Modelica a numerické metody

28. listopadu 2012

Rozdělení úloh

Explicitní ODE

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

Implicitní ODE

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

DAE

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

Hybridní DAE

Explicitní ODE

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

$\bar{x}$ vektor stavových proměnných, $\dot{\bar{x}}$ vektor derivací stavů, t čas. Počet stavů musí být rovný počtu rovnic, tzn. délka vektorů $\bar{x}$ a $\bar{f}(\bar{x}, t)$ je stejná.
Počáteční podmínky

$$\bar{x}(0) = \bar{x}_o$$

Počáteční úloha $\rightarrow$ řešením je funkce $x(t)$.
Např.

$$\dot{v} = -kv, \qquad v(0) = 10$$

$$v(t) = 10e^{-kt}$$

Numerické řešení explicitní ODE - Eulerova metoda

Známe stavy →spočteme derivace $\dot{x}$ z $f(x,t)$→spočteme změnu stavů v kroce

Diskretizace proměnných

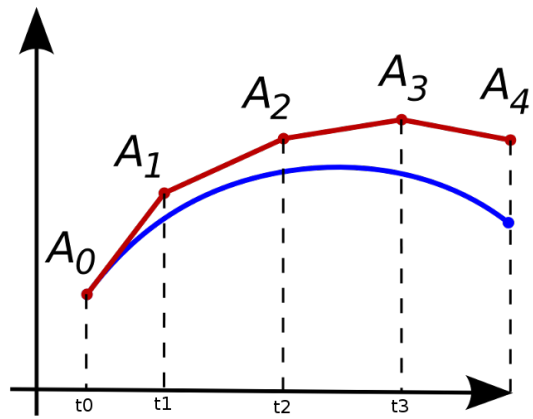
$$(t,x) \rightarrow (t_n, x_n), \quad n \in N \qquad\qquad \Delta t_n = t_{n+1} - t_n$$

Diskretizace rovnice (náhrada derivace diferencí)

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

Počítáme iterativně

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



Implicitní ODE

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

pro F platí předpoklady věty o implicitní funkci, hlavně jakobián

$$\frac{\partial F}{\partial \dot{x}} = \begin{pmatrix} \frac{\partial F_1}{\partial \dot{x}_1} & \cdots & \frac{\partial F_1}{\partial \dot{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial \dot{x}_1} & \cdots & \frac{\partial F_n}{\partial \dot{x}_n} \end{pmatrix}$$

je regulární $\Rightarrow$ pro dáné x a t můžeme řešením rovnice $F(x, \dot{x}, t) = 0$ spočítat neznámé $\dot{x}$. Kdy je singulární?
Příklad implicitní ODE:

$$\dot{x}x + 1 = 0$$

Převedení implicitní na explicitní snadné:

$$F(x, \dot{x}, t) := \dot{\bar{x}} - \bar{f}(\bar{x}, t)$$

DAE

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

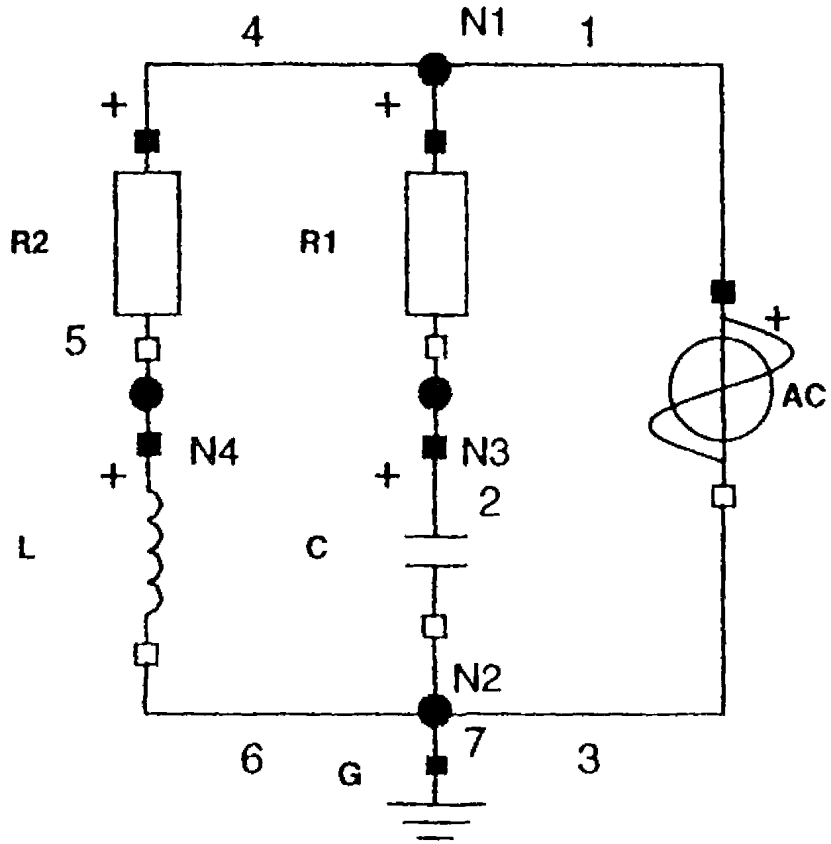
F nesingulární jakobián vzhledem k $\dot{x}$.

- $F(x, \dot{x}, y, t)$ .. 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 $\dot{x}$.
Převedení DAE na explicitní ODE - řešením rovnic $F(...) = 0$, $G(...) = 0$ pro neznámé y a $\dot{x}$ dostáváme $\dot{x} = f(x, t)$ (numericky).

Převedení DAE na explicitní ODE - příklad - schéma

RLC obvod:



Převedení DAE na explicitní ODE - příklad - rovnice

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

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

BLT transformace

	$der(L.i)$	$der(C.v)$	$L.v$	$C.i$	$R1.p.v$	$R1.v$	$R2.v$
$der(L.i) * L.L - L.v = 0$	**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
$u(time) - R1.p.v = 0$	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
$u(time) - R1.p.v = 0$	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

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

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.

Výpočet DAE modelu

Výpočet založen na převádění DAE na explicitní ODE popsaným způsobem.
ODE je pak řešena nějakou numerickou metodou (podobně jako Euler - ale přesnější a robustnější).

- BLT transformace - analytická úprava systému rovnic - provádí se jen jednou při překladu modelu
- strong-componenty numericky řešeny v **každém kroce** simulace (numericky - pro konkrétní hodnoty stavů a vstupů)
  - strong-componenty mohou být nelineární
  - veliké (pro nelineární $N \gtrsim 10$)
  - $\Rightarrow$výpočet strong-component **může být časově velice náročné** $\Rightarrow$u velkých modelů **potřeba strongcomponenty minimalizovat**
  - používá se Newtonova iterační metoda (metoda tečen) - počáteční odhad - interpolací hodnot v minunulých krocích

Index DAE systému

Někdy není možné vyjádřit derivace výše popsaným způsobem, např.

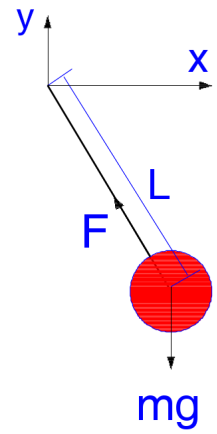
$$2x + \dot{x} + y + \dot{y} = 1$$
$$x + 2y = 1$$

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

$$2x + y - \dot{y} = 1$$
$$x + 2y = 1$$

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.

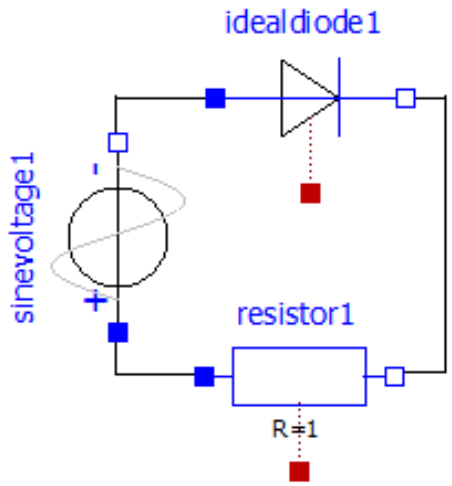
Dynamic state selection - rovinné kyvadlo

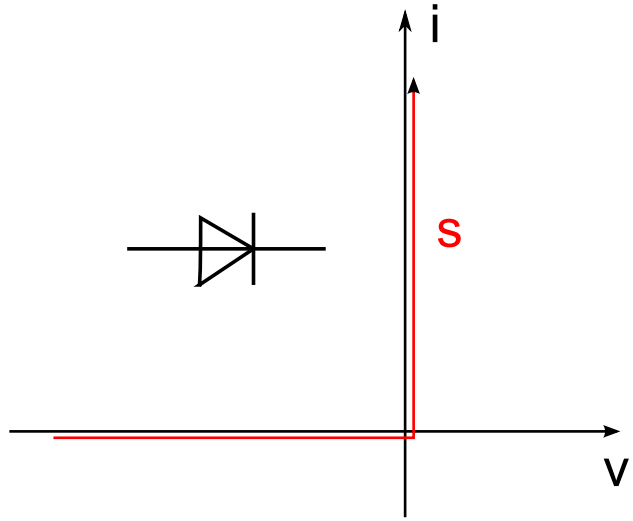


$$m\ddot{x} = -\frac{x}{L}F$$
$$m\ddot{y} = -\frac{y}{L}F - mg$$
$$x^2 + y^2 = L^2$$

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.

Hybridní systémy - podmíněné výrazy a rovnice





$$u_d + u_R + u_z(t) = 0$$
$$i\,R = u_R$$
$$u_d = \text{if } off \text{ then } s \text{ else } 0$$
$$\text{if } off \text{ then } i = 0 \text{ else } i = s$$
$$off = s < 0$$

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 $off = true$
- Řešíme rovnice, jako DAE. Nerovnice, nevyhodnocujeme.
- V podmínkách je jasné, která větev platí.

$$u_d + u_R + u_z(t) = 0$$
$$i\,R = u_R$$
$$u_d = s$$
$$i = 0$$

- **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 rovnice 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`

Inicializace

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

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

atribut `start`

- `fixed=true` - pevná počáteční hodnota
- `fixed=false` - počáteční odhad

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.
Pro všechny rovnice jsou vygenerovány residuální funkce

$$r(x, \dot{x}, y, t) = \begin{pmatrix} F(x, \dot{x}, y, t) \\ G(x, y, t) \\ I(x, \dot{x}, y, t) \end{pmatrix}$$

minimalizace čtverců residuálních rovnic - Simplexová metoda.
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
```

Reziduální funkce

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

x je zadána napevno, hledáme $y, \dot{x}, \dot{y}$. Stejně rovnic jako neznámých.

$$\min_{(y,\dot{x}\dot{y})}(\dot{y}^2 + (\dot{x} - x - y)^2 + (\dot{y} - x + y)^2)$$

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

$$M\,\mathrm{der}(v) = F(t)$$
$$v = \mathrm{der}(x)$$

v Simulinku

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

Příklad s robotem

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

V Modelice snadné

$$M\,\mathrm{der}(v) = F$$
$$v = \mathrm{der}(x(t))$$

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

O překladu a výpočtu

Překlad

- Zdrojový kód modeliky (file.mo)
- překladač flatenizuje model $\rightarrow$ flat model
- parser $\rightarrow$model AST reprezentaci
- analyzer zjednoduší a seřadí rovnice (BLT) $\rightarrow$kauzalita
- generování kódu (templatky) $\rightarrow$model v C nebo C# (funODE(), funDAE(), residualFunc() ...)

Simulace

- runtime zpracovává události
- volá solver (třeba Euler, nebo DASSL) pro výpočet *continuous time*

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