Modelica a numerické metody

30. listopadu 2016

Obyčejná diferenciální rovnice (ODE)

Algebraická rovnice – neznámá resp. řešení číslo

$$4 = x^2$$

Diferenciální rovnice – vztah mezi funkcí a její derivací, neznámá funkce
Explicitně:

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

Implicitně:

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

F nesingulární jakobián vzhledem k $\dot{x}$.
Pro systém – počet proměnných = počet rovnic.

Příklad – loď, počáteční podmínky

Setrvačný pohyb lodi ve vodě

$$\dot{v} = -\frac{1}{m} k v^2$$

řešení

$$v = \frac{m}{kt + c}$$

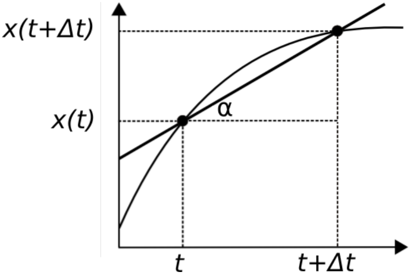
Systém řešení závisí na integrační konstantě c

- potřebujeme znát počáteční rychlost.

Derivace

Fyzikálně – rychlost změny původní veličiny (rychlost/poloha, zrychlení/rychlost, proud/náboj, tepelný tok/teplo)
Definice

$$\dot{x} = \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t}$$



Numerické řešení explicitní ODE - Eulerova metoda

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

Diskretizace proměnných

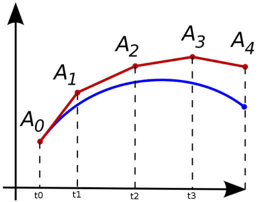
$$(t,x) \rightarrow (t_n, x_n), \quad n \in N \qquad\qquad \Delta t_n = t_{n+1} - t_n$$

Diskretizace rovnice (náhrada derivace diferencí)

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

x_0 známe z PP→počítáme iterativně:

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



DAE

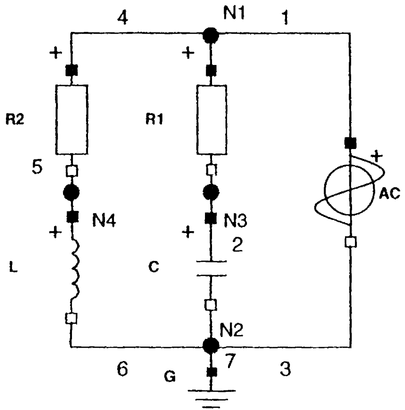
$$
\begin{aligned}
F(x, \dot{x}, y, t) &= 0 \\
G(x, y, t) &= 0
\end{aligned}
$$

F nesingulární jakobián vzhledem k $\dot{x}$.

- $F(x, \dot{x}, y, t)$ .. diferenciální rovnice
- $G(x, y, t)$ .. algebraické rovnice
- x .. stavové proměné (jsou v rovnicích derivovány)
- y .. algebraické proměnné (derivovány nejsou)
- Počet stavových proměnných = počet diferenciálních rovnic
- Počet algebraickýc proměnných = počet algebraických rovnic

Stav systému je určen vektorem stavových proměnných. Pro dané x a t je možné z rovnic určit y i $\dot{x}$.

Překlad modelu – příklad – RLC



RLC – textově – objekty

```
model rlc
  Modelica.Electrical.Analog.Basic.Inductor inductor1(L = 3);
  Modelica.Electrical.Analog.Basic.Capacitor capacitor1(C = 4);
  ...
equation
  connect(inductor1.n, capacitor1.n);
  connect(capacitor1.n, sineVoltage1.n);
  ...
end rlc;
```

```
model Inductor "Ideal linear electrical inductor"
  extends Interfaces.OnePort(i(start = 0));
  parameter SI.Inductance L(start = 1) "Inductance";
equation
  L * der(i) = v;
end Inductor;
```

Front-end, flat model

při překladu – flatenizace – odstranění tříd, odstranění jednotek ... $\rightarrow$ vše v jednom modelu, jen základní typy a rovnice.

$$
\begin{aligned}
der(L.i) * L.L - L.v &= 0\\
der(C.v) * C.C - C.i &= 0\\
C.v - R1.p.v + R1.v &= 0\\
C.i - R1.v/R1.R &= 0\\
u_0 \cdot \sin(f \cdot time) - R1.p.v &= 0\\
R1.p.v - R2.v - L.v &= 0\\
R2.v - R2.R * L.i &= 0
\end{aligned}
$$

- Známe: parametry, vstupy, stavy
- Potřebuji spočítat: algebraické (abych mohl spočítat derivace), derivace (abych mohl udělat časový krok - spočítat změnu stavů)

Back-end, BLT transformace

potřebujeme spočíst ze stavů derivace
proměnné spočítat pokud možno postupným vyhodnocováním rovnic

	$der(L.i)$	$der(C.v)$	$L.v$	$C.i$	$R1.p.v$	$R1.v$	$R2.v$
$der(L.i) * L.L - L.v = 0$	**1**	0	**1**	0	0	0	0
$der(C.v) * C.C - C.i = 0$	0	**1**	0	**1**	0	0	0
$C.v - R1.p.v + R1.v = 0$	0	0	0	0	**1**	**1**	0
$C.i - R1.v / R1.R = 0$	0	0	0	**1**	0	**1**	0
$u_0 \cdot \sin(f \cdot time) - R1.p.v = 0$	0	0	0	0	**1**	0	0
$R1.p.v - R2.v - L.v = 0$	0	0	**1**	0	**1**	0	**1**
$R2.v - R2.R * L.i = 0$	0	0	0	0	0	0	**1**

- Incidenční matice - jednička, pokud rovnice obsahuje proměnnou
- Prohazováním řádků a sloupců se snažíme převést na dolní trojúhelníkovou - BLT (Block Lower Triangular) transformace

Transformovaná incidenční matice

	$R2.v$	$R1.p.v$	$L.v$	$R1.v$	$C.i$	$der(L.i)$	$der(C.v)$
$R2.v - R2.R * L.i = 0$	1	0	0	0	0	0	0
$u_0 \cdot \sin(f \cdot time) - R1.p.v = 0$	0	1	0	0	0	0	0
$R1.p.v - R2.v - L.v = 0$	1	1	1	0	0	0	0
$C.v - R1.p.v + R1.v = 0$	0	1	0	1	0	0	0
$C.i - R1.v / R1.R = 0$	0	0	0	1	1	0	0
$der(L.i) * L.L - L.v = 0$	0	0	1	0	0	1	0
$der(C.v) * C.C - C.i = 0$	0	0	0	0	1	0	1

- Vzniká kauzalita:
  - Rovnice vyhodnocujeme odshora dolů
  - Proměnné vlevo od diagonály známe z vyhodnocení předchozích rovnic - vstupy
  - Proměnná na diagonále je z rovnice určena - výstup

Strong-componenty

Převod matice na LT není vždy možný, např. (jen algebraický systém)

$$\begin{aligned} d + c + b + 1 &= 0 \\ -c + b - 2 &= 0 \\ d - 1 &= 0 \\ d + a + 3 &= 0 \end{aligned}$$

převedu na

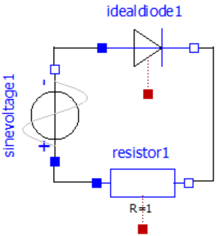
$$\begin{array}{lcr} d & = & 1 \\ d + c + b & = & -1 \\ \quad - c + b & = & 2 \\ d \qquad\quad + a & = & -3 \end{array}$$

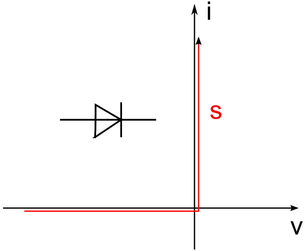
c a b tvoří systém rovnic, který není možný rozřešit prohazováním řádků a sloupců - *strong-component* - řeší se numericky.
BLT transformace - Tarjánův algoritmus z teorie grafů, vrcholy - neznámé, orientované hrany - závislosti mezi proměnnými. Snaha vytvořit co nejmenší strongcomponenty.

Úpravy rovnic, generování kódu, simulační runtime

- další úpravy a zjednodušení rovnic
- generování kódu (templatky) $\rightarrow$model v C (C#, JavaScript, ...)
  - např. funODE() $\sim \dot{x} = f(x, t)$
- slinkování se simulačním runtimem $\rightarrow$ spustitelná aplikace $\rightarrow$ simulace
  - numerické metody – časová integrace (Euler a další přesnější metody)

Hybridní systémy - podmíněné výrazy a rovnice





$$
\begin{aligned}
u_d + u_R + u_z(t) &= 0 \\
i\,R &= u_R \\
u_d &= \textbf{if}\ \mathit{off}\ \textbf{then}\ s\ \textbf{else}\ 0 \\
\textbf{if}\ \mathit{off}\ \textbf{then}\ i &= 0\ \textbf{else}\ i = s \\
\mathit{off} &= s < 0
\end{aligned}
$$

off - diskrétní proměnná, mění se jen při změně nerovnosti $s < 0$ (událost). Pro detekci události definuje *zero-crossing* funkci $g(s) = s$.

Hybridní systémy - continuous time

- V průběhu continuous time nenastávají události - diskrétní proměnné se nemění, např. pro $\mathit{off} = \mathit{true}$
- Řešíme rovnice, jako DAE. Nerovnice, nevyhodnocujeme.
- V podmínkách je jasné, která větev platí.

$$\begin{aligned} u_d + u_R + u_z(t) &= 0 \\ i\,R &= u_R \\ u_d &= s \\ i &= 0 \end{aligned}$$

- **Sledujeme zero-crossing funkci g $(= s)$. Když změní znaménko z** *true* **na** *false* - **událost. Přepneme na režim *discrete time.***

Hybridní systémy - discrete time

- Najde se přesný čas události. Dále se čas nemění.
- Zpracovává se událost. Je potřeba najít nové hodnoty všech diskrétních a algebraických proměnných (stavy zůstávají)
  - změna jedné diskrtétní proměnné může způsobit nespojitou změnu nějaké algebraické proměnné
  - ta může způsobit změnu další diskrétní proměnné - další událost
  - $\Rightarrow$řeší se celý systém rovnic včetně nerovnic. Nerovnice jsou zahrnuty do BLT transformace. Nerovnice mohou být součástí strongcomponent, diskrétní proměnné jsou pak ve s.c. neznámé.
- Po nalezení řešení $\rightarrow$continuous time
  - na začátek reset solveru (začíná se velmi krátkým krokem, postupně se prodlužuje) kvůli možným nespojitostem (proto událost drahá)
  - **expr - výraz s nerovnicemi - pokud víme, že nevznikne nespojitost a další události - noEvent(expr) - pak není vyvolána událost - zrychlí výpočet**

Ukázat přeložený model diodeExample.mo.

When

```
model BouncingBall
  parameter Real g=9.81 "gravity acceleration";
  parameter Real c=0.7 "coefficient of restitution";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";
equation
  der(h) = v;
  der(v) = -g;
  when {h <= 0.0} then
    reinit(v, -c*pre(v_new));
  end when;
end BouncingBall;
```

`when` vyvolá událost - rovnice ve `when` platí pouze během discrete time

`reinit`

`pre`

Shrnutí – překlad a výpočet

Překlad

- Flatenizace – odstranění tříd
- BLT transformace - analytická úprava systému rovnic
- generování kódu

Výpočet – v každém kroce:

- vyhodnocení rovnic - stavy $\rightarrow$algebraické $\rightarrow$derivace
  - vyřešení strong-component
  - detekce událostí
- časový krok
- případně zpracování události

strong-componenty numericky řešeny v **každém kroce** simulace, mohou být nelineární

- výpočetně náročné, potřeba strongcomponenty minimalizovat – info ve výpisu při překladu

Inicializace

Před simulací probíhá inicializace - přiřazuje se hodnota všem proměnným (stavovým, derivacím, algebraickým, parametrům).

- Pomocí `start` a `fixed`, napr. :

  ```
  Real x(start = 1, fixed = true);
  ```

  atribut `start` – počáteční hodnota, atribut `fixed`:
  - `true` - hodnota proměnné je neměnná
  - `false` - proměnná vystupuje v inicializaci jako neznámá, start hodnota je počáteční odhad
  - defaultně: parametry - `true`; stavy, algebraické (,derivace) - `false`
- 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 chybí start hodnoty, defaultně 0.
Rovnice jsou při překladu řešeny analyticky.
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
  Real z;
initial equation
  der(y) = 0;
equation
  der(x) = x + y;
  der(y) = x - y);
  z = 2*x;
end InitExample
```

2 stavové ($\Rightarrow$2 derivace), 1 algebraická $\Rightarrow$ potřebujeme 5 vazeb
1 start-fixed-true, 1 initial eq, 3 eq $\Rightarrow$ OK

Index DAE systému

Někdy není možné vyjádřit derivace výše popsaným způsobem, např.

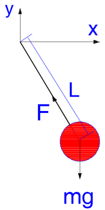
$$\begin{aligned} 2x + \dot{x} + y + \dot{y} &= 1 \\ x + 2y &= 1 \end{aligned}$$

1 diferenciální rovnice, 1 algebraická rovnice, **2 stavové proměnné ?! Ne zvolíme jen jednu**, např. y
Zderivujeme algebraickou rovnici: $\dot{x} + 2\dot{y} = 0$, vyjádříme $\dot{x} = -2\dot{y}$
a $\dot{x}$ z rovnic eliminujeme

$$\begin{aligned} 2x + y - \dot{y} &= 1 \\ x + 2y &= 1 \end{aligned}$$

dále známý postup pro DAE $\rightarrow$ vyjádření $\dot{x} = f(x,t)$.
Minimální počet, kolikrát musíme derivovat systém rovnic (nebo jeho část), abychom mohli vyjádřit $\dot{x} = f(x,t)$ nazýváme *index systému*. Čím vyšší index, tím větší problémy při řešení.

Dynamic state selection - rovinné kyvadlo



$$m\ddot{x} = -\frac{x}{L}F$$
$$m\ddot{y} = -\frac{y}{L}F - mg$$
$$x^2 + y^2 = L^2$$

Derivace nejdou spočítat – nani jak spočítat $F \Rightarrow$ derivujeme, dosazujeme …

Systém s vyšším indexem. Stavová proměnná je jen jedna - stav je určen jen jednou proměnnou - 1 stupeň volnosti (vlastně 2, ještě derivace). Zvolíme za stav např. x. Co se stane, když přejdeme do vodorovné polohy? x tady nemůže být stavem $\Rightarrow$dynamic state selection - přepneme na y.

Blokově orientovaný vs. rovnicový

nebo signálový a akauzální

Blokově orientovaný - kauzalita je přímo v modelu.

Rovnicový - pracuje se s rovnicemi, kauzalita se vytváří automaticky až při překladu (BLT).

$$M\ddot{x} = F$$

$F(t)$ je vstup, počítáme $x(t)$:

v Modelice

$$\begin{aligned} M\,\mathrm{der}(v) &= F(t) \\ v &= \mathrm{der}(x) \end{aligned}$$

v Simulinku

$$\boxed{F(t)} \to \boxed{\frac{\cdot}{M}} \to \boxed{\int \cdot} \to \boxed{\int \cdot} \to \boxed{x}$$

Příklad s robotem

- robot
- má se pohybovat tělesempředepsaným způsobem
- chceme spočítat sílu
- tzn. máme zadané $x(t)$, chceme spočítat $F(t)$ - záměna vstupu a výstupu.

V Modelice snadné

$$
\begin{aligned}
M\,\mathrm{der}(v) &= F \\
v &= \mathrm{der}(x(t))
\end{aligned}
$$

V Simulinku potřeba předělat celý model - obrácený směr, místo integrátorů derivace atd).
Modelica blíže fyzikálnímu popisu.

Materiály

- Modelica by Example – http://book.xogeny.com/
- Principles of Object-oriented modeling and simulation with Modelica (Peter Fritzson)
- Specifikace Modeliky - www.modelica.org/documents
- 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