Numerické metody, diferenciální rovnice, Modelica

October 29, 2014

Úlohy numerické matematiky

úlohy numericky formulované

- úlohy lineární algebry
  - řešení soustav lineárních rovnic
  - hledání vlastních vektorů a čísel matic

úlohy převáděné na numerické úlohy

- aproximace funkcí
- řešení nelineárních rovnic
- hledání extrémů funkcí
- úlohy matematické analýzy
  - numerické derivace, integrace
  - řešení obyčejných diferenciálních rovnic (ODE)
  - řešení parciálních diferenciálních rovnic (PDE)

Pokud lze řešit analyticky – je to lepší (vyjímky).

Chyby

Numerické řešení – přibližné
Zdroje chyb

- chyba vstupních dat
- zaokrouhlovací chyba – výpočet v aritemetice s konečnou přesností
- chyba metody – numerická metoda řeší matematickou úlohu přibližně

x .. přesná hodnota, $\tilde{x}$.. přibližná (spočítaná) hodnota.
Absolutní chyba

$$A(x) = |x - \tilde{x}|$$

Relativní chyba

$$R(x) = \frac{|x - \tilde{x}|}{|x|} = \frac{A(x)}{|x|}$$

Sledujeme hlavně relativní chybu

Chyba rozdílu podobných čísel

$x_1 \cong x_2$, $R(x_1)$ a $R(x_2)$ malá

$$R(x_1 - x_2) = \frac{|x_1 - x_2 - (\tilde{x}_1 - \tilde{x}_2)|}{|x_1 - x_2|} = \frac{|x_1 - \tilde{x}_1 - (x_2 - \tilde{x}_2)|}{|x_1 - x_2|}$$

absolutní chyba zůstává statisticky podobná (rozdíl dvou náhodných veličin), jmenovatel malý $\Rightarrow$

! **relativní chyba veliká** – problém při Gaussově eliminaci

násobení a dělení jsou bezpečné

Metody pro ODE – Euler

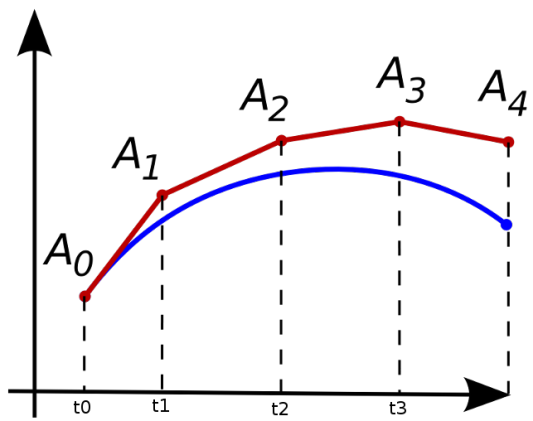
$$\dot{x} = f(x, t)$$

Diskretizace proměnných a rovnice (náhrada derivace diferencí)

$$\frac{x_{n+1} - x_n}{h} = f(x_n, t_n)$$

h .. časový krok
Počítáme iterativně

$$x_{n+1} = x_n + h \cdot f(x_n, t_n)$$



Řád přesnosti metody

- požadavky na výpočet – přesnost, rychlost – jde proti sobě, krátký krok = přesnější výsledek, více iterací

řád přesnosti α (značíme $O(h^\alpha)$): chyba řešení $R(x) \sim h^\alpha$

- Čím vyšší řád přesnosti, tím lepší: např. pro $O(h^3)$ – zkrátíme krok na $1/2$, chyba se zmenší na $1/8$

Odvození řádu přesnosti Eulera

Lokální chyba (chyba jednoho kroku):

$$\text{Taylor: } x(t+h) = x(t) + h \cdot \dot{x}(t) + \frac{1}{2}h^2\ddot{x}(t) + \ldots$$

$$\text{Euler: } x_{n+1} = x_n + h \cdot f(x_n, t_n) = x_n + h \cdot \dot{x}$$

$$\text{Rozdíll: } x(t+h) - x_{n+1} = R(x_{n+1}) = \frac{1}{2}h^2\ddot{x}(t) + \ldots$$

Lokální chyba je $R(x_{n+1}) \sim h^2$.
Globální chyba $R(x) \sim R(x_{n+1}) \cdot N$
Počet kroků $N \sim \frac{1}{h}$ – musím vynásobit
$R(x) \sim h^2 \cdot \frac{1}{h} = h$
Euler je prvního řádu přesnosti – $O(h^1)$.

Numerický výpočet řádu přesnosti

- znám přesné řešení $x(T)$
- počítám opakovaně numerické řešení x_N – zvětšuji počet kroků na dvojnásobek (tzn. zmenšuji krok)
- sleduji snižování chyby

i	kroků	přesné	numerické	chyba R_i	$\frac{R_i}{R_{i-1}}$	řád $= log_2\left(\frac{R_{i-1}}{R_i}\right)$
1	10	1	4.127	3.127	-	-
2	20	1	2.515	1.515	0.484	1.045
3	40	1	1.63	0.63	0.415	1.265
4	80	1	1.279	0.279	0.442	1.175
5	160	1	1.131	0.131	0.469	1.090
6	320	1	1.0636	0.0636	0.485	1.042

- pokud metoda nekonverguje (dostatečně rychle) – nejspíš někde chyba
- pokud neznám přesné řešení – spočítám s velmi krátkým krokem - sleduji jak výpočet konverguje k tomuto řešení

Runge-Kutta

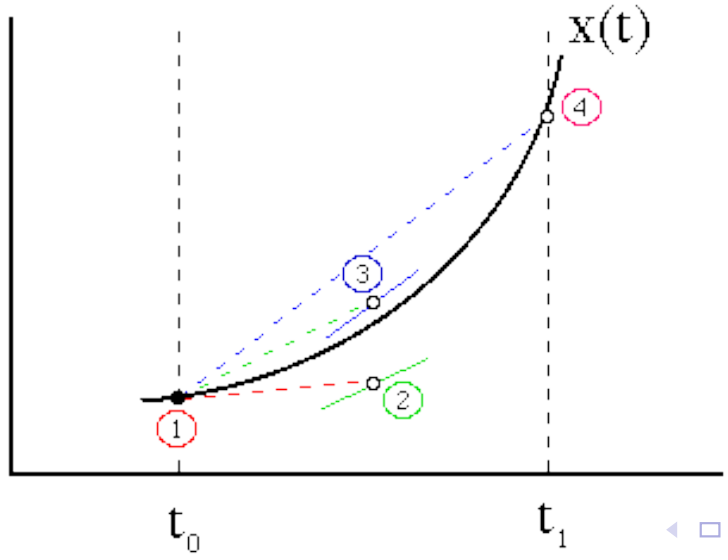
- třída metod
- nejpoužívanější RK-4, $O(h^4)$:

$$\dot{x} = f(x,t)$$

$$k_1 = f(t_n, x_n) \qquad k_2 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2) \qquad k_4 = f(t_n + h, x_n + hk_3)$$

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$



Numerická stabilita, podmíněnost úlohy, stiff rovnice

- stabilita
  - nestabilní metoda – chyby v jednotlivých krocích se akumulují tak, že dojde k naprosté ztrátě přesnosti
  - stabilní – chyba roste s počtem kroků maximální lineárně
  - obor stability lze pro konkrétní rovnice a metody odvodit $\rightarrow$omezení časového kroku
- podmíněnost – podíl relativní změny řešení ku relativní změně vstupních dat (motýlí efekt)
  - špatně podmíněná $\Rightarrow$ malá chyba výpočtu výrazně ovlivní pozdější výsledek $\Rightarrow$ potřeba počítat přesně
- stiff rovnice – přestože metoda by měla být stabilní, musíme volit velmi krátký krok
  - specielní robustní metody

Implicitní metody

např. implicitní Euler

$$\dot{x} = f(x, t)$$

$$\frac{x_{n+1} - x_n}{h} = f(x_{n+1}, t_{n+1})$$

x_{n+1} není možné z rovnice vyjádřit, v každém kroce řešíme (nelineární) rovnici

- Implicitní metody bývají robustní (stabilní), vhodné pro stiff rovnice
- radau - implicitní runge-kutta

Soustava diferenciálních a algebraických rovnic – DAE

Diferenciální rovnice

$$\bar{F}(\bar{x}, \dot{\bar{x}}, \bar{y}, t) = \bar{0}$$

nebo explicitně

$$\dot{\bar{x}} = f(\bar{x}, \bar{y}, t)$$

a algebraické rovnice

$$\bar{G}(\bar{x}, \bar{y}, t) = \bar{0}$$

$\bar{\mathbf{x}}$ .. stavové proměnné – jsou derivované

- stejný počet stavových proměnných jako diferenciálních rovnic

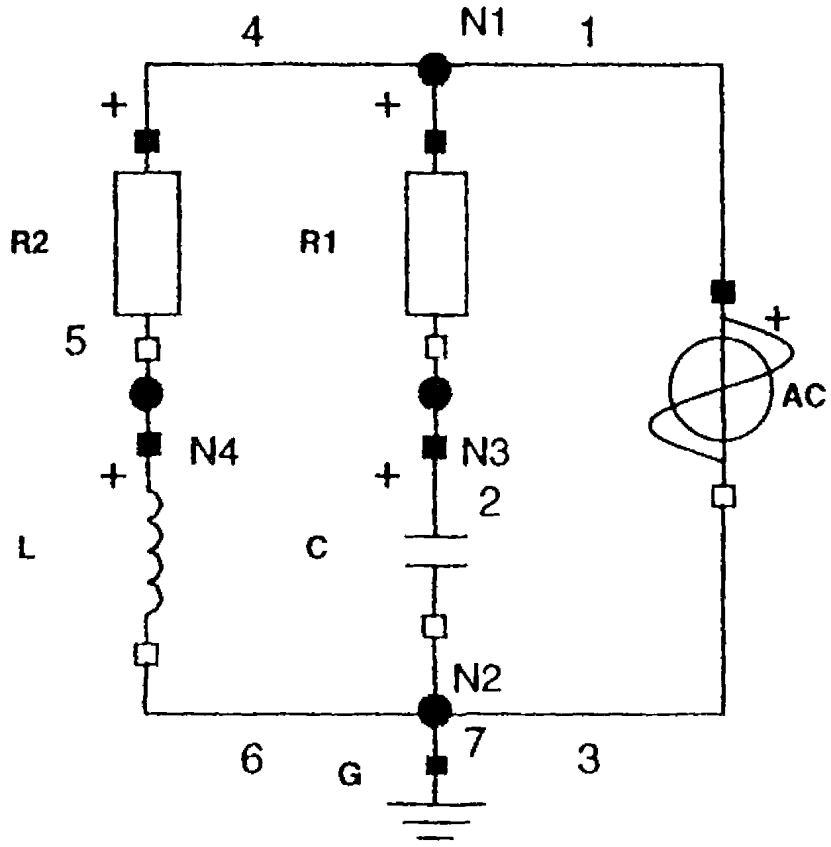
$\dot{\bar{\mathbf{x}}}$ .. derivace stavových proměnných

$\bar{\mathbf{y}}$ .. algebraické proměnné

- stejný počet algebraických proměnných jako algebraických rovnic

Příklad – schéma

RLC obvod:



Řešenív Modelice – seřazení rovnic

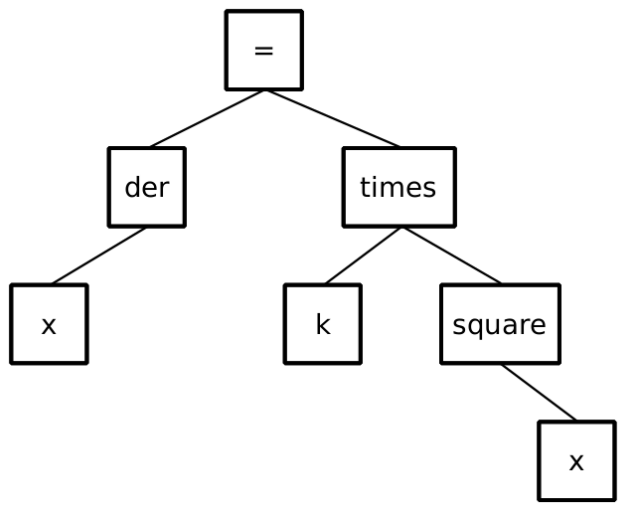
$$der(L.i) * L.L - L.v = 0 \quad 7$$
$$der(C.v) * C.C - C.i = 0 \quad 6$$
$$C.v - R1.p.v + R1.v = 0 \quad 4$$
$$C.i - R1.v / R1.R = 0 \quad 5$$
$$u(time) - R1.p.v = 0 \quad 2$$
$$R1.p.v - R2.v - L.v = 0 \quad 3$$
$$R2.v - R2.R * L.i = 0 \quad 1$$

- parametry, vstupy
- stavy
- algebraické
- derivace

soustava rovnic – strong component – v každé iteraci numericky

Překlad modelu

1. Flatenizace modelu – hierarchie objektů nahrazena jedinou soustavou rovnic (viz. předchozí příklad)
2. Vytvoření AST (abstract syntax tree) modelu (parsování textových rovnic)
3. Analyzer – algebraické manipulace – zjednodušení rovnic, BLT transformace
4. Generování kódu v C (C#), slinkování se solverem $\rightarrow$ simulace



$$\dot{x} = kx^2$$

Hybridní systémy - podmíněné výrazy a rovnice

Volný pád ve vzduchu a vodě. Vodní hladina $x = 0$:

$$
\begin{aligned}
inAir &= x > 0 \\
f &= -gm - (\text{if } inAir \text{ then } k_1 v \text{ else } k_2 |v| v) \\
\dot{v} &= f/m \\
\dot{x} &= v
\end{aligned}
$$

- Zero-crossing funkce $g(t) = x$
- Continuous time
  - řeší se diferenciální a algebraické rovnice a postupuje se v čase
  - nenastávají události, jsme jen v jedné větvi podmínky
  - sleduje se, jestli g nemění znaménko, událost $\rightarrow$přepnutí na
- discrete time – zpracovává se událost – čase se nemění
  - přesná detekce času události
  - řeší se dohromady spojité i diskrétní rovnice
    - ★ mezi diskrétnímy proměnnými mohou být složité vztahy (např. několik diod), událost může vyvolat další
    - ★ hledají se hodnoty diskrétních a algebraických (mohou být nespojité) proměnných vyhovující rovnicím

Výpočet DAE systému – rekapitulace

- Při překladu – tzn. jen jednou
  - Algebraické úpravy rovnic a BLT transformace
- V každé iteraci výpočtu
  - vyhodnocování rovnic – $\bar{x} \rightarrow \dot{\bar{x}}$ – numerické řešení strong-component
    - ⋆ strong-componenty mohou být nelineární, veliké (pro nelineární $N \gtrsim 10$) $\Rightarrow$**může být časově velice náročné** $\Rightarrow$u velkých modelů **potřeba strongcomponenty minimalizovat**
  - výpočet nových stavů
  - detekce zero-crossingů
  - případně zpracování události

Inicializace

Před simulací inicializace - přiřazují se hodnota všem proměnným (stavovým, derivacím, algebraickým, parametrům).
atribut `start`

- počáteční hodnota proměnné

atribut `fixed`

- `true` – počáteční hodnota proměnné je neměnná
- `false` – počáteční hodnota je jen odhad, proměnná vystupuje v inicializaci jako neznámá
- defaultně: stavy, algebraické (,derivace) - `false`, parametry - `true`

sekce `initialEquation`

- rovnice, které mají být při inicializaci navíc splněny společně s rovnicemi modelu

Inicializace 2

Řeší se dohromady rovnice modelu a iniciální rovnice

$$F(x, \dot{x}, y, t) = 0$$
$$G(x, y, t) = 0$$
$$I(x, \dot{x}, y, t) = 0$$

Počet všech rovnic by měl odpovídat počtu „nonFixed" proměnných(+derivací). Pokud rovnic méně, nastaví se některé proměnné na fixed.
Pokud neni nastaven `start` – defaultně 0
pro počáteční hodnoty – odvozeny výrazy, inicializace – vyhodnocení (někdy část numericky)
Pokud chci zadat počáteční hodnotu, je lepší použít `start = ..., fixed = true` **než používa pro přiřazení** `initial equation`.

Inicializace - příklad

```
model InitExample
  Real x(start = 1, fixed = true);
  Real y(start = 2); //fixed = false by default
initial equation
  der(y) = 0;
equation
  der(x) = x + y;
  der(y) = x - y;
end InitExample
```

$$r_1 = \dot{y}$$
$$r_2 = \dot{x} - x - y$$
$$r_3 = \dot{y} - x + y$$

x je zadána napevno, y počáteční odhad
řeší se 3 rovnice pro 3 neznámé neznámé $y, \dot{x}, \dot{y}$. Stejně rovnic jako neznámých.

Materiály

- Specifikace Modeliky - www.modelica.org/documents
- Principles of Object-oriented modeling and simulation with Modelica (Peter Fritzson)
- články
  - Event Handling in the OpenModelica Compiler and Runtime System (Håkan Lundvall, Peter Fritzson, Bernhard Bachmann)
  - Dynamic Selection of States in Dymola (Sven Erik Mattsson, Hans Olsson and Hilding Elmqvist)
- Přeložené modely, zdrojový kód OpenModeliky