	Макс. грешка
Метод SV					
15	1/16	0.08	0.00	2.1587e-05	4.1066e-05
31	1/32	0.27	0.01	5.3960e-06	1.0290e-05
63	1/64	1.07	0.06	1.3489e-06	2.5727e-06
127	1/128	4.79	0.43	3.3723e-07	6.4340e-07
255	1/256	25.35	3.13	8.4308e-08	1.6085e-07
511	1/512	145.23	29.68	2.1077e-08	4.0213e-08
1023	1/1024	1383.00	537.06	5.2716e-09	1.0058e-08
Метод FASV					
15	1/16	0.41	0.00	2.1587e-05	4.1066e-05
31	1/32	1.18	0.03	5.3960e-06	1.0290e-05
63	1/64	5.45	0.14	1.3489e-06	2.5727e-06
127	1/128	25.62	0.87	3.3723e-07	6.4340e-07
255	1/256	123.44	5.63	8.4307e-08	1.6085e-07
511	1/512	600.80	48.00	2.1077e-08	4.0213e-08
1023	1/1024	3078.00	685.93	5.2716e-09	1.0058e-08



(а) $n = 31$ (б) $n = 63$

Фигура 4.5: Грешка за пример 2

Времето за изпълнение на FASV, при увеличаване на n, расте съобразно с функцията

$$\beta_1 n^2 \log n + \beta_2 n^2.$$

При сравняване на времената за изпълнение на SV и FASV се вижда, че въпреки че необходимият брой операции за изпълнение на FASV е по-малък от този за изпълнение на SV за $l \geq 4$, времето за изпълнение на FASV е по-голямо. Това се дължи на факта, че при реализацията са използвани готови програми на MATLAB за намиране на собствени вектори и собствени стойности и за LU факторизация на матрици, както и специални структури за съхраняване на матриците $A^{(k,s)}$ и векторите $\mathbf{f}^{(k,s)}$ и $\mathbf{x}^{(k,s)}$.

Проследявайки по какъв начин се изменят времената за изпълнение на SV и FASV, може да се предположи, че при $n \geq 4095$ времето за изпълнение на FASV ще е по-малко от това за изпълнение на SV. Това е трудно да се провери, тъй като средата на MATLAB, веднъж заела голямо количество памет, при изчистване на ненужните променливи не освобождава по-голямата част от паметта.

При сравняване на резултатите от изпълнението на FASV за **Пример 1** с тези получени от T. Rossi [3] (таблица 4.4) за същия пример се вижда, че грешката е почти същата, но с MATLAB времето за изпълнение е много по-голямо.

Таблица 4.4: Резултати за пример 1 на T. Rossi

n	h	cpu	Макс. грешка
31	1/32	0.006	8.04e-04
63	1/64	0.027	2.01e-04
127	1/128	0.124	5.02e-05
255	1/256	0.619	1.25e-05
511	1/512	2.841	3.14e-06
1023	1/1024	12.870	7.84e-07
2047	1/2047	58.500	1.96e-07

Тъй като най-често решаването на елиптични задачи с разделящи се променливи (чрез метода FASV) се използва на някои от стъпките за решаването на по-сложни задачи, факторът време за намиране на решението е от особено значение. Получените времена показват, че не е препоръчително да се използва MATLAB за решаване на големи и сложни задачи.

Библиография

[1] O. Axelson, P. Vassilevski, *"Solutin of equations R^n (part II) Iterative solution of linear systems"*, in: P. Ciarlet, J. Lions(Eds), Handbook on Numerical Methods, North Holland,**1996**, to appear.

[2] Sv. Petrova, *"Parallel implementation of fast elliptic solver"*, Parallel Computing 23(**1997**) 1113-1128.

[3] T. Rossi, *"Fictitious domain methods with separable preconditioners"*, University of Jyväskylä,Department of mathematics, report 69, **1995**.

[4] А. А. Самарский, *"Теория разностных схем"*, Москва, "Наука",**1983**.

[5] Б. Н. Парлет, *"Симетричная проблема собственных значений, Численные методы"*, Москва, "Мир",**1983**.

[6] R. E. Bank and D. J. Rose, *"Marching algorithms for elliptic boundary value problems. I: The constant coefficient case "*, SIAM J. Numer.Anal.,14 (**1977**) pp.792-829.

[7] R. E. Bank , *"Marching algorithms for elliptic boundary value problems. II: The variable coefficient case "*, SIAM J. Numer.Anal.,14 (**1977**) pp.950-970.

Приложение

Програми на MATLAB за реализация на методите SV и FASV за числено решаване на елиптични уравнения с разделящи се променливи.