Modelování a simulace v elektrotechnice

Iterační metody řešení soustav lineárních algebraických rovnic

František Mach

Katedra teoretické elektrotechniky
Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

3. cvičení, 6.10.2016

FAKULTA
ELEKTROTECHNICKÁ
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Obsah

Uvažujme soustavu lineárních algebraických rovnic ve tvaru

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

kde $\mathbf{A}$ je matice soustavy, $\boldsymbol{x}$ vektor neznámých a $\boldsymbol{b}$ vektor pravých stran. Metody řešení soustav lze pak rozdělit na

- metody přímé (Gaussova eliminační metoda, LU rozklad, …) a
- metody iterační, nebo-li nepřímé (Jacobiho metoda, Gauss-Seidelova metoda, …).

Přímými metodami získáme teoreticky přesné řešení iterativním procesem o konečně mnoha krocích. Iteračními metodami pak iterativním procesem o nekonečně mnoha krocích.

Použití přímých metod pro řešení velmi rozsáhlých úloh s velkým počtem neznámých a tedy velkým počtem rovnic je často velmi komplikované vzhledem k velké výpočetní a paměťové náročnosti. Iterační metody mají obecně nižší paměťové nároky a jsou také při použití v určitých případech rychlejší. Výhodou iteračních metod je také jejich snadná paralelizace, velkou nevýhodou pak nižší stabilita při řešení určitých typů úloh.

Mezi důležité výhody iteračních metod patří:

- aproximace řešení v každém iteraci (vždy existuje řešení, které se postupně zpřesňuje),
- algebraicky jednoduché operace (násobení matice a vektoru),
- nízká citlivost na zaokrouhlovací chyby než u metod přímých.

Iterační metody umožňují řešení soustav lineárních algebraických rovnic pomocí postupného přibližování se k přesnému řešení. Pro soustavu rovnic

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

zvolíme nejprve počáteční aproximaci řešení $\boldsymbol{x}_0$ a z ní následně počítáme posloupnost aproximací $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k$ pro kterou platí, že

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}\,.$$

Přesné řešení tedy získáme po výpočtu nekonečně mnoha aproximací. To je však nereálné a proto se využívá vždy jen konečný počet aproximací, které vede k určité přesnosti řešení. V každém kroku metody je tedy vždy vyhodnocena chyba řešení, která slouží jako zastavovací kritérium metody.

Abychom mohli iterativně počítat dílčí aproximace řešení, musíme využít zobrazení (předpis), který nám umožní z aproximace předchozí $\boldsymbol{x}_k$ vypočítat novou $\boldsymbol{x}_{k+1}$. Zobrazení lze obecně zapsat ve tvaru

$$\boldsymbol{x}_{k+1} = \Phi(\boldsymbol{x}_k)\,,$$

kde $\boldsymbol{x}_i$ značí k-tou iteraci metody.

Pro použití iteračních metod nejprve převedeme soustavu

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

na ekvivalentní soustavu v iteračním tvaru

$$\boldsymbol{x} = \mathbf{C}\boldsymbol{x} + \boldsymbol{d}\,,$$

kde $\mathbf{C}$ je iterační matice a $\boldsymbol{d}$ sloupcový vektor. Pro výpočet nové aproximace $\boldsymbol{x}_{k+1}$ soustavy využijeme předpis

$$\boldsymbol{x}_{k+1} = \Phi(\boldsymbol{x}_k) = \mathbf{C}\boldsymbol{x}_k + \boldsymbol{d}\,,\ k = 0, 1, 2, ..., \infty\,,$$

kde $\boldsymbol{x}_0$ je libovolně zvolená počáteční aproximace. Jestliže posloupnost aproximací $\boldsymbol{x}_i$ konverguje k přesnému řešení $\boldsymbol{x}$, pak je $\boldsymbol{x}_\infty$ řešením ekvivalentní i původní soustavy rovnic.

Vzhledem k tomu, že přesné řešení získáme po nekonečně mnoha iteracích, musíme zavést kritérium, které výpočet zastaví po konečném počtu iterací, jejichž výsledek bude odpovídat aproximaci přesného řešení se zvolenou přesností. Zavedeme tedy požadovanou chybu řešení, kterou lze definovat jako kritérium

$$|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k| \leq \varepsilon\,,$$

kde ε značí požadovanou chybu, které se snažíme dosáhnout a platí, že $\varepsilon > 0$.

Řešme níže uvedenou soustavu lineárních algebraických rovnic.

$$\begin{aligned} 11x_1 + 2x_2 + x_3 &= 15 \\ x_1 + 10x_2 + 2x_3 &= 16 \\ 2x_1 + 3x_2 - 8x_3 &= 1 \end{aligned}$$

Soustavu nejprve převedeme na iterační tvar. Možností jak provést převod je zřejmě více, uveďme tedy alespoň dva způsoby.

$$\begin{aligned} x_1 &= \frac{1}{11}(15 - 2x_2 - x_3) \\ x_2 &= \frac{1}{10}(16 - x_1 - 2x_3) \\ x_3 &= \frac{1}{8}(-1 + 2x_1 + 3x_2) \end{aligned}$$

$$\begin{aligned} x_1 &= 15 - 10x_1 - 2x_2 - x_3 \\ x_2 &= 16 - x_1 - 9x_2 - 2x_3 \\ x_3 &= -1 + 2x_1 + 3x_2 - 7x_3 \end{aligned}$$

Pomocí iteračního tvaru rovnic pro oba uvedené způsoby proveďme několik iterací.

```
C = [0,-2/11,-1/11; -1/10,0,-2/10; 2/8,3/8,0]
d = [15/11; 16/10; -1/8]

x = [0;0;0];
for i = 1:10
    xk = x;
    x = C*xk + d
end
```

```
C = [-10,-2,-1; -1, -9, -2; -2, 3, -7]
d = [15; 16; -1]

x = [0;0;0];
for i = 1:10
    xk = x;
    x = C*xk + d
end
```

Srovnejme nyní výsledky řešení získané oběma způsoby.

$$\boldsymbol{x}_1 = \begin{bmatrix} 1.4 \\ 1.6 \\ -0.13 \end{bmatrix}, \boldsymbol{x}_{10} = \begin{bmatrix} 1.06 \\ 1.36 \\ 0.65 \end{bmatrix} \qquad \boldsymbol{x}_1 = \begin{bmatrix} 15 \\ 16 \\ -1 \end{bmatrix}, \boldsymbol{x}_{10} = \begin{bmatrix} -4e^{10} \\ -2e^{10} \\ -3e^{9} \end{bmatrix}$$

Je jasné, že se výsledky značně liší. Porovnáme-li výsledky s přesným řešením původní soustavy pomocí Gassovy eliminační metody ($x_1 = 1.06, x_2 = 1.36, x_3 = 0.65$), můžeme konstatovat, že první uvedený způsob konverguje k přesnému řešení. Naproti tomu druhý uvedený způsob silně diverguje. Konvergence metody tedy silně závisí na způsobu, jak vytvoříme ekvivalentní soustavu!

Jacobiho metoda

Předpokládejme, že diagonální prvky matice soustavy $\mathbf{A}$ řádu n (počet neznámých a tím také počet potřebných rovnic k řešení) jsou nenulové, tj. $a_{ii} \neq 0$. Pro i-tou rovnici soustavy, kterou lze obecně rozepsat ve tvaru

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i \;\; i = 1, 2, \ldots, n$$

vyjádříme i-tou neznámou a převedeme tak rovnici na tvar

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_i - \sum_{j=i+1}^{n} a_{ij}x_i \right) \;, \; i = 1, 2, ..., n \,.$$

Každá k-ta iterace Jacobiho metody pak již představuje jen vyčíslení nové aproximace řešení $x_{i,k+1}$ pomocí rekurentního vzorce

$$x_{i,k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_{i,k} - \sum_{j=i+1}^{n} a_{ij}x_{i,k} \right) \;, \; i = 1, 2, \ldots, n \;, \; k = 0, 1, \ldots, \infty \,.$$

```
function x = jacobi(A, b, x0, epsilon)
n = length(b);
x = x0;

for k = 1:1e4
    % vypocet nove aproximace
    xk = x;
    for i = 1:n
        x(i) = 1/A(i,i) * (b(i) - (A(i,[1:i-1]) * xk([1:i-1])) ...
                                - (A(i,[i+1:n]) * xk([i+1:n])));
    end

    % vypocet relativni chyby reseni
    err = max(abs(x-xk))/(max(x));

    % kontrola na pozadovanou chybu reseni
    if err <= epsilon
        return
    end
end
end
```

Gaussova-Seidelova metoda

Gaussova-Seidelova metoda je velmi podobná Jacobiho metodě. Jediný rozdíl mezi oběma metodami je v tom, že při použití Gauss-Seidelovy metody využíváme při výpočtu nové aproximace řešení $\boldsymbol{x}_{k+1}$ vždy nejnovější hodnoty x_i, které máme k dispozici.

Tedy, zatímco při využití Jacobiho metody využíváme pro výpočet nové aproximace řešení $x_{i,k+1}$ rekurentní vzorec

$$x_{i,k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{i,k} - \sum_{j=i+1}^{n} a_{ij} x_{i,k} \right) , \; i = 1, 2, \ldots, n \, , \; k = 0, 1, \ldots, \infty \, ,$$

při využití metody Gauss-Seidelovy metody využijeme vzorec

$$x_{i,k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{i,k+1} - \sum_{j=i+1}^{n} a_{ij} x_{i,k} \right) , \; i = 1, 2, \ldots, n \, , \; k = 0, 1, \ldots, \infty \, ,$$

a tím tedy využíváme pro výpočet $x_{i>j}$ vždy nejnovější možné hodnoty.

```
function x = gauss_seidel(A, b, x0, epsilon)
n = length(b);
x = x0;

for k = 1:1e4
    % vypocet nove aproximace
    xk = x;
    for i = 1:n
        x(i) = 1/A(i,i) * (b(i) - (A(i,[1:i-1]) * x([1:i-1])) ...
                                - (A(i,[i+1:n]) * xk([i+1:n])));
    end

    % vypocet relativni chyby reseni
    err = max(abs(x-xk))/(max(x));

    % kontrola na pozadovanou chybu reseni
    if err <= epsilon
        return
    end
end
end
```