Modelování a simulace v elektrotechnice

Pokročilé metody řešení obyčejných diferenciálních rovnic

František Mach

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

5. cvičení, 20.10.2016

FAKULTA
ELEKTROTECHNICKÁ
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Obsah

Uvažujme obyčejnou diferenciální rovnici ve tvaru

$$y' = f(x, y(x)) \, ,$$

kterou budeme řešit pomocí Eulerovy metody. Aproximaci řešení $y(x_{i+1})$ tak budeme počítat podle iteračního předpisu

$$y(x_{i+1}) = y(x_i) + f(x_i, y(x_i))h \, .$$

Lokální diskretizační chyba τ_{i+1} daného kroku bude přitom přímo úměrná druhé mocnině diskretizačního kroku h a lze jí tedy obecně vyjádřit ve tvaru $\tau_i = ch^2$, kde c je předem neznámá konstanta.

Proveďme výpočet nové aproximace řešení, tentokrát však s polovičním diskretizačním krokem $h/2$. Abychom dosáhli stejného kroku jako pro $y^*(x_{i+1})$ (hvězdička pro přehlednost značí nově provedené řešení), musíme provést iteraci nejprve pro $x_{i+\frac{1}{2}}$

$$y^*(x_{i+\frac{1}{2}}) = y(x_i) + f(x_i, y(x_i))\frac{h}{2}$$

a následně již pro x_{i+1} pomocí již známého kroku $x_{i+\frac{1}{2}}$, tedy

$$y^*(x_{i+1}) = y^*(x_{i+\frac{1}{2}}) + f(x_{i+\frac{1}{2}}, y^*(x_{i+\frac{1}{2}})) \, \frac{h}{2} \, .$$

Lokální diskretizační chyba τ^*_{i+1} musí být potom rovna

$$\tau^*_{i+1} = c\left(\frac{h}{2}\right)^2 + c\left(\frac{h}{2}\right)^2 = 2c\left(\frac{h}{2}\right)^2 = \frac{1}{2}ch^2 = \frac{1}{2}\tau_{i+1}\,,$$

tedy lokální diskretizační chyba τ^*_{i+1} získaná výpočtem s polovičním krokem $h/2$ je poloviční než diskretizační chyba τ_{i+1} získaná výpočtem s krokem h.

Pokud budeme dále uvažovat, že neznámá konstanta c je v rámci daného intervalu $< x_i, x_{i+1} >$ konstantní (tato úvaha bude platit pro malé h), můžeme lokální diskretizační chybu vyjádřit ve tvaru

$$\tau^*_{i+1} = y(x_{i+1}) - y^*(x_{i+1})\,.$$

Takto vyjádřená diskretizační chyba, která závisí jen na hodnotě dvou provedených řešení s rozdílným krokem h, může být využita v průběhu řešení k volbě kroku, tak aby nebyla překročena požadovaná velikost chyby τ.

Jednoduchým příkladem může být algoritmus, který provádí postupné půlení kroku h dokud odhad chyby není menší než požadovaná hodnota, tedy $\tau^*_{i+1} < \tau$. Ve chvíli, kdy je tato podmínka splněna, pokračuje algoritmus ve výpočtu opět s krokem h. Během výpočtu tak dochází k adaptivnímu dělení kroku (krok h se v průběhu řešení mění).

Nevýhodou tohoto adaptivního přístupu a odhadu chyby metodou polovičního kroku obecně je nutnost provádět výpočet $y(x_{i+1})$ vícekrát. Odhad chyby metodou půlení intervalu je nejjednodušším přístupem. Mezi pokročilejší a nejčastěji používané metody pak patří Fehlbergova metoda.

```
function [x, y, tau] = aeuler(fce, interval, y0, h, tolerance)
x = [interval(1)];
y = [y0];
tau = [];

i = 1;
while x(end) < interval(2)
    hi = h;

    for j = 1:1e2
        y0 = y(i,:) + hi * fce(x(i), y(i,:))';

        % odhad chyby metodou polovicniho kroku
        hi = hi/2;
        y1 = y(i,:) + hi * fce(x(i), y(i,:))';
        y2 = y1 + hi * fce(x(i)+hi, y1)';
        taui = max(abs(y0 - y2));

        % kontrola nastavene tolerance
        if tolerance > taui
            x(i+1) = x(i) + 2*hi;
            y(i+1,:) = y0;
            tau(i) = taui;
            break
        end
    end

    i = i + 1;
end
end
```

Metody Runge-Kutta využívají pro vyčíslení aproximace řešení y_{i+1}, výpočet derivace vyšších řádů a pro stanovení nového kroku berou vážený průměr těchto derivací s váhami α_i. Každý krok metody se tedy určuje na základě vztahu

$$y(x_{i+1}) = y_i + \sum_{j=1}^{r} \alpha_j k_j \cdot h \,,$$

přičemž pro aproximaci derivací k_j platí

$$\begin{aligned} k_1 &= f(x_i, y(x_i)) \\ &\vdots \\ k_j &= f(x_i + h\lambda_j, y(x_i) + h\mu_j k_{j-1}) \,,\ k > 1 \,. \end{aligned}$$

Z uvedeného odhadu derivace k_1 tak plyne, že Eulerova metoda je Runge-Kuttova metoda prvního řádu. Speciální volbou koeficientů α, λ a μ pak získáme Runge-Kuttovi metody vyšších řádů.

Standardní metoda 2. řádu (modifikovaná Eulerova metoda)

$$
\begin{aligned}
y(x_{i+1}) &= y(x_i) + hk_2 \\
k_1 &= f(x_i, y(x_i)) \\
k_2 &= f(x_i + \frac{1}{2}h, y(x_i) + \frac{1}{2}hk_1)
\end{aligned}
$$

Crank-Nicolsonova metoda

$$
\begin{aligned}
y(x_{i+1}) &= y(x_i) + h\frac{k_1 + k_2}{2} \\
k_1 &= f(x_i, y(x_i)) \\
k_2 &= f(x_{i+1}, y(x_i) + hk_1)
\end{aligned}
$$

Standardní metoda 4. řádu (4RK)

$$
\begin{aligned}
y(x_{i+1}) &= y(x_i) + h\frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \\
k_1 &= f(x_i, y(x_i)) \\
k_2 &= f(x_i + \frac{1}{2}h, y(x_i) + \frac{1}{2}hk_1) \\
k_3 &= f(x_i + \frac{1}{2}h, y(x_i) + \frac{1}{2}hk_2) \\
k_4 &= f(x_i + h, y(x_i) + hk_3)
\end{aligned}
$$

```
function [x, y] = runge_kutta(fce, interval, y0, n)

a = interval(1);
b = interval(2);

% vytvoreni vektoru nezavisle promenne
x = linspace(a, b, n+1)';
h = (b-a)/n;

% vytvoreni prazdneho vektoru reseni
y = zeros(n+1, length(y0));

y(1,:) = y0;
for i = 1 : (length(x) - 1)
    % odhady derivaci resene soustavy
    k_1 = fce(x(i), y(i,:))';
    k_2 = fce(x(i) + 1/2*h, y(i,:) + 1/2*h*k_1)';
    k_3 = fce(x(i) + 1/2*h, y(i,:) + 1/2*h*k_2)';
    k_4 = fce(x(i) + h, y(i, :) + k_3*h)';

    % vypocet reseni v novem kroku
    y(i+1,:) = y(i,:) + h*(k_1 + 2*k_2 + 2*k_3 + k_4)/6;
end
end
```