VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra Matematiky

Řešení okrajových úloh metodou fiktivních oblastí

2009 Adam Zdráhala

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5.května 2009 Adam Zdráhala

Poděkování

Na tomto místě bych rád poděkoval panu Doc. Ing. Tomáši Kozubkovi, Ph.D., za trpělivost a podnětné připomínky k této diplomové práci.

Abstrakt

Tato diplomová práce rozebírá možnosti využití rychlých řešičů založených na fourierových transformacích pro řešení eliptických okrajových úloh ve 2D. Využijeme k tomu metodu fiktivních oblastí, díky které mají matice soustav, vzniklé diskretizací metodou konečných prvků, jednoduchou strukturu
Klíčová slova: metoda konečných prvků, fourierova transformace, sinová transformace, metoda fiktivních oblastí, lagrangeovy multiplikátory

Abstract

This thesis investigate possibilities of fast solvers of 2D elliptic boundary value problem based on Fourier transformation. The requirement of specific stiffness matrix structure is accomplished by fictitious domain method.
Key words: finite element method, fourier transform, sine transform, fictitiouse domain method, lagrangian multipliers

Obsah

Úvod

Tato práce se zaměřuje na efektivní implementaci metody fiktivních oblastí. Přestože metoda konečných prvků je efektivní způsob řešení parciálních diferenciálních rovnic, při požadavku na velkou přesnost řešení vznikají obrovské soustavy, jejichž výpočet je velice náročný. Existuje několik způsobů jak toto zlepšit. Jedním z nich je právě metoda fiktivních oblastí. Její princip spočívá v záměně problému formulovaného na oblasti se složitou geometrií (ω) za problém formulovaný na oblasti s pravidelnou geometrií (Ω)(např. obdelník) obsahující původní oblast a jež je s původní úlohou určitým způsobem svázána. A sice jeho řešení zúžené na původní oblast je řešením původního problému. Informace o geometrii oblasti je v našem případě „zakódována“ pomocí lagrangeovských multiplikátorů. Díky tomuto přístupu má matice tuhosti speciální vlastnosti, které lze úspěšně využít pro řešení pomocí rychlých řešičů založených na fourierových transformacích.

Kapitola 1

Použité nástroje

1.1 Operace s maticemi

Násobení skalárem

$$\mathbf{A} \in \mathbb{R}^{m\times n},\ \alpha \in \mathbb{R}$$

Pak

$$[\alpha\mathbf{A}]_{ij} = \alpha\,[\mathbf{A}]_{ij} \qquad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n \tag{1.1}$$

Příklad

$$5\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix}$$

Součet matic

Sčítat lze pouze matice stejného typu, tj. $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$ Pak

$$[\mathbf{A} + \mathbf{B}]_{ij} = [\mathbf{A}]_{ij} + [\mathbf{B}]_{ij} \qquad i = 1, 2 \ldots, m,\ j = 1, 2, \ldots, n \tag{1.2}$$

Příklad

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 4 & 6 \end{pmatrix}$$

Pro sčítání matic a násobení matic skalárem platí následující pravidla.Nechť jsou dány matice $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{m\times n}$ a skaláry $\alpha, \beta \in \mathbb{R}$. Pak platí

$$\begin{aligned} \mathbf{A} + (\mathbf{B} + \mathbf{C}) &= (\mathbf{A} + \mathbf{B}) + \mathbf{C} \\ \mathbf{A} + \mathbf{B} &= \mathbf{B} + \mathbf{A} \\ \alpha(\mathbf{A} + \mathbf{B}) &= \alpha\mathbf{A} + \alpha\mathbf{B} \\ (\alpha + \beta)\mathbf{A} &= \alpha\mathbf{A} + \beta\mathbf{A} \\ \alpha(\beta\mathbf{A}) &= (\alpha\beta)\mathbf{A} \\ 1\mathbf{A} &= \mathbf{A} \end{aligned} \tag{1.3}$$

Násobení matic

Nechť je dána matice $\mathbf{A} \in \mathbb{R}^{m\times p}$ a $\mathbf{B} \in \mathbb{R}^{p\times n}$. Pak

$$[\mathbf{AB}]_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{ip}b_{pj}. \tag{1.4}$$

Nechť jsou dány matice $\mathbf{ABC} \in \mathbb{R}^{m\times n}$ a skalár α.Platí následující pravidla vždy, když jsou definovány jednotlivé operace:

$$\begin{aligned} \mathbf{A}(\alpha\mathbf{B}) &= (\alpha\mathbf{A})\mathbf{B} \\ \mathbf{A}(\mathbf{B}+\mathbf{C}) &= \mathbf{AB}+\mathbf{AC} \\ (\mathbf{A}+\mathbf{B})\mathbf{C} &= \mathbf{AC}+\mathbf{BC} \end{aligned} \tag{1.5}$$

Rovněž platí asociativní zákon:

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$$

Obecně ovšem neplatí komutativní zákon,tj.

$$\mathbf{AB} \neq \mathbf{BA}$$

a to i v případě, že **AB** i **BA** jsou definované.

Důležitou maticí je takzvaná jednotková matice. Označuje se nejčastěji **I** nebo **E**. Jednotková matice má tvar:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Pro násobení jednotkové matice **I** s libovolnou maticí **A** platí (opět pokud jsou tyto operace definované):

$$\begin{aligned} \mathbf{AI} &= \mathbf{A} \\ \mathbf{IA} &= \mathbf{A} \end{aligned}$$

Transponování matic

Při této operaci se prohodí řádky za sloupce.

$$[\mathbf{A}^T]_{ij} = [\mathbf{A}]_{ji}$$

Pro transponování matic platí následující pravidla

$$\begin{aligned}(\mathbf{A}+\mathbf{B})^T &= \mathbf{A}^T+\mathbf{B}^T \\ (\alpha\mathbf{A})^T &= \alpha(\mathbf{A}^T) \\ (\mathbf{AB})^T &= \mathbf{B}^T\mathbf{A}^T.\end{aligned} \tag{1.6}$$

Sčítání, násobení, transponování blokových matic

Bloková matice

$$\begin{pmatrix}\mathbf{B} & \mathbf{C}\\ \mathbf{D} & \mathbf{E}\end{pmatrix} \tag{1.7}$$

Bloková matice se často používá v případech, že matice má po blocích nějaké zajímavé vlastnosti, kterých se dá využít pro efektivní výpočty. Například blok $\mathbf{B}$ matice 1.7 může být diagonální matice, blok E nulová matice atd.

Sčítání blokových matic

Mějme dány blokové matice

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\\ \mathbf{A}_3 & \mathbf{A}_4\end{pmatrix},\quad \mathbf{B}=\begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\\ \mathbf{B}_3 & \mathbf{B}_4\end{pmatrix}.$$

Jednotlivé sobě odpovídající bloky matice $\mathbf{A}$ a $\mathbf{B}$ mají stejný typ, tj. Typ $\mathbf{A}_1$ má stejný typ jako $\mathbf{B}_1$.Pak

$$\mathbf{A}+\mathbf{B}=\begin{pmatrix}\mathbf{A}_1 & \mathbf{A}_2\\ \mathbf{A}_3 & \mathbf{A}_4\end{pmatrix}+\begin{pmatrix}\mathbf{B}_1 & \mathbf{B}_2\\ \mathbf{B}_3 & \mathbf{B}_4\end{pmatrix}=\begin{pmatrix}\mathbf{A}_1+\mathbf{B}_1 & \mathbf{A}_2+\mathbf{B}_2\\ \mathbf{A}_3+\mathbf{B}_3 & \mathbf{A}_4+\mathbf{B}_4\end{pmatrix} \tag{1.8}$$

Násobení blokových matic

$$\mathbf{A}=\begin{pmatrix}\mathbf{A}_{11} & \dots & \mathbf{A}_{1p}\\ \vdots & \ddots & \vdots\\ \mathbf{A}_{m1} & \dots & \mathbf{A}_{np}\end{pmatrix},\qquad \begin{pmatrix}\mathbf{B}_{11} & \dots & \mathbf{A}_{1n}\\ \vdots & \ddots & \vdots\\ \mathbf{B}_{p1} & \dots & \mathbf{A}_{pn}\end{pmatrix}$$

Jsou dvě blokové matice rozdělené na bloky tak, že počet sloupců bloků $\mathbf{A}_{jk}$ je stejný jako počet řádků bloků $\mathbf{B}_{kj}$, pak se libovolný blok $\mathbf{C}_{ij}$ součinu $\mathbf{AB}$ vyčíslí podle pravidla

$$\mathbf{C}_{ij}=\mathbf{A}_{i1}\mathbf{B}_{1j}+\ldots+\mathbf{A}_{ip}\mathbf{B}_{pj} \tag{1.9}$$

Speciální případ je násobení blokové matice s blokovým vektorem

$$\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{pmatrix} \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{By} + \mathbf{Cz} \\ \mathbf{Dy} + \mathbf{Ez} \end{pmatrix} \tag{1.10}$$

Transponování blokových matic

$$\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{pmatrix}^T = \begin{pmatrix} \mathbf{B}^T & \mathbf{D}^T \\ \mathbf{C}^T & \mathbf{E}^T \end{pmatrix} \tag{1.11}$$

Pozorování Při efektivních implementacích je také velice důležité brát v úvahu aspekt složitosti jednotlivých operací.

Pro jednoduchost uvažujme čtvercové matice $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n}$ a vektor $\mathbf{v} \in \mathbb{R}^n$. Pak násobení matice $\mathbf{A}, \mathbf{B}$ má složitost n^3. Násobení matice $\mathbf{A}$ a vektoru $\mathbf{v}$ má složitost pouze n^2. V případě, žě máme spočítat například takovýto výraz

$$\mathbf{ABv}$$

máme dvě možnosti

- spočítat $(\mathbf{AB})\mathbf{v}$
- nebo $\mathbf{A}(\mathbf{Bv})$

Zatímco první varianta bude mít celkovou složitost $n^3 + n^2$. Výsledkem násobení matice a vektoru je opět vektor a proto druhá varianta bude mít složitost pouze $2n^2$. Stejné je to v případě výrazu$(\mathbf{A} + \mathbf{B})\mathbf{C}$. Je vhodné postupovat podle uvedeného uzávorkování a neroznásobovat. Dosáhneme potom složitosti $n^3 + n^2$ místo $2n^3$.

1.2 Vektorové prostory

Každý se už asi setkal s vektory. Například ve fyzice reprezentují hodnoty veličin, pro něž je důležitá nejen „velikost“, ale také směr působení. Jednou z takových veličin je síla, rychlost atd.

Toto je ovšem velice konkrétní pohled na vektor. V matematice existuje mnohem obecnější kostrukce, která umožňuje, za jasných pravidel, mnohem větší „svobodu“.

Definice Vektorový prostor nad tělesem S (prvek $s \in S$ se nazývá skalár) je množina V společně se dvěmi operacemi:

- sčítání vektorů $V \otimes V \to V$ značíme $\mathbf{v} + \mathbf{w}, \qquad \mathbf{v}, \mathbf{w} \in V$
- násobení skalárem $S \otimes V \to V$ značíme $s \cdot \mathbf{w}, \qquad s \in S, \mathbf{w} \in V$

pro něž platí ($a, b \in S,\ \mathbf{u}, \mathbf{v}, \mathbf{w} \in V$):

1. Existuje neutrální prvek $\mathbf{0} \in V$ tak, že $\forall \mathbf{v} \in V, \mathbf{v} + \mathbf{0} = \mathbf{v}$. Prvek $\mathbf{0}$ se názývá nulový prvek
2. $\forall \mathbf{v} \in V$ existuje opačný prvek $\mathbf{w} \in V$tak, že $\mathbf{v} + \mathbf{w} = \mathbf{0}$
3. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
4. $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$
5. $a(b\mathbf{v}) = (ab)\mathbf{v}$
6. $1\mathbf{v} = \mathbf{v}$, 1 je jednotkový prvek tělesa
7. $a(\mathbf{v} + \mathbf{w}) = a\mathbf{v} + a\mathbf{w}$
8. $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$.

1.3 Lineární nezávislost a báze

Mějme vektorový prostor V nad tělesem S. Nechť $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in V$ a $s_1, s_2, \ldots, s_n \in S$. Pak výraz

$$s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \ldots + s_n\mathbf{v}_n$$

nazveme lineární kombinací vektorů $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Lineární nezávislost Vektory $\mathbf{v}_1, \ldots, \mathbf{v}_n$ jsou lineárně nezávislé, je-li jejich lineární kombiace rovna nulovému vektoru pouze v případě, že všechny skaláry $s_1, \ldots, s_n$ jsou nulové, tj. rovnici

$$s_1\mathbf{v}_1 + s_2\mathbf{v}_2 + \ldots + s_n\mathbf{v}_n = \mathbf{0} \tag{1.12}$$

splňuje pouze triviální kombinace.

Lineární obal Nechť je dána množina $\mathbf{M}$. Pak lineárním obalem $\mathbf{M}$ rozumíme množinu všech lineárních kombinací vektorů množiny$\mathbf{M}$.

Jedním z velice důležitých pojmů je báze vektorového prostoru.

Báze vektorového prostoru V je konečná množina lineárně nezávislých vektorů, jejíž lineární obal je roven celému prostoru V.

1.4 Inverze a regularita

Inverzní matice je další nástroj lineární algebry. Při numerických výpočtech se používá, díky její velké výpočetní náročnosti, jen zřídka. Ovšem při teoretických úvahách a důkazech je často nenahraditelná.

Hodnost matice je číslo, které určuje počet lineárně nezávislých řádků matice.

Regulární matice Čtvercová matice $\mathbf{A} \in \mathbb{R}^{n\times n}$ je regulární jestliže její hodnost je rovna n,tj. má všechny řádky lineárně nezávislé.

Inverzní matice Nechť je dána regulární matice $\mathbf{A}$, pak matici $\mathbf{A}^{-1}$ nazveme inverzní maticí k matici $\mathbf{A}$. pokud platí:

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}. \tag{1.13}$$

1.5 Kroneckerův součin

Definice Nechť jsou dány matice $\mathbf{A}_x \in \mathbb{R}^{m_x,n_x}$, $\mathbf{A}_y \in \mathbb{R}^{m_y,n_y}$. Kroneckerův součin je definován

$$\mathbf{A}_x \otimes \mathbf{A}_y = \begin{pmatrix} a^x_{11}\mathbf{A}_y & \cdots & a^x_{1n}\mathbf{A}_y \\ \vdots & \ddots & \vdots \\ a^x_{m1}\mathbf{A}_y & \cdots & a^x_{mn}\mathbf{A}_y \end{pmatrix} \tag{1.14}$$

Pro Kroneckerův součin platí:

$$\begin{aligned} \mathbf{A} \otimes (\mathbf{B} + \mathbf{C}) &= \mathbf{AB} + \mathbf{AC}, \\ (\mathbf{A} + \mathbf{B}) \otimes \mathbf{C} &= \mathbf{A} \otimes \mathbf{C} + \mathbf{B} \otimes \mathbf{C}, \\ (k\mathbf{A}) \otimes \mathbf{B} &= \mathbf{A} \otimes (k\mathbf{B}) = k(\mathbf{A} \otimes \mathbf{B}), \\ \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} &= \mathbf{A} \otimes (\mathbf{B} \otimes \mathbf{C}), \end{aligned} \tag{1.15}$$

kde $\mathbf{A}, \mathbf{B}, \mathbf{C}$ jsou matice a k je skalár.

Kroneckerův součin obecně není komutativní, tj. $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{BA}$. Avšak $\mathbf{A} \otimes \mathbf{B}$ a $\mathbf{B} \otimes \mathbf{A}$ jsou permutačně ekvivalentní. Existují permutační matice $\mathbf{P}$, $\mathbf{Q}$ tak, že platí:

$$\mathbf{A} \otimes \mathbf{B} = \mathbf{P}(\mathbf{B} \otimes \mathbf{A})\mathbf{Q}$$

Pro nás ovšem budou nejdůležitější následující vlastnosti:

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AB} \otimes \mathbf{CD} \tag{1.16}$$

Poznámka 1.5.1. Důkaz této relace přímo vychází z definice - stačí si pouze napsat matice po složkách a promyslet si, jak budou jednotlivé součiny vypadat.

Inverze Kroneckerova součinu je možná pouze v případě, pokud k jednotlivým maticím existují inverze. Pak

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1} \tag{1.17}$$

Velmi důležitou vlastností Kroneckerova součinu je násobení tohoto součinu s vektorem, který lze rozdělit na součin jednotlivých matic s vektorem. Než si ukážeme, jak toto násobení probíhá, musíme ještě nadefinovat jednu funkci.

Definice Nechť $\mathbf{V}$ je matice typu $n_x \times n_y$, pak vec($\mathbf{V}$) je vektor o $n_x n_y$ složkách. Jinými slovy lze říci, že funkce vec „přeskládá“ matici na vektor po řádcích. Potom je také možno definovat inverzní funkci vec^{-1}, která z vektoru vytvoří matici. V Matlabu je nadefinována funkce reshape($\mathbf{A}$,n_x,n_y), která matici $\mathbf{A}$ typu $m_x \times m_y$ přeskládá na matici typu $n_x \times n_y$. Musí však platit $n_x n_y = m_x m_y$.

Pro násobení Kroneckerova součinu s vektorem platí:

$$(\mathbf{A}_x \otimes \mathbf{A}_y)\mathbf{v} = vec(\mathbf{A}_x \mathbf{V} \mathbf{A}_y), \qquad \mathbf{V} = vec^{-1}(\mathbf{v}) \tag{1.18}$$

1.6 Řešení soustav lineárních rovnic

1.6.1 Gaussova eliminace

Gaussova eliminace je jedna z nejrozšířenějších metod pro řešení soustav lineárních rovnic. Mějme systém m rovnic o n neznámých. Pak matice soustavy má rozměr $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x},\ \mathbf{b} \in \mathbb{R}^m$

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

Napišme si rozšířenou matici soustavy:

$$\left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ a_{21} & \cdots & a_{2n} & b_2 \\ \vdots & \cdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_n \end{array}\right) \tag{1.19}$$

Řešení soustavy se nezmění, pokud budeme provádět pouze takzvané ekvivalentní úpravy:

1. výměna dvou řádků,
2. vynásobení řádku nenulovým číslem,
3. přičtení násobku řádku k jinému řádku.

Úpravy provádíme vždy na celé rozšířené matici 1.19, tj. i na vektoru $\mathbf{b}$

Díky těmto úpravám upravíme matici do takového tvaru, že pro $1 \leq i \leq m$ platí

- řádek i je buď nulový
- anebo má první nenulovou souřadnici na pozici p_i a všechny řádky pod ním mají na prvních p_i pozicích nuly.

Příklad 1.6.1.

$$\left(\begin{array}{ccc|c} 3 & -1 & 4 & 0 \\ 0 & 5 & -4 & 1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \end{array}\right)$$

Poznámka 1.6.1. Gaussova eliminace se dá také použít například k určení báze lineárního obalu daných vektorů.

Nástin, jak toho lze dosáhnout:

Mějme dány vektory $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m \in \mathbb{R}^n$. Sestavíme z nich matici tak, že jednotlivé vektory budou tvořit řádky matice.

$$\begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_m \end{pmatrix} \tag{1.20}$$

Nenulové řádky v matici 1.20 po provedení algoritmu Gaussovy eliminace tvoří bázi lineárního obalu daných vektorů.

1.6.2 LU rozklad

Pomocí LU rozkladu můžeme rozložit regulární matici $\mathbf{A}$ na tvar:

$$\mathbf{A} = \mathbf{LUP}, \tag{1.21}$$

kde $\mathbf{L}$ je dolní trojúhelníková matice, $\mathbf{U}$ je horní trojúhelníková matice a $\mathbf{P}$ je permutační matice.

Místo řešení soustavy $\mathbf{Ax} = \mathbf{b}$ potom řešíme soustavu

$$\mathbf{L}(\mathbf{U}(\mathbf{Px})) = \mathbf{b}, \tag{1.22}$$

což řešíme postupně

$$\begin{aligned} \mathbf{Lz} &= \mathbf{b} \\ \mathbf{Uy} &= \mathbf{z} \\ \mathbf{Px} &= \mathbf{y} \end{aligned} \tag{1.23}$$

LU rozklad můžeme získat pomocí Gaussovy eliminace.

Algoritmus 1. ***LU rozklad ikj-varianta [5]***

```
function [ L,U ] = myLU( A )
n = size(A,2);
for i = 2:n
    for k = 1:i-1
        A(i,k) = A(i,k)/A(k,k);
        for j = k+1:n
            A(i,j) = A(i,j) - A(i,k)*A(k,j);
        end
    end
end
U = triu(A);
L = tril(A,-1) + speye(n);
```

Tento algoritmus 1 je výhodný, protože v jednom kroku vypočte jak prvky matice $\mathbf{L}$, tak prvky matice $\mathbf{U}$. Protože všechny úpravy se provádějí přímo nad vstupní maticí $\mathbf{A}$, poslední dva řádky nám z matice $\mathbf{A}$ získají očekávaný výstup (matice $\mathbf{L}$, $\mathbf{U}$).

1.6.3 Metoda sdružených gradientů

Obě předešlé metody byly takzvané přímé. To znamená, že po provedení celého algoritmu máme přesné[1] řešení dané soustavy. Metoda sdružených gradientů byla také považována za

[1] Přesného řešení je dosaženo pouze v přesné aritmetice, kde se zanedbávají zaokrouhlovací chyby

další přímou metodu. Ovšem o něco později bylo objeveno její kouzlo jako iterační [2]metody. Opět budeme chtít řešit sustavu lineárních rovnic

$$\mathbf{Ax} = \mathbf{b} \qquad \mathbf{A} \in \mathbb{R}^{n\times n},\ \mathbf{b} \in \mathbb{R}^n, \tag{1.24}$$

kde navíc matice $\mathbf{A}$ je symetrická pozitivně definitní.

Poznámka 1.6.2.

- pro symetrické matice platí $\mathbf{A} = \mathbf{A}^T$
- pro pozitivně definitní matice platí $\mathbf{x}^T\mathbf{Ax} > 0, \forall \mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq \mathbf{0},\ \mathbf{A} \in \mathbb{R}^{n\times n}$

  Definujme skalární součin:

$$(\mathbf{u}, \mathbf{v})_{\mathbf{A}} = (\mathbf{Au}, \mathbf{v}), \qquad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^n \tag{1.25}$$

a systém vektorů

$$\mathbf{P} = \{\mathbf{p}^k\}_{k=0}^{n-1}, \qquad k = 0, \ldots, n-1, \tag{1.26}$$

pro něž platí

$$(\mathbf{Ap}^l, \mathbf{k}^k) = 0 \qquad k \neq l, \tag{1.27}$$

$$(\mathbf{Ap}^k, \mathbf{p}^k) > 0 \tag{1.28}$$

Tento systém budeme nazývat A-ortogonální. Přibližné řešení soustavy 1.24 potom můžeme psát ve tvaru

$$\mathbf{x}^{k+1} = \mathbf{x}^0 - \sum_{l=0}^{k} \alpha_l \mathbf{p}^l = \mathbf{x}^k - \alpha_k \mathbf{p}^k, \qquad k = 0, \ldots, n-1, \tag{1.29}$$

kde

$$\alpha_k = \frac{(\mathbf{r}^k, \mathbf{p}^k)}{\mathbf{Ap}^k, \mathbf{p}^k)},\ \mathbf{r}^k = \mathbf{Ax}^k - \mathbf{b}, \qquad k = 0, \ldots, n-1 \tag{1.30}$$

[2]Iterační metody krůček po krůčku zpřesňují řešení. My tak máme možnost stanovit si požadovanou přesnost a v případě jejího dosažení algoritmus ukončit. Nebo také stanovit maximální počet iterací a zajistit tak menší časovou náročnost

Důležitý poznatek je, že [6]

$$(\mathbf{r}^{k+1}, \mathbf{p}^l) = 0, \qquad l = 0, \ldots, k, \;\; k = 0, \ldots, n-2 \tag{1.31}$$

Nyní můžeme přistoupit ke konstrukci A-ortogonálního systému $\mathbf{P}$ pomocí Gramm-Schmidtova ortogonalizačního procesu. Mějme dám nějaký systém $\mathbf{W}$ lineárně nezávislých vektorů $\mathbf{w}^k$. Položme $\mathbf{p}^0 = \mathbf{w}^0$. Potom můžeme zkonstruovat A-ortogonální vektory $\mathbf{p}^k, \;\; k = 0, \ldots, n-2$ následujícím způsobem:

$$\mathbf{p}^{k+1} = \mathbf{w}^{k+1} - \sum_{l=0}^{k} \beta_{kl} \mathbf{p}^l, \qquad \beta_{kl} = \frac{(\mathbf{A}\mathbf{w}^{k+1}, \mathbf{p}^l)}{(\mathbf{A}\mathbf{p}^l, \mathbf{p}^l)}, \qquad l = 0, \ldots, k. \tag{1.32}$$

Z 1.31 a 1.32 plyne

$$(\mathbf{r}^{k+1}, \mathbf{w}^l) = (\mathbf{r}^{k+1}, \mathbf{p}^l) + \sum_{j=0}^{l-1} \beta_{lj} (\mathbf{r}^{k+1}, \mathbf{p}^j) = 0 \tag{1.33}$$

Pokud je vektor $\mathbf{r}^{k+1}$ nulový, máme řešení soustavy. Obecně se toto ovšem nestane. Protože systém vektorů $(\mathbf{w}^0, \ldots, \mathbf{w}^k, \mathbf{r}^{k+1})$ je lineárně nezávislý, můžeme volit $\mathbf{w}^{k+1} = \mathbf{r}^{k+1}$. Z 1.33 plyne

$$(\mathbf{r}^{k+1}, \mathbf{r}^l) = 0 \qquad l = 0, \ldots, k.$$

Z předpisu pro koeficienty α_k definované v 1.30 dostáváme

$$(\mathbf{r}^k, \mathbf{p}^k) = (r^k, r^k) - \sum_{l=0}^{k-1} \beta_{k-1,l} (\mathbf{r}^k, \mathbf{p}^l) = (\mathbf{r}^k, \mathbf{r}^k)$$

použitím rekurentního předpisu 1.32 a ortogonálního vztahu 1.31. Z $(\mathbf{r}^l, \mathbf{r}^l) > 0$ plyne, že $\alpha_l > 0, \;\; l = 0, \ldots, k$, a proto můžeme psát

$$\mathbf{A}\mathbf{p}^l = \frac{1}{\alpha_l}(\mathbf{r}^l - \mathbf{r}^{l+1}).$$

Dále můžeme psát

$$(\mathbf{A}\mathbf{w}^{k+1}, \mathbf{p}^l) = (\mathbf{r}^{k+1}, \mathbf{A}\mathbf{p}^l) = \frac{1}{\alpha_l}(\mathbf{r}^{k+1}, \mathbf{r}^l - \mathbf{r}^{l+1}) = 0 \qquad l = 0, \ldots, k-1$$

a

$$(\mathbf{A}\mathbf{w}^{k+1}, \mathbf{p}^k) = -\frac{1}{\alpha_k}(\mathbf{r}^{k+1}, \mathbf{r}^{k+1})$$

Když dáme tyto poznatky dohromady, dostáváme $\beta_{kl} = 0,\ l = 0, \ldots, k-1$ a

$$\beta_{kk} = \beta_k = -\frac{(\mathbf{r}^{k+1}, \mathbf{r}^{k+1})}{(\mathbf{r}^k, \mathbf{r}^k)},$$

takže

$$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta_k \mathbf{p}^k.$$

Algoritmus 2. ***Algoritmus metody sdružených gradientů***

```
function [ x,iter ] = myCG( A,b, x, eps, maxIt )
iter = 0;
r = b - A*x;
p = r;
while norm(r)/norm(x) > eps
    if(iter >= maxIt)
        break;
    end
    iter=iter+1;
    Ap = A*p;
    ptAp=p'*Ap;
    alfa = (r'*p)/ptAp;
    x = x + alfa*p;
    r = r - alfa*Ap;
    beta = (r'*Ap)/ptAp;
    p = r + beta*p;
    iter=iter+1;
end
```

Předpodmínění metody sdružených gradientů

Pokud je matice soustavy 1.24 špatně podmíněná [3], sdružené gradienty konvergují pomalu. Zlepšit odmíněnost se dá například takto:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{1.34}$$

[3] číslo podmíněnosti pro SPD matici $\mathbf{A}$ se vypočte jako poměr největšího vlastního čísla ku nejmenšímu, značí se $cond(\mathbf{A})$

$$cond(\mathbf{A}) = \frac{\lambda_{max}}{\lambda_{min}}$$

nebo

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{u} = \mathbf{b},\ \mathbf{x} = \mathbf{M}^{-1}\mathbf{u}., \tag{1.35}$$

kde matice $\mathbf{M}$ je SPD a platí

- $cond(\mathbf{M}^{-1}\mathbf{A}) < cond(\mathbf{A})$ respektive $cond(\mathbf{A}\mathbf{M}^{-1}) < cond(\mathbf{A})$
- řešení soustavy $\mathbf{M}\mathbf{y} = \mathbf{b}$ je levné.

Obecně nemusí být matice $\mathbf{M}^{-1}\mathbf{A}$ symetrická. Nesymetrie by narušila předpoklady sdružených gradientů. Zavedeme si proto nový skalární součin

$$(\mathbf{x}, \mathbf{y})_M = (\mathbf{M}\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{M}\mathbf{y}). \tag{1.36}$$

Klasický skalární součin v metodě sdružených gradientů nahradíme tímto „M-skalárním součinem“. Algoritmus metody sdružených gradientů s předpodmíněním je uveden dále 3. Pro získání matice $\mathbf{M}$ byla použita neúplná LU-faktorizace viz [5].

Srovnání metody sdružených gradientů s a bez předpodmínění neúplnou LU-faktorizací

	Počet iterací	
n	bez předpodmínění	Předpodmínění neúplnou LU-faktorizací
4	1	1
9	3	1
25	5	3
81	15	18
289	46	33
1089	92	61
4225	178	114

Algoritmus 3. ***Algoritmus metody sdružených gradientů s předpodmíněním***

```
function [ x,iter] = myCGPrecond( A,b,eps,maxIt )
n = size(A,2);
x = ones(n,1);
r = b - A*x;
iter=0;
tic;
[L,U] = luinc(A,'0');
z = U\(L\r);
p = z;
while norm(r)/norm(b) > eps
    if(iter >= maxIt)
        break;
    end
    iter=iter+1;
    Ap = A*p;
    ptAp=p'*Ap;
    alfa = (r'*z)/ptAp;
    x = x + alfa*p;
    r_last = r;
    r = r - alfa*Ap;
    z_last = z;
    z = U\(L\r);
    beta = (r'*z)/(r_last'*z_last);
    p = z + beta*p;
end
```

Kapitola 2

Fourierovy Transformace

2.1 Fourierovy řady

Často je vhodné funkci $f(t)$ rozvinout v nekonečnou řadu. Jednou z možností je tzv. Fourierův rozvoj.

2.1.1 Fourierova řada v komplexním tvaru

$$f(t) = \sum_{-\infty}^{\infty} C_n e^{i\omega nt},\ t \in \langle 0, T\rangle\ ,\ \omega = \frac{\pi}{L},\ T = 2L \tag{2.1}$$

Koeficienty C_n vypočteme následovně

$$\begin{aligned} C_n &= \tfrac{\omega}{2\pi} \int_a^{a+T} f(t) e^{-i\pi n\omega t} dt,\ n \in \mathbb{Z},\ a \in \mathbb{R} \\ C_n &= \tfrac{1}{2L} \int_a^{a+T} f(t) e^{-i\pi n\omega t} dt,\ n \in \mathbb{Z},\ a \in \mathbb{R} \end{aligned} \tag{2.2}$$

Kde a je počáteční bod periody, $a + 2L$ resp. $a + T$ je koncový bod periody.

Je-li funkce $f(t)$ periodická o periodě $T = 2\pi$, lze potom pro její Fourierovu řadu a koeficienty psát

$$f(t) = \sum_{-\infty}^{\infty} C_n e^{int},\ C_n = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-int} dt \tag{2.3}$$

2.1.2 Dirichletovy podmínky

Funkci lze rozvinout ve Fourierovu řadu, jestliže splňuje tzv. Dirichletovy podmínky

1. $f(t)$ je periodická nebo periodicky rozšířitelná funkce,
2. na zadaném intervalu je funkce $f(t)$ alespoň po částech spojitá, to je má konečný počet bodů nespojitosti I. druhu,
3. na zadaném intervalu má funkce $f(t)$ konečný počet extrémů (konstantní části se nepočítají),
4. funkce je definována v koncových bodech intervalu (to je nabývá v nich konečných hodnot) nebo existují příslušné jednostranné limity funkce v těchto bodech.

2.1.3 Fourierova řada v reálném tvaru

Vztah mezi Fourierovou řadou funkce $f(t)$ v komplexním a reálném tvaru

$$\begin{aligned} f(t) &= \sum_{-\infty}^{\infty} C_n e^{int} = \\ &= \sum_{-\infty}^{1} C_n e^{int} + C_0 + \sum_{1}^{\infty} C_n e^{int} = \\ &= C_0 + \sum_{1}^{\infty} (C_{-n} e^{-int} + C_n e^{int}) \end{aligned} \tag{2.4}$$

$$f(t) = C_0 + \sum_{1}^{\infty} (C_{-n}(\cos(\omega nt) + \imath \sin(\omega nt)) + C_n(\cos(\omega nt) + \imath \sin(\omega nt))) \tag{2.5}$$

$$f(t) = C_0 + \sum_{1}^{\infty} ((C_{-n} + C_n)\cos(\omega nt) + (C_n - C_{-n})\sin(\omega nt)) \tag{2.6}$$

Zaveďme substituci

$$a_0 = 2C_0,\ a_n = C_n + C_{-n},\ b_n = \imath(C_n - C_{-n})$$

Nyní již máme Fourierovu řadu v reálném tvaru

$$f(t) = \frac{a_0}{2} + \sum_{-\infty}^{\infty} a_n \cos(\omega nt) + b_n \sin(\omega nt),\ t \in \mathbb{R} \tag{2.7}$$

$$a_0 = 2C_0 = \frac{2}{T} \int_a^{a+T} f(t)dt \tag{2.8}$$

$$a_n = C_n + C_{-n} = \frac{2}{T} \int_a^{a+T} f(t)\cos(\omega nt)dt \tag{2.9}$$

$$b_n = \imath(C_n - C_{-n}) = \frac{2}{T} \int_a^{a+T} f(t)\sin(\omega nt)dt \tag{2.10}$$

Fourierova řada sudé funkce

Pokud je funkce $f(t)$ definovaná na intervalu $\langle -L, L\rangle$ sudá, to znamená $f(t) = f(-t)$, pak vzorce pro Fourierovy koeficienty jejího rozvoje budou

$$a_n = \frac{2}{L}\int_0^L f(t)\cos(\omega nt)dt,\ T = 2L,\ \omega = \frac{\pi}{L},\ n = 0, 1, \ldots \tag{2.11}$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(t)\sin(\omega nt)dt = 0,\ n = 0, 1, \ldots \tag{2.12}$$

Fourierova řada liché funkce

Podobně můžeme postupovat i v případě, že funkce $f(t)$ definovaná na intervalu $\langle -L, L\rangle$ je lichá, to znamená $f(t) = -f(-t)$. Pro koeficienty jejího Fourierova rozvoje bude platit

$$a_0 = \frac{1}{L}\int_{-L}^{L} f(t)dt = 0 \tag{2.13}$$

$$a_n = \frac{1}{L}\int_{-L}^{L} f(t)\cos(\omega nt)dt = 0,\ T = 2L,\ \omega = \frac{\pi}{L},\ n = 0, 1, \ldots \tag{2.14}$$

$$b_n = \frac{2}{L}\int_0^L f(t)\sin(\omega nt)dt,\ n = 0, 1, \ldots \tag{2.15}$$

Příklad 2.1.1. *Mějme zadánu následující funkci*

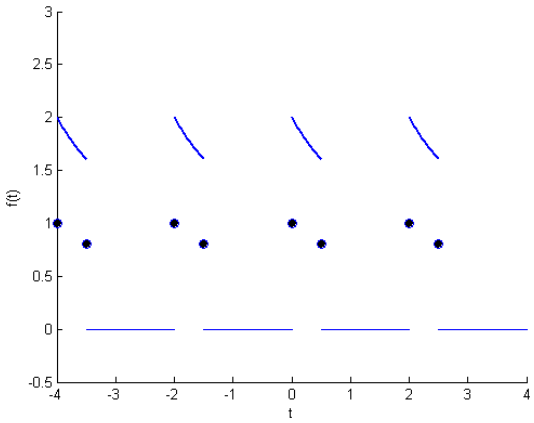
$$f(t) = \left\{ \begin{array}{cc} 1 + e^{-t}, & t \in (0, \frac{1}{2}) \\ 0, & t \in (\frac{1}{2}, 2) \end{array} \right\} \tag{2.16}$$

Fourierův rozvoj funkce 2.16 (periodické prodloužení)

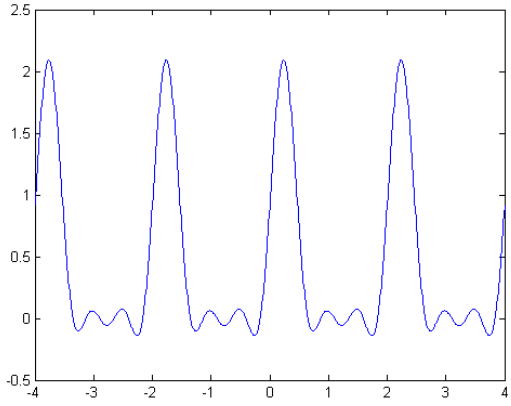
$$\begin{aligned}
C_n &= \frac{1}{T}\int_a^{a+T} f(t)e^{-in\omega t}dt \\
\omega &= \frac{2\pi}{T} = \frac{2\pi}{2} = \pi \\
C_n &= \frac{1}{2}\int_0^{1/2}(1+e^{-t})e^{-in\omega t}dt + \frac{1}{2}\int_0^{1/2} 0e^{-in\omega t}dt \\
&= \frac{1}{2}\int_0^{1/2}(1+e^{-t})e^{-in\omega t}dt \\
&= \frac{1-e^{-\frac{in\pi}{2}}}{i2n\pi} + \frac{1-e^{-\frac{1+in\pi}{2}}}{2+i2n\pi} \\
C_0 &= \frac{1}{2}\int_0^{1/2}(1+e^{-t}dt = \frac{1}{2}\left(\frac{3}{2} - e^{-\frac{1}{2}}\right)
\end{aligned}$$

Fourierova řada má tvar:

$$f(t) = \frac{1}{2}\left(\frac{3}{2} - e^{-\frac{1}{2}}\right) + \sum_{-\infty}^{\infty} \frac{1-e^{-\frac{in\pi}{2}}}{i2n\pi} + \frac{1-e^{-\frac{1+in\pi}{2}}}{2+i2n\pi}$$



Obrázek 2.1: Součet Fourierovy řady



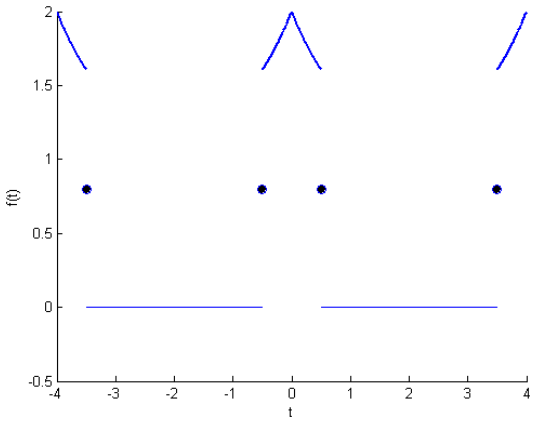
Obrázek 2.2: Součet prvních tří členů Fourierovy řady

Kosinový rozvoj funkce 2.16 (sudé prodloužení)

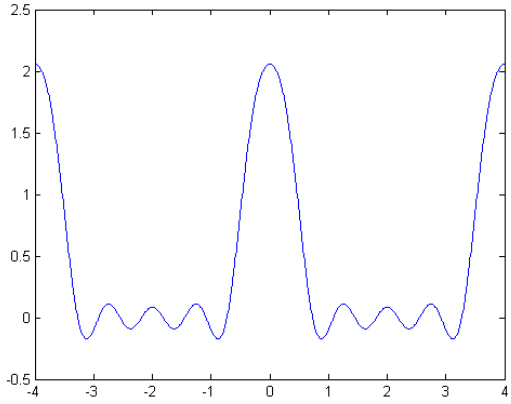
$$\begin{aligned} a_n &= \frac{2}{L}\int_0^L f(t)\cos\frac{n\pi t}{L}dt \\ a_n &= \int_0^{1/2}(1+e^{-t})\cos\frac{n\pi t}{2}dt \\ &= -\frac{2\sin\left(\frac{n\pi}{4}\right)}{n\pi} + 2\frac{-2e^{-\frac{1}{2}}\cos\left(\frac{n\pi}{4}\right)+e^{-\frac{1}{2}}n\pi\sin\left(\frac{n\pi}{4}\right)+2}{4+n^2\pi^2} \\ a_0 &= \frac{2}{L}\int_0^L f(t)dt = \int_0^{1/2}(1+e^{-in\pi t})dt = \frac{3}{2}-e^{-\frac{1}{2}} \end{aligned}$$

Kosinová řada má tvar:

$$f(t) = \frac{1}{2}\left(\frac{3}{2}-e^{-\frac{1}{2}}\right) + \sum_{-\infty}^{\infty} -\frac{2\sin\left(\frac{n\pi}{4}\right)}{n\pi} + 2\frac{-2e^{-\frac{1}{2}}\cos\left(\frac{n\pi}{4}\right)+e^{-\frac{1}{2}}n\pi\sin\left(\frac{n\pi}{4}\right)+2}{4+n^2\pi^2}$$



Obrázek 2.3: Součet Kosinovy řady



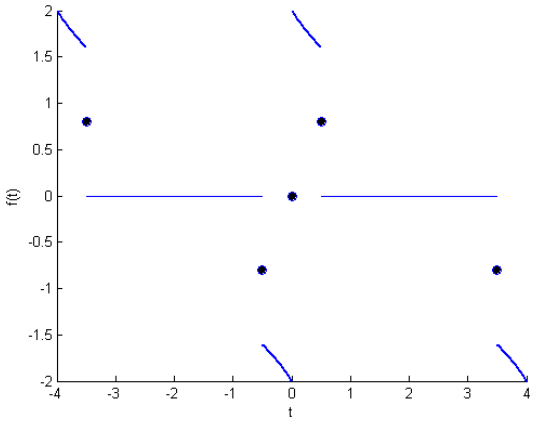
Obrázek 2.4: Součet prvních tří členů Kosinovy řady

Sinový rozvoj funkce 2.16 (liché prodloužení)

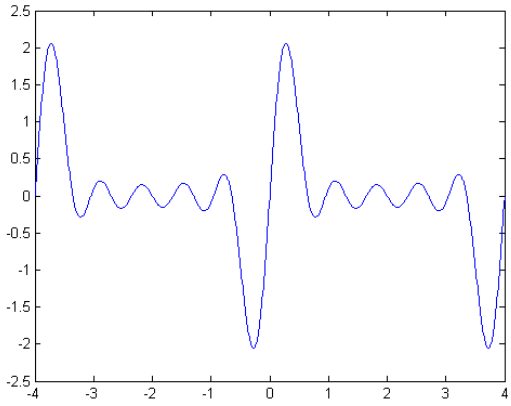
$$\begin{aligned} a_n &= \frac{2}{L}\int_0^L f(t)\sin\frac{n\pi t}{L}dt \\ a_n &= \int_0^{1/2}(1+e^{-t})\sin\frac{n\pi t}{2}dt \\ &= -\frac{2\cos\left(\frac{n\pi}{4}\right)-1}{n\pi} - 2\frac{e^{-\frac{1}{2}}n\pi\cos\left(\frac{n\pi}{4}\right)+2e^{-\frac{1}{2}}\sin\left(\frac{n\pi}{4}\right)-n\pi}{4+n^2\pi^2} \end{aligned}$$

Sinová řada má tvar:

$$f(t) = \sum_{-\infty}^{\infty} -\frac{2\cos\left(\frac{n\pi}{4}\right) - 1}{n\pi} - 2\frac{e^{-\frac{1}{2}}n\pi\cos\left(\frac{n\pi}{4}\right) + 2e^{-\frac{1}{2}}\sin\left(\frac{n\pi}{4}\right) + n\pi}{4 + n^2\pi^2}$$



Obrázek 2.5: Součet Sinovy řady



Obrázek 2.6: Součet prvních tří členů Sinovy řady

2.2 Diskrétní fourierova transformace

Diskrétní Fourierovu transformaci můžeme zapsat ve tvaru [1]

$$\hat{\mathbf{a}} = \mathbf{W}\mathbf{a},$$
$$\mathbf{W} = (w_{kl}) \in \mathbb{C}^{n\times n},\ w_{kl} = e^{\frac{-i(k-1)(l-1)}{2}\pi n},\ k,l = 1,2,\ldots,n,\ a \in \mathbb{R}^n$$

Obrazem DFT je vektor s obecně komplexními komponentami. Označme $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ sloupcový rozklad matice DFT.

Věta 2.2.1. *Nechť* $\mathbf{T} \in \mathbb{R}^{n\times n}$ *je matice posunutí,* $\mathbf{W} \in \mathbb{C}^{n\times n}$ *je matice DFT a* $\mathbf{a} \in \mathbb{R}^n$. *Pak platí:*

$$\mathbf{W}(\mathbf{T}^l\mathbf{a}) = \mathbf{W}_l(\mathbf{W}\mathbf{a}), l = 0,1,\ldots,n-1, \tag{2.17}$$

kde $\mathbf{W}_l \in \mathbb{C}^{n\times n}$ *je diagonální matice s diagonálou tvořenou vektorem* $\mathbf{w}_{l+1}$. *Toto tvrzení nám vyjadřuje skutečnost, že Fourierovou transformací operátoru posunutí je operátor modulace.*

Věta 2.2.2. *Nechť* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *je cirkulantní matice s prvním sloupcem* $\mathbf{a} \in \mathbb{R}^n$*. Pak platí:*

$$\mathbf{A} = \mathbf{W}^{-1}\mathbf{D}\mathbf{W}, \tag{2.18}$$

kde $\mathbf{W} \in \mathbb{C}^{n\times n}$ *je matice diskrétní Fourierovy transformace a* $\mathbf{D} \in \mathbb{C}^{n\times n}$ *je diagonální matice s diagonálou tvořenou vektorem* $\hat{\mathbf{a}} = \mathbf{W}\mathbf{a}$*. Jinými slovy se dá říci, že každou cirkulantní matici lze rozložit pomocí tohoto spektrálního rozkladu na diagonální matici* $\mathbf{D}$*, tvořenou vlastními čísly matice* $\mathbf{A}$*, které lze získat pomocí DFT prvního sloupce matice* $\mathbf{A}$*. Dále pak na obecně komplexní matici* $\mathbf{W}$*, která je pro všechny cirkulantní matice stejná.*

Důkaz. Předpis pro cirkulantní matici vynásobíme maticí $\mathbf{W}$. Tím dostaneme:

$$\mathbf{WA} = (\mathbf{Wa}, \mathbf{W}(\mathbf{Ta}), \mathbf{W}(\mathbf{T}^2\mathbf{a}), \ldots, \mathbf{W}(\mathbf{T}^{n-1}\mathbf{a}))$$

Použijeme výše uvedenou větu 2.2.1 a získáme:

$$\mathbf{WA} = (\mathbf{W}_0\hat{\mathbf{a}}, \mathbf{W}_1\hat{\mathbf{a}}, \mathbf{W}_2\hat{\mathbf{a}}, \ldots, \mathbf{W}_{n-1}\hat{\mathbf{a}})$$

Po další úpravě dostáváme přímo tvrzení věty:

$$\mathbf{WA} = \mathbf{DW} \Rightarrow \mathbf{A} = \mathbf{W}^{-1}\mathbf{DW}$$

□

2.3 Algoritmus rychlé Fourierovy transformace

Při řešení sedlobodových soustav lineárních rovnic budeme pro výpočet DFT používat algoritmus rychlé Fourierovy transformace, který v Matlabu provádí funkce fft. Poprvé se o fft zmínil Gauss v roce 1805, ale uvedena do praxe byla až dvěma programátory z IBM v roce 1965. Existuje několik postupů, jak efektivně spočítat DFT. Jeden z nich se jmenuje tzv. dělící metoda. Ta vychází z původní DFT, tzn. máme diskrétní dělení intervalu o délce N a příslušné funkční hodnoty x_n (s periodou N), kde $n = 0, \ldots, N-1$. Podle definice spočítáme DFT takto:

$$X_k = \sum_{n=0}^{N-1} x_n \omega_N^{-nk},\ k = 0, \ldots, N-1,\ \omega_N = e^{\frac{i2\pi}{N}} \tag{2.19}$$

Rozdělíme x_n na dvě stejné části y_n a z_n, kde $y_n = x_{2n}$ a $z_n = x_{2n+1}$ a původní definice nám přejde na tvar:

$$X_k = \sum_{n=0}^{N/2-1} (y_n \omega_N^{-2nk} + z_n \omega_N^{-(2n+1)k}),\ k = 0, \ldots, N-1,\ \omega_N = e^{\frac{i2\pi}{N}} \tag{2.20}$$

Teď využijeme symetrickou vlastnost komplexní exponenciály:

$$\omega_N^{-2nk} = \omega_{N/2}^{-nk}, \text{ což odpovídá výrazu} e^{-\imath 4\pi nk/N} = e^{-\imath 2\pi nk/(N/2)}$$

S využitím této vlastnosti můžeme výraz přepsat do tvaru:

$$X_k = \sum_{n=0}^{N/2-1} y_n \omega_{N/2}^{-nk} + \omega_N^{-k} \sum_{n=0}^{N/2-1} z_n \omega_{N/2}^{-nk} \tag{2.21}$$

Nyní máme spočítat namísto DFT délky N dvě DFT o délce N/2. Tyto dvě DFT si označíme jako Yk a Zk a nyní můžeme psát:

$$\begin{aligned} X_k &= Y_k + \omega_N^{-k} Z_k, \\ X_{k+N/2} &= Y_{k+N/2} + \omega_N^{-k+n/2} Z_{k+N/2} \end{aligned} \tag{2.22}$$

Všimněme si, že $\omega_N^{-N/2} = -1$, a protože perioda Yk,Zk je $N/2$, můžeme předchozí výraz přepsat do tvaru (pro $k = 0, \ldots, N/2-1$):

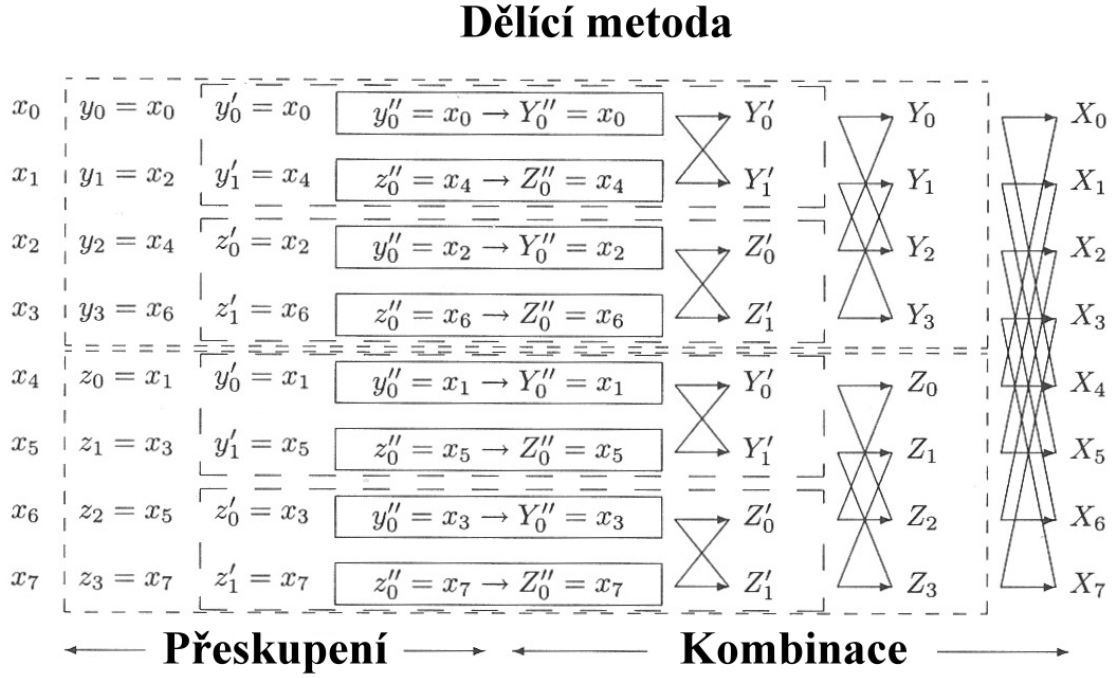
$$\begin{aligned} X_k &= Y_k + \omega_N^{-k} Z_k, \\ X_{k+N/2} &= Y_k - \omega_N^{-k} Z_k \end{aligned} \tag{2.23}$$

Nyní umíme jednu DFT o délce N rozdělit na dvě DFT o délce N/2. Předchozí postup budeme aplikovat tak dlouho, dokud nám z původní DFT nevznikne N DFT o délce jedna (to je možné pouze, když N je mocnina dvou, jinak rozdělujeme až do lichých částí).

Tento postup má dvě hlavní části:

1. **přeskupení**, kde původní posloupnost úplně rozdělíme na stejně velké liché podposloupnosti.

2. **kombinační**, kde postupným spojováním podposloupností (o délce 1 na posloupnost délky 2 ...) získáme výsledek X_k.



Obrázek 2.7: Schéma dělící metody

Schéma postupu dělící metody [3]:

2.4 Diskrétní sinová transformace

DST můžeme zapsat ve tvaru [2]:

$$\hat{\mathbf{a}} = \mathbf{S}\mathbf{a}, \tag{2.24}$$

$$\mathbf{S} = \{s\}_{kl} \in \mathbb{R}^{n\times n},\ s_{kl} = \sin(kl\pi h),\ k,l = 1,2,\ldots,n,\ h = \frac{a}{n+1}$$

2.4.1 Propojení DST a DFT

Platí:

$$\begin{aligned} e^{i\lambda} &= \cos\lambda + \imath\sin(\lambda), \qquad \lambda \in \mathbb{R}, i = \sqrt{-1}, \\ e^{in\lambda} &= \cos n\lambda + \imath\sin(n\lambda), \qquad n \in \mathbb{Z}, \\ e^{-i\lambda} &= \cos\lambda - \imath\sin(\lambda), \\ \sin\lambda &= \tfrac{e^{i\lambda}-e^{-i\lambda}}{2\imath}. \end{aligned} \tag{2.25}$$

Pak

$$\begin{aligned} \omega_N = e^{-i2\pi/N} &= \cos(\tfrac{2\pi}{N}) - \imath\sin(\tfrac{2\pi}{N}), \\ \omega_{2N+2}^{jk} = e^{\imath 2jk\pi}/2N+2 = e^{-\imath jk\pi h} &= \cos(jk\pi h) - \imath\sin(jk\pi h). \end{aligned} \tag{2.26}$$

Věta 2.4.1. *Nechť je dán vektor* $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *a vektor* $\mathbf{y} \in \mathbb{R}^{2n+2}$, *který vznikne z vektoru takto:*

$$\mathbf{y} = (0, x_1, x_2, \ldots, x_n, 0, -x_n, -x_{n-1}, \ldots, -x_1).$$

Pak platí:

$$DST(\mathbf{x})_k = -2\imath DFT(\mathbf{y})_{k+1}, \qquad k = 1, 2, \ldots, n.$$

Důkaz.

$$\begin{aligned} (DFT_{2n+2}\mathbf{y})_{k+1} &= \textstyle\sum_{j=1}^{n} x_j wjk - \sum_{j=n+2}^{2n+2} x_{2n+2-j} wjk = \\ &= \textstyle\sum_{j=1}^{n} x_j w^{jk} - \sum_{j=1}^{n} x_j w^{(2n+2-j)k} = \\ &= \textstyle\sum_{j=1}^{n} x_j w^{jk} - \sum_{j=1}^{n} x_j w^{(2n+2)k} w^{-jk} = \\ &= \textstyle\sum_{j=1}^{n} x_j (w^{jk} - w^{-jk}) = \\ &= -2\imath \textstyle\sum_{j=1}^{n} x_j \sin(jk\pi h). \end{aligned} \tag{2.27}$$

□

Kapitola 3

Numerické řešení

Nyní se budeme zabývat numerickým řešením dirichletovy okrajové úlohy

$$\begin{aligned} -\Delta u &= f \qquad v\, \omega \\ +\text{o.p.} &\qquad na\, \partial\omega. \end{aligned} \tag{3.1}$$

Pro jednoduchost budeme uvažovat homogenní dirichletovy okrajové podmínky, tj. $u = 0$ na $\partial\omega$.

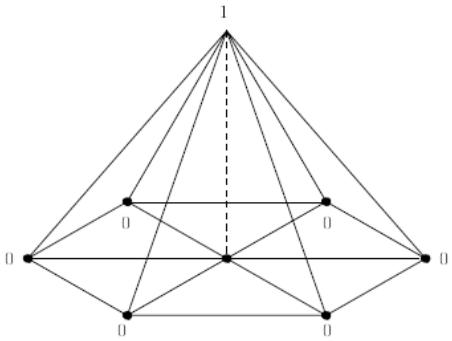
Slabá formulace úlohy 3.1 má tvar. Najdi funkci $u \in \mathbb{H}_0^1$ takovou, že platí

$$\int_\omega \nabla u \nabla v dx = \int_\omega f v dx \qquad \forall v \in \mathbb{H}_0^1(\omega) \tag{3.2}$$

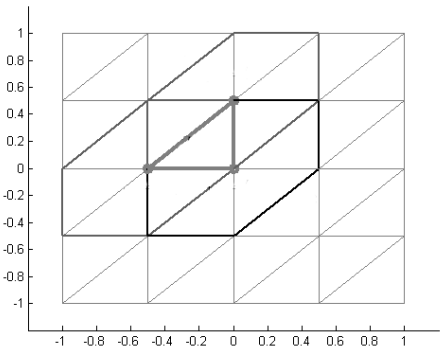
Takto formulovaná úloha jde výhodně numericky řešit metodou konečných prvků.

3.1 Stručně o Metodě konečných prvků (MKP)

Hlavní myšlenka MKP spočívá v nahrazení prostoru s nekonečnou dimenzí $\mathbb{H}$ prostorem dimenze konečné. Nejčastěji jde o po částech lineární funkce s malým nosičem. Jak je



Obrázek 3.1: Bázová funkce pro trojúhelníkové elementy



Obrázek 3.2: Ukázka půniku bázových funkcí

vidět na obrázku 3.2. Díky malému nosiči jsou nad každým elementem maximálně tři funkce nenulové (pro trojúhelníhové elementy). To znamená, že pro každý element stačí vypočítat pouze 9 integrálů $\int \nabla u \nabla v dx$. Nevýhodou ovšem je, že hranice oblasti ω může mít

velice složitou geometrii. Pro její věrohodnou aproximaci je tedy potřeba, aby diskretizační síť byla na jejím okolí dostatečně jemná. Pokud bychom zachovali jemnost diskretizace dovnitř oblasti, bude výsledná soustava lineárních rovnic obrovská, a tedy obtížně řešitelná. Pokud zvolíme směrem dovnitř oblasti síť hrubší, výsledná soustava bude menší, ovšem pravděpodobně špatně podmíněná. Ideální je, pokud má diskretizační síť nějakou speciální strukturu, které lze využít pro efekivní řešení výsledné soustavy lineárních rovnic. Jednou z metod, jak toho dosáhnout, je takzvaná metoda fiktivních oblastí (MFO).

3.2 Metoda fiktivních oblastí

uvažujme úlohu 3.1 s homogenní dirichletovou okrajovou podmínkou.

Metoda fiktivních oblastí spočívá ve vnoření oblasti se složitou geometrií ω do oblasti s pravidelnou geometrií Ω. A nahrazení úlohy 3.1 úlohou 3.3 definovanou následovně:

$$\begin{array}{cc} \int_\Omega \nabla\tilde{u}\nabla v dx = \int_\Omega \tilde{f} v dx & \forall v \in \mathbb{H}_0^1(\Omega) \\ +o.p. & na\ \partial\Omega \end{array} \tag{3.3}$$

a platí:

- $\tilde{f}$ je vhodné rozšíření f z oblasti ω na oblast Ω, $\tilde{f} \in \mathbb{L}^2$
- úloha je volena tak, aby restrikce $\tilde{u}|_\omega$ bylo řešením původního problému 3.1

Omezení, které nám zajistí, aby restrikce $\tilde{u}|_\omega$ bylo řešením původního problému, budeme vynucovat pomocí lagrangeovských multiplikátorů.

Podrobnější popis metody fiktivních oblastí a konstrukce duálního prostoru lagrangeových multiplikátorů můžeme nalézt v [7].

3.2.1 Řešení úlohy

Úloha tak, jak je definovaná v předchozí části, vede na řešení soustavy lineárních rovnic tvaru

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \end{pmatrix} \tag{3.4}$$

A je matice tuhosti. **B** matice transformace, která spolu s vektorem lagrangeových multiplikátorů λ zajišťuje splnění okrajových podmínek na hranici $\partial\omega$. **F** vektor zatížení. **X** je vektor uzlových hodnot řešení. Jednotlivé matice a vektory vypočteme následovně:

$$\begin{aligned} \mathbf{A} &= \{a_{ij}\},\ a_{ij} = \textstyle\int_\Omega \nabla v_i \nabla v_j dx, \qquad i,j = 1,\ldots,n, \\ \mathbf{B} &= \{b_{kj}\},\ b_{bj} = \textstyle\int_{l_k l_{k+1}} v_j ds, \qquad j = 1,\ldots,n,\ k = 1,\ldots,m, \\ \mathbf{F} &= \mathbf{C}(\tilde{f}),\ \mathbf{C} = \{c\}_{ij}, c_{ij} = \textstyle\int_\Omega v_i v_j dx, \qquad i,j = 1,\ldots,n, \end{aligned}$$

kde $\{v_j\}_{j=1}^n$ jsou bázové funkce $\tilde{\mathbb{V}}$. $\{l_k l_{k+1}\}_{k=1}^m$ je systém dělení hranice $\partial\omega$,n je počet bázových funkcí nad oblastí Ω, m je počet bázových funkcí nad hranicí $\partial\omega$.

Asi nejsnadnější by bylo sestavit celou matici

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

a příslušné vektory. A pro výpočet pak použít například gaussovu eliminaci. Náročnost výpočtu je obstojná. Ovšem už samotné sestavování jednotlivých matic je neúnosně drahé. A paměťové nároky na jejich uchovávání by byly obrovské. Matice je sice pouze 9 diagonální (pro bilineární prvky - rektangulaci oblasti). Avšak pro jemnou diskretizaci by pás, který se zaplňuje při gausové eliminaci byl značně široký. Pro výrazné snížení paměťových nároků je důležitá si všimnout vlastností bázových funkcí, a to jejich lineární separability. Můžeme tedy psát $v(x,y) = v_x(x)v_y(y)$[1]. Dosaďme si tedy tuto separovatelnou funkci do výrazu na

[1]Ukázka, jak vypadá část jedné bázové funkce nad jedním elementem pro bilineární prvky

$$v_i|_{element_i} = \frac{1}{4}(1+x)(1+y)$$

levé straně rovnice 3.3

$$\begin{aligned}\int_\Omega \nabla v_i \nabla v_j dxdy &= \int_\Omega \frac{\partial v_i}{\partial x}\frac{\partial v_j}{\partial x}dxdy + \int_\Omega \frac{\partial v_i}{\partial y}\frac{\partial v_j}{\partial y}dxdy \\ &= \int_\Omega \frac{\partial v_{xi}v_{yi}}{\partial x}\frac{\partial v_{xj}v_{yj}}{\partial x}dxdy + \int_\Omega \frac{\partial v_{xi}v_{yi}}{\partial y}\frac{\partial v_{xj}v_{yj}}{\partial y}dxdy \\ &= \int_\Omega v_{yi}v_{yj}\frac{\partial v_{xi}}{\partial x}\frac{\partial v_{xj}}{\partial x}dxdy + \int_\Omega v_{xi}v_{xj}\frac{\partial v_{yi}}{\partial y}\frac{\partial v_{yj}}{\partial y}dxdy \\ &= \int(v_{yi}v_{yj}\int(\frac{\partial v_{xi}}{\partial x}\frac{\partial v_{xj}}{\partial x})dx)dy + \int(v_{xi}v_{xj}\int(\frac{\partial v_{yi}}{\partial y}\frac{\partial v_{yj}}{\partial y})dy)dx\end{aligned}$$

A protože

$$\mathbf{A}_s = \{a^s\}_{ij}, \qquad a^s_{ij} = \int(\frac{\partial v_{si}}{\partial s}\frac{\partial v_{sj}}{\partial s})ds,$$

$$\mathbf{M}_s = \{m^s\}_{ij}, \qquad m^s_{ij} = \int(v_{si}v_{sj})ds,$$

kde $s = x$ respektive y. $\mathbf{A}_s$ je jednodimenzionální matice tuhosti, $\mathbf{M}_s$ jednodimenzionální matice hnotnosti. Tyto matice lze pro uniformní síť jednoduše vyčíslit

$$\mathbf{A}_s = \frac{1}{h_s}\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \qquad \mathbf{M}_s = \frac{h_s}{6}\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{pmatrix}. \tag{3.5}$$

Pomocí kroneckerova součinu potom můžeme psát

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{A}_y \tag{3.6}$$

Tímto výrazně snížíme náročnost sestavování matice $\mathbf{A}$. Ovšem my půjdeme ještě dál a ukážeme si přístupy, při kterých nebude potřeba vůbec celou matici $\mathbf{A}$ sestavovat.

3.3 Základní myšlenka uváděných algoritmů

Všechny tři algoritmy, které budou postupně popsány mají stejnou základní kostru. Chceme řešit soustavu rovnic

$$\mathbf{A}\mathbf{x} + \mathbf{B}^T\lambda = \mathbf{f} \tag{3.7}$$

$$\mathbf{Bx} = 0. \tag{3.8}$$

Z rovnice 3.7 si vyjádříme neznámou $\mathbf{x}$.

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}^T\lambda) \tag{3.9}$$

po dosazení do druhé rovnice 3.8 dostáváme

$$\mathbf{BA}^{-1}\mathbf{B}^T\lambda = \mathbf{B}(\mathbf{A}^{-1}\mathbf{f}) \tag{3.10}$$

Tuto soustavu ve všech případech řešíme metodou sdružených gradientů. Pravou stranu vyčíslíme tak, jak jsou uvedeny závorky, abychom minimalizovali počet potřebných oerací. Matici $\mathbf{BA}^{-1}\mathbf{B}^T$ ze stejného důvdu nevyčíslujeme, ale vždy, když je potřeba touto maticí násobit vektor. Postupujeme odzadu $\mathbf{e} = \mathbf{B}^T(\mathbf{A}^{-1}(\mathbf{Bg}))$.

Takto získáme vektor lagrangeových multiplikátorů λ a dosadíme jej zpět do rovnice 3.7. Získali jsme vektor uzlových hodnot řešení $\mathbf{x}$. Jediné, co zatím zůstává nevyřešené, je výpočet $\mathbf{A}^{-1}\mathbf{y}$. A právě v tomto bodě se jednotlivé metody od sebe více či méně liší.

3.4 Nahrazení inverze matice sdruženými gradienty

Chceme získat vektor $\mathbf{y} = \mathbf{A}^{-1}\mathbf{h}$. Máme ovšem omezující podmínku. Potřebujeme jej získat bez nutnosti vyčíslit matici $\mathbf{A}$.

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{A}_y$$

$$\mathbf{A}^{-1} = (\mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes)^{-1}$$

Je tedy jasné, že tudy cesta nepovede. Není těžké si však uvědomit, že výpočet $\mathbf{y} = \mathbf{A}^{-1}\mathbf{h}$ lze nahradit řešením soustavy $\mathbf{Ay} = \mathbf{h}$. Na tuto vnitřní soustavu můžeme opět aplikovat metodu sdružených gradientů. Tím se vyhneme potřebě získat inverzi $\mathbf{A}$. A dokonce i nutnosti vyčíslovat celou matici. Vždy, když v metodě sdružených gradientů bude potřeba

násobit vektor maticí $\mathbf{A}$, provedeme to podle pravidla 1.18 pro násobení kroneckerova součinu s vektorem následovně:

$$\mathbf{Aw} = (\mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{A}_y)\mathbf{w}, \text{[2]}$$

$$(\mathbf{Aw}) = \mathbf{A}_x\mathbf{W}\mathbf{M}_y + \mathbf{M}_x\mathbf{W}\mathbf{A}_y. \tag{3.11}$$

$\mathbf{W}$ je vektor $\mathbf{w}$ přeskládaný na matici tak, že $\mathbf{W} \in \mathbb{R}^{n_x \times n_y}$. Výsledkem poslední operace 3.11 je matice, kterou je podle potřeby možné přeskládat na očekávaný vektor. Někdy je ovšem lepší ponechat výsledek jako matici a ušetřit tak přeskládávání, které jinak také spotřebuje nějaké takty procesoru.

Kdybychom násobili celou maticí $\mathbf{A}$, dostali bychom se ke složitosti $n_x^2 n_y^2$. Předpokládejme, že $n = n_x = n_y$. Pak řádová složitost této operace je $o(n^4)$. Kdežto při použití pravidla pro násobení kroneckerova součinu dostáváme $4n^3$, což odpovídá složitosti $o(n^3)$. Vzhledem k situaci, že se toto násobení objevuje v každém cyklu metody sdružených gradientů[3], je to významná úspora.

3.5 Inverze pomocí Fourierových transformací

V tomto případě využijeme možnosti provést spektrální rozklad matice efektivně pomocí Fourierových trasformací. Matici vzniklou diskretizací jednodimenzionální úlohy metodou konečných prvků tak můžeme rozepsat

$$\mathbf{A}_x = \hat{\mathbf{T}}\mathbf{D}_x\mathbf{T},$$

[2] kde $\mathbf{A} \in \mathbb{R}^{n_x n_y \times n_x n_y}$, $\mathbf{A}_x, \mathbf{M}_x \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{A}_y, \mathbf{M}_y \in \mathbb{R}^{n_y \times n_y}$, $\mathbf{w} \in \mathbb{R}^{n_x n_y}$

[3] Dokonce se objevuje v cyklu sdružených gradientů(CG), které jsou opět krokem nadřazeného cyklu CG

kde $\mathbf{T}$ je matice Fourierovy respektive sinovy transformace. Celou matici $\mathbf{A}$ můžeme tedy psát ve tvaru:

$$\begin{aligned}\mathbf{A} &= \mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{A}_y = \\ &= \hat{\mathbf{T}}\mathbf{D}_{A_x}\mathbf{T} \otimes \hat{\mathbf{T}}\mathbf{D}_{M_y}\mathbf{T} + \hat{\mathbf{T}}\mathbf{D}_{M_x}\mathbf{T} \otimes \hat{\mathbf{T}}\mathbf{D}_{A_y}\mathbf{T} = \\ &= (\hat{\mathbf{T}} \otimes \hat{\mathbf{T}})(\mathbf{D}_{A_X}\mathbf{T} \otimes \mathbf{D}_{M_y}\mathbf{T}) + (\hat{\mathbf{T}} \otimes \hat{\mathbf{T}})(\mathbf{D}_{M_y}\mathbf{T} \otimes \mathbf{D}_{A_y}\mathbf{T}) = \\ &= (\hat{\mathbf{T}} \otimes \hat{\mathbf{T}})(\mathbf{D}_{A_x} \otimes \mathbf{D}_{M_y} + \mathbf{D}_{M_x} \otimes \mathbf{D}_{A_y})(\mathbf{T} \otimes \mathbf{T})\end{aligned}$$ [4]

Díky fourierovým transformacím tedy jednoduše získáme diagonální rozklad matice $\mathbf{A}$. A vektor $\mathbf{y} = \mathbf{A}^{-1}\mathbf{h}$ takto:

$$\mathbf{y} = (\hat{\mathbf{T}} \otimes \hat{\mathbf{T}})(\mathbf{D}_{A_x}^{-1} \otimes \mathbf{D}_{M_y}^{-1} + \mathbf{D}_{M_x}^{-1} \otimes \mathbf{D}_{A_y}^{-1})(\mathbf{T} \otimes \mathbf{T})\mathbf{h}$$

$$\mathbf{D}^{-1} = \mathbf{D}_{A_x}^{-1} \otimes \mathbf{D}_{M_y}^{-1} + \mathbf{D}_{M_x}^{-1} \otimes \mathbf{D}_{A_y}^{-1}$$

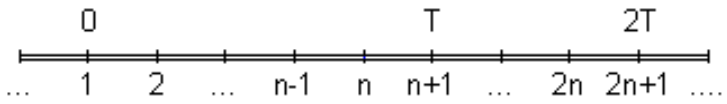
Ve skutečnosti však nebudeme sestavovat inverzi diagonální matice, ale sestavíme pouze vektor diagonálních prvků matice $\mathbf{D}$. Pak stačí pouze provést dvourozměrnou fourierovu transformaci vektoru $\mathbf{h}$ a vydělit příslušné prvky výsledku prvky na odpovídajících pozicích diagonály $\mathbf{D}$. Po aplikaci inverzní dvojrozměrné transformace máme vektor $\mathbf{y}$.

Díky existenci algoritmu rychlé fourierovy transformace, která má složitost $o(n \log n)$ (její dvourozměrná varianta $o(n^2 \log n)$),je celé násobení inverzní maticí $\mathbf{A}^{-1}$ velice efektivní a má složitost pouze $o(n^2 \log n)$.

3.5.1 Specifika Fourierovy transformace

Jak víme, pokud chceme udělat fourierův rozvoj funkce. Musí být tato funkce periodická. Není nic divného, že to samé platí pro tuto transformaci i v diskrétním tvaru. Mějme dánu funkci definovanou na intervalu T. Dělme tento interval na n dílů podle obrázku 3.3 Výsledné řešení bude tedy také periodická funkce. To se na matici tuhosti a hmotnosti

[4] $\hat{\mathbf{T}}$ je matice inverzní transformace. Například pro sinovou transformaci se $\hat{\mathbf{T}} = \mathbf{T}$ a paltí $\hat{\mathbf{T}}\mathbf{T} = \mathbf{I}$. Matice sinové transformace je ortonormální.



Obrázek 3.3: Diskretizace oblasti pro fourierovu transformaci

projeví následujícím způsobem

$$\mathbf{A}_s = \frac{1}{h_s}\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & \cdots & -1 \\ -1 & 2 & -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ -1 & 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \qquad \mathbf{M}_s = \frac{h_s}{6}\begin{pmatrix} 4 & 1 & 0 & 0 & 0 & \cdots & 1 \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 1 & 4 \end{pmatrix}.$$

$$s = x \text{ resp. } y,\ h_s = \frac{1}{n_s},\ n_s \text{ je dělení ve směru osy } x \text{ resp. } y$$

Nevýhodou ovšem je, že takto definované matice jsou singulární. My jsme si při numerické realizaci pomohli malou berličkou a neřešili jsme rovnici $-\Delta u = f$, ale rovnice $-\Delta u + u = f$. Tím jsme zaručili regularitu matice $\mathbf{A}$. Na výslednou soustavu to mělo pouze tento vliv:

$$\mathbf{A} = \mathbf{A}_x \otimes \mathbf{M}_y + \mathbf{M}_x \otimes \mathbf{A}_y + \mathbf{M}_x \otimes \mathbf{M}_y.$$

Opravdu silnou zbraní Fourierovy transformace je fakt, že pro získání vlastních číslel matice, diaginizovatelné touto transformací, stačí provést transformaci pouze prvního sloupce matice. Získáme tedy spektrální rozklad za cenu pouze $n \log n$ operací.

3.5.2 Specifika sinovy transformace

Jestliže u Fourierovy transformace musí být funkce periodická, u sinovy trasformace musí být ještě k tomu lichá vzhledem ke středu intervalu. Z toho plyne, že výsledné řešení musí splňovat homogenní dirichletovy okrajové podmínky na hranici fiktivní oblasti[5]. Výhodou

[5]toto lze pěkně vidět na obrázku 2.5 Součet sinové řady

je, že tím máme zajištěnu regularitu soustavy a počítáme uzlové hodnoty pouze ve vnitřních bodech diskretizace. Matice tuhosti a hnotnosti pro použití sinové transformace vypadájí přesně jako matice 3.5.

Vlastní čísla matice tuhosti přimo známe:

$$\lambda_{Aj} = 4\sin^2\left(\frac{j\pi h}{2}\right), \qquad h = \frac{1}{n+1},$$

kde n je počet uzlů uvnitř 1D oblasti. Vlastní čísla matice hmotnosti musíme vypočítat pomocí diskrétní sinové transformace. Tyto vlastní čísla získáme za cenu složitosti $o(n^2 \log n)$.

Důležité je nezapomenout na fakt, že vektor předkládaný rychlé fourierově transformaci, musí mít rozměr 2^k, kde $k = 1, 2, \ldots$ Jinak by si tento algoritmus neuchoval svou efektivitu.

3.6 Numerické experimenty

Jako oblast ω zvolme jednotkovou kružnici s parametrizací $(\cos t, \sin t)$. Potřebujeme najít nějakou funkci, která splňuje homogenní dirichletovy okrajové podmínky na této oblasti. Takovou funkcí je například funkce $u = 1 - x^2 - y^2$. A právě tuto funkci si vybereme pro numerické testy.

Dopočítáme pravou stranu f:

$$f = -\Delta u = -\frac{\partial^2 u}{\partial^2 x} - \frac{\partial^2 u}{\partial^2 y} = 4$$

Přesné zadání řešené úlohy má tedy tvar:

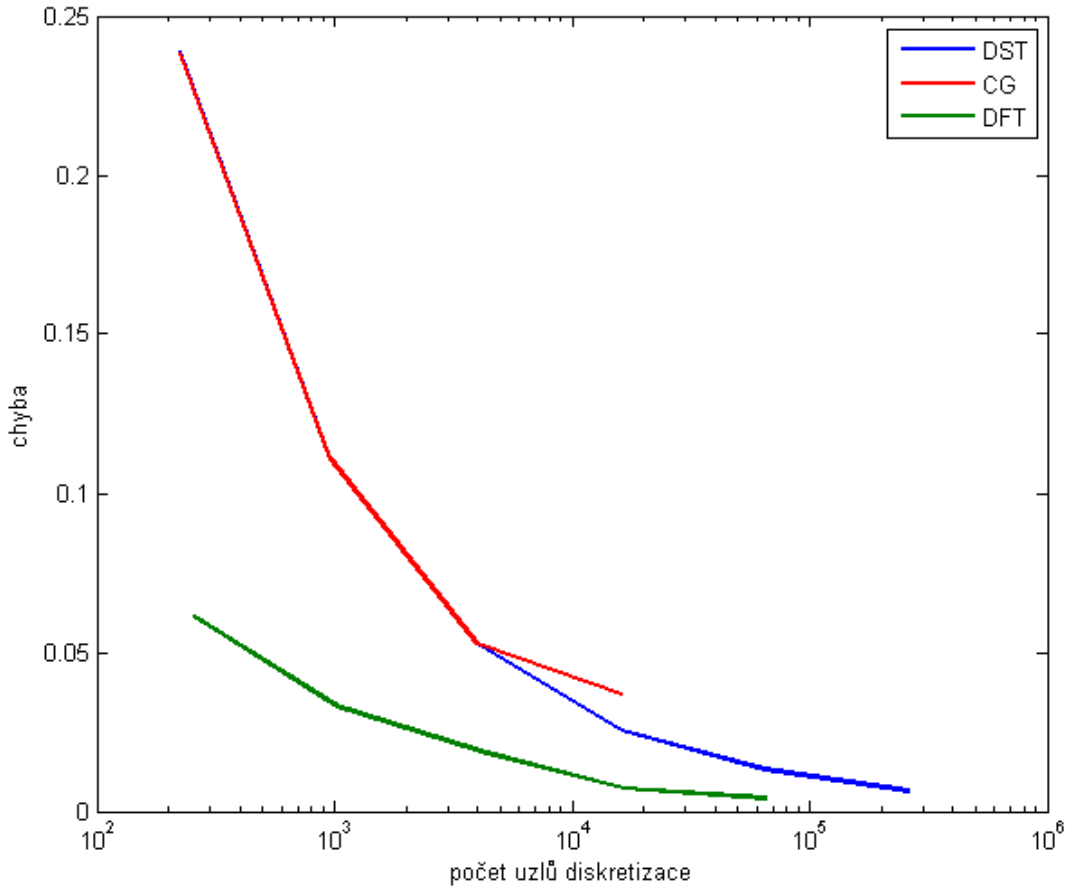
$$-\Delta u = 4,$$
$$u(\cos t, \sin t) = 0, \qquad t \in (0, 2\pi).$$

Poznámka 3.6.1. Pro fourierovu transformaci řešíme úlohu

$$-\Delta u + u = f$$

n	DST	CG	n	DFT
255	0,2386	0,2382	256	0,613
961	0,1107	0,1107	10246	0,0331
3969	0,0525	0,0527	4096	0,0189
16125	0,0255	0,0369	16384	0,0072
65025	0,0131		65536	0,0042
261121	0,0065			

Tabulka 3.1: Vývoj chyby řešení v L^2 normě



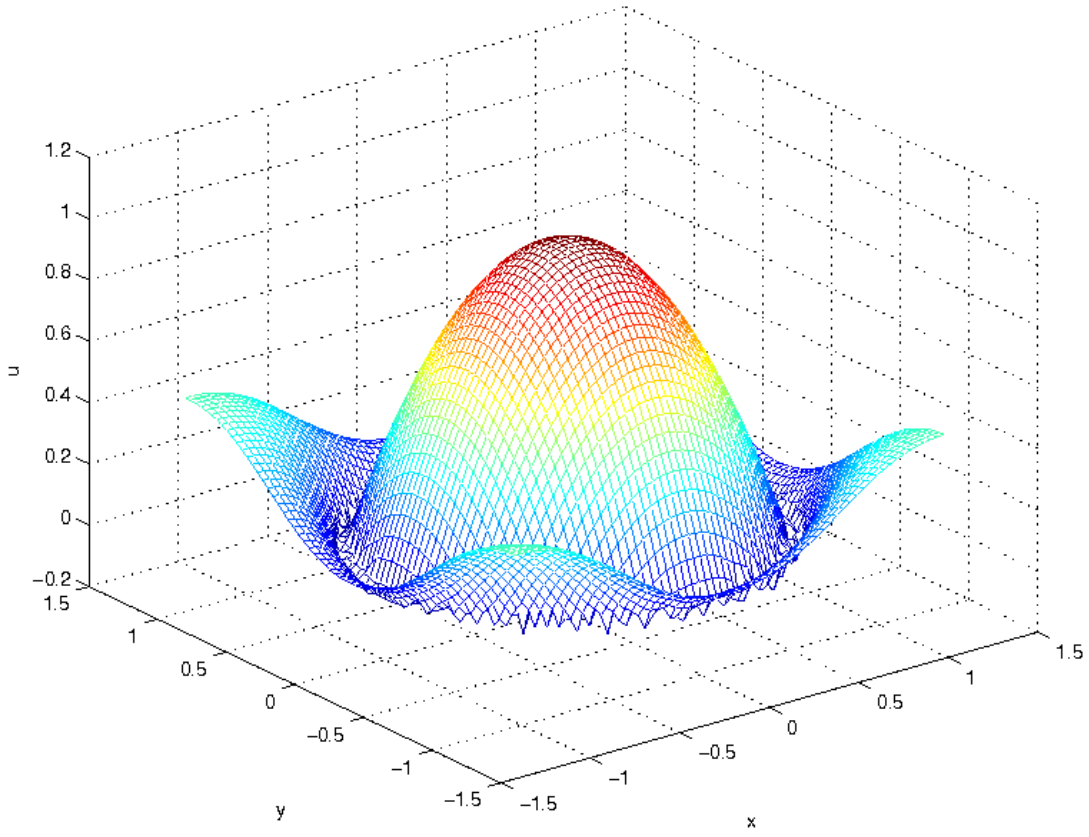
Obrázek 3.4: Vývoj chyby

Závěr

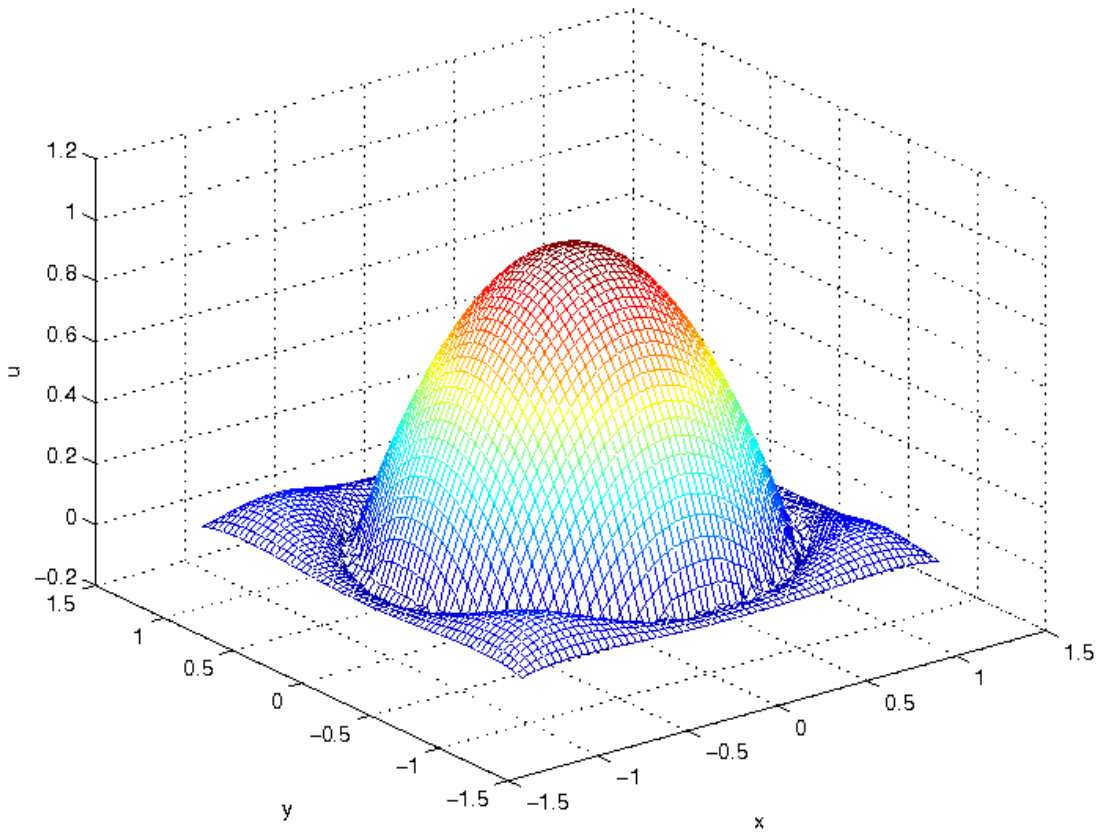
Metoda fiktivních oblastí se ukazuje jako velice efektivní řešič parciálních diferenciálních rovnic. Paměťové nároky, bez ztráty informace, už snad ani nemůžou být menší. Díky uniformní síti je i preprocesing relativně levný. Matici tuhosti nemusíme nijak algoritmicky sestavovat, protože známe přímo její tvar (stejně tak i matici hmotnosti). Přibylo sice sestavení matice transformace. Ta je ovšem vždy v jednom rozměru násobně menší než matice tuhosti. Pomocí rychlých řešičů založených na Fourierových transformacích je i výpočet velice efektivní. Rychlá fourierova transformace je taká dobře paralelizovatelná. Nevýhodou je, že okrajová podmínka na hranici dané oblasti je splněna pouze v integrálním smyslu, tj. integrální průměr nad určitým úsekem hranice je roven nule (pro dirichletovy okrajové podmínky). To způsobuje velice pomalou konvergenci řešení v okolí hranice. Tato chyba se samozdřejmě šíří i směrem dovnitř oblasti. Musíme tedy počítat větší soustavy, abychom dosáhli stejné přesnosti.

Je snadno vidět, že řešič pomocí Fourierovy transformace je efektivnější než pomocí sinovy transformace. U sinové transformace se totiž aplikuje rychlá fourierova transformace na vektor o velikosti $2n + 2$, kde n je velikost původního vektoru. Tudíž má dva krát větší jak paměťové, tak výpočetní nároky. Nemá ovšem problémy s regularitou soustavy, což u Fourierovy trasformace může nastat. V tom případě je nutné použít pro výpočet pseudoinverze. To sebou nese určité komplikace.

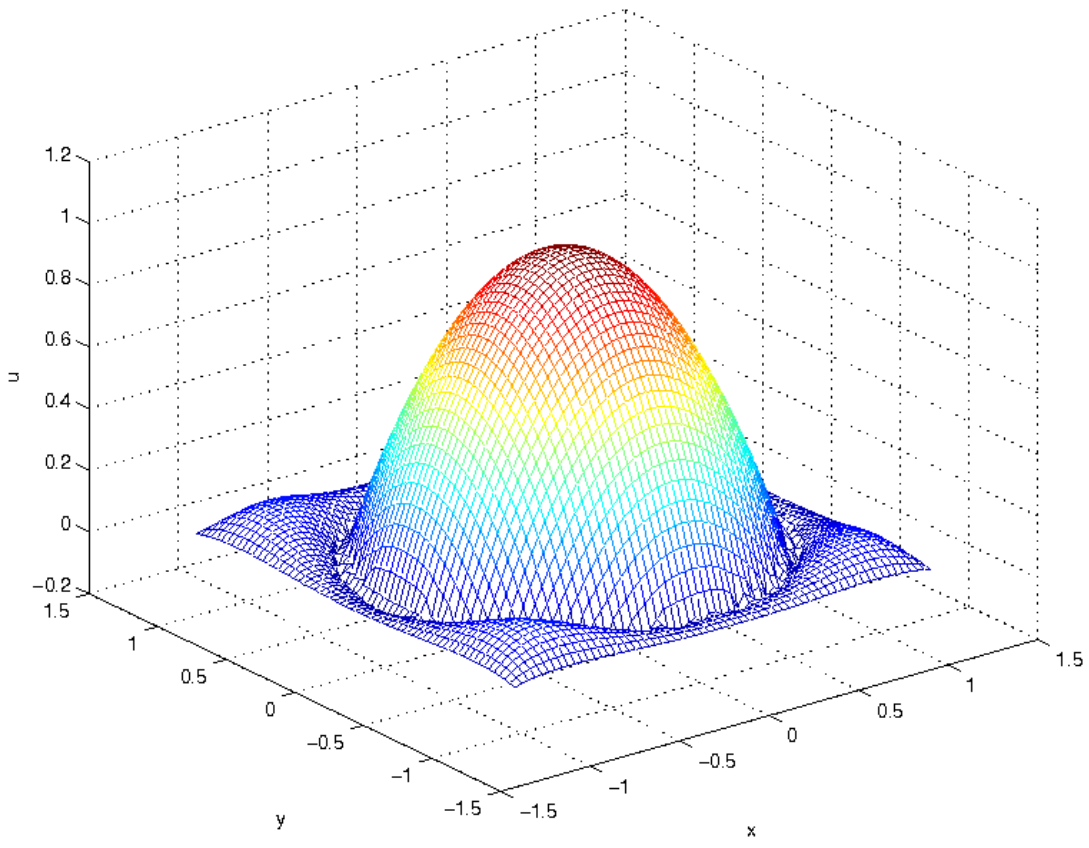
Další uplatnění by metoda fiktivních oblastí mohla mít v metodách rozložení oblastí. Kdy by se jednotlivé podoblasti vložili do fiktivních oblastí. Tím by se otevřela cesta řešit tyto dílčí soustavy velice efektivně. Navíc dělení podoblastí si volíme sami a tím pádem by i konvergence v blízkosti hranice mohla být výrazně lepší.



Obrázek 3.5: Řešení pomocí DFT



Obrázek 3.6: Řešení pomocí DST



Obrázek 3.7: Řešení pomocí Sdružených gradientů

Literatura

[1] *Radek Kučera* **Complexity of an algorithm for solving saddle point systems with singular blocks arasing in wavelet – Galerkin discretizations**, Applications of Mathematics, 2005

[2] *Tom Lyche* **Fast Poisson Solvers and FFT**,University of Oslo, Norway, 2002

[3] *W. L. Briggs, V. E. Henson*,**The DFT**, SIAM Philadelphia, 1995.

[4] *Zdeněk Dostál* **Lineární algebra**, VŠB - TUO 2004

[5] *Yousef Saad* **Iterative methods for Sparse Linear systems** Second Edition, 2000

[6] *Zdeněk Dostál* **Optimal Quadratic Programming Algorithms** Springer 2009

[7] *Tomáš Kozubek* **Metody fiktivních oblastí pro řešení úloh tvarové optimalizace**, Diplomová práce VŠB - TUO 1998