Simulation au niveau « système » : vers une approche acausale

Sébastien FURIC
Imagine

Sommaire

- Qu'est-ce que la modélisation au niveau « système » ?
- Exemples de modélisation
- Exploitation des modèles
- Cas de la simulation dynamique
  - L'approche causale
  - L'approche acausale

Définitions préliminaires (1)

- Qu'est-ce qu'un système ?
  - C'est un objet ou une collection d'objets dont nous voulons étudier des propriétés
- Qu'est-ce qu'un modèle ?
  - C'est une abstraction d'un système sur laquelle des expériences peuvent être menées (observation d'un phénomène)

Définitions préliminaires (2)

- Qu'est-ce qu'une représentation d'état ?
  - C'est une abstraction particulière d'un système dans laquelle on va représenter les contraintes (physiques, de commande) par des équations de « variables d'état » (discrètes et/ou continues)
  - C'est la forme priviligiée par des outils comme Scicos pour l'étude des propriétés dynamiques des modèles

Qu'est-ce que la modélisation (au niveau) « système » ?

- Une grande partie des phénomènes en jeu dans un modèle trop fidèle est négligeable (et coûte cher à prendre en compte)
- La modélisation « système » consiste à ne retenir dans les modèles que les « contributions significatives » de chaque partie d'un système (sous-système), pour chaque phénomène physique susceptible d'être observé
  - Création de « sous-modèles »

Exemple de modélisation (1)

```
connector Pin
  Real v;
  flow Real i;
end Pin;
```

```
partial model TwoPin
  Pin p, n;
  Real v, i;
equation
  v = p.v – n.v;
  i = p.i;
  i = -n.i;
end TwoPin;
```

```
model Resistor extends TwoPin;
  parameter Real R;
equation
  v = R * i;
end Resistor;
```

```
model Capacitor extends TwoPin;
  parameter Real C;
equation
  C * der(v) = i;
end Capacitor;
```

Exemple de modélisation (2)

```
model Switch1 extends TwoPin;
  constant Real RMIN = 1e-6;
  constant Real RMAX = 1e6;
  Boolean open;
equation
  if open then
    v = RMAX * i;
  else
    v = RMIN * i;
  end if;
end Switch1;

model Switch2 extends TwoPin;
  Boolean open;
equation
  if open then
    i = 0.0;
  else
    v = 0.0;
  end if;
end Switch2;
```

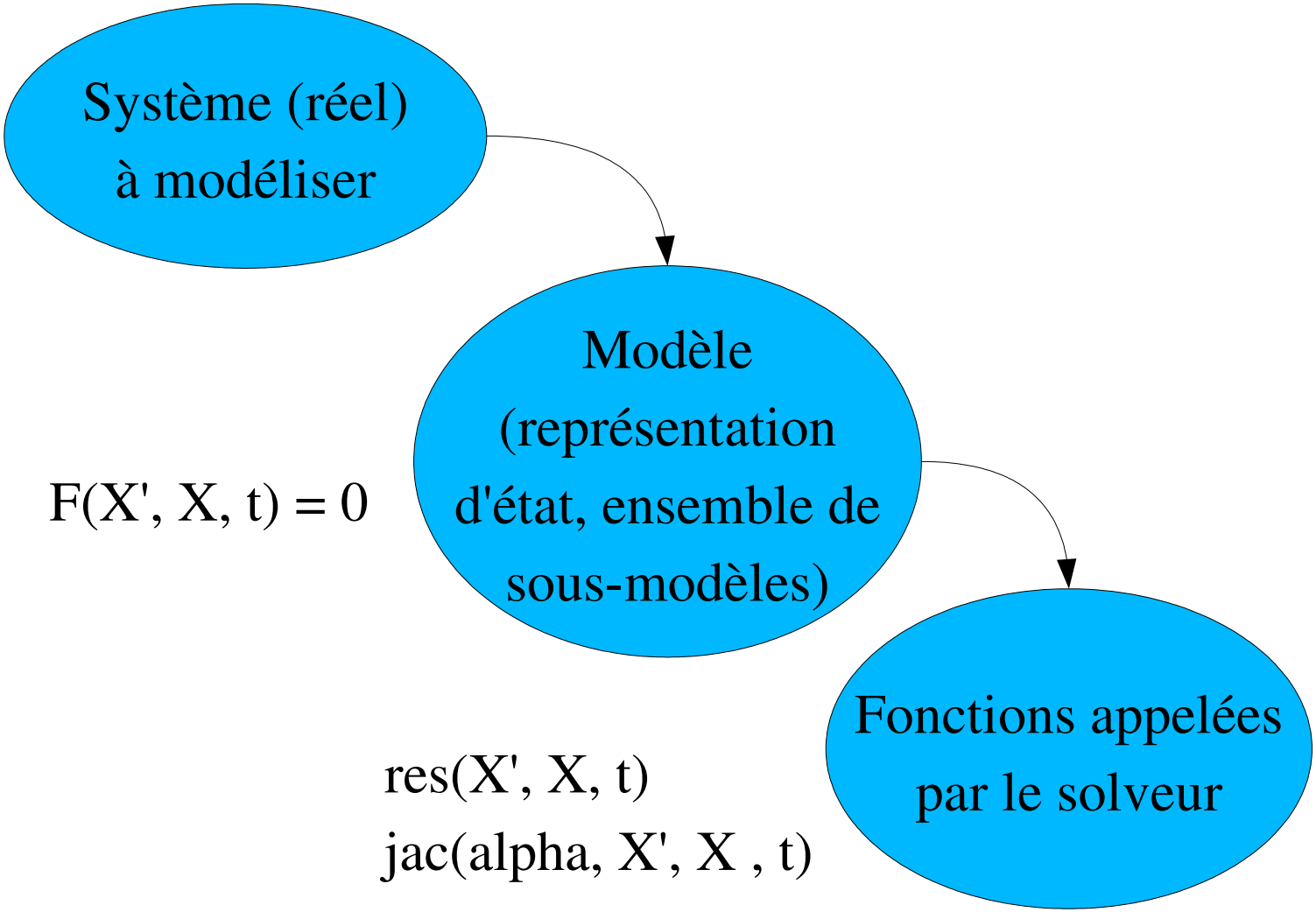
Exploitation des sous-modèles

- Plusieurs modes d'exploitation des sous-modèles sont possibles :
  - Simple extraction d'information
    - Pour l'assemblage graphique
  - Dimensionnement
  - Inversion
  - Simulation dynamique
  - Génération de code « temps réel »
    - Pour des simulations réalistes
    - Pour une cible embarquée

Cas de la simulation dynamique

- Nécessite la résolution numérique d'un système d'équations (représentation d'état) par un solveur (DASKR par exemple)
  - Une « mise en forme » du système d'équations doit être opérée car le solveur requiert des fonctions (C ou FORTRAN) lui fournissant des informations sur les équations pour affiner sa recherche de solution

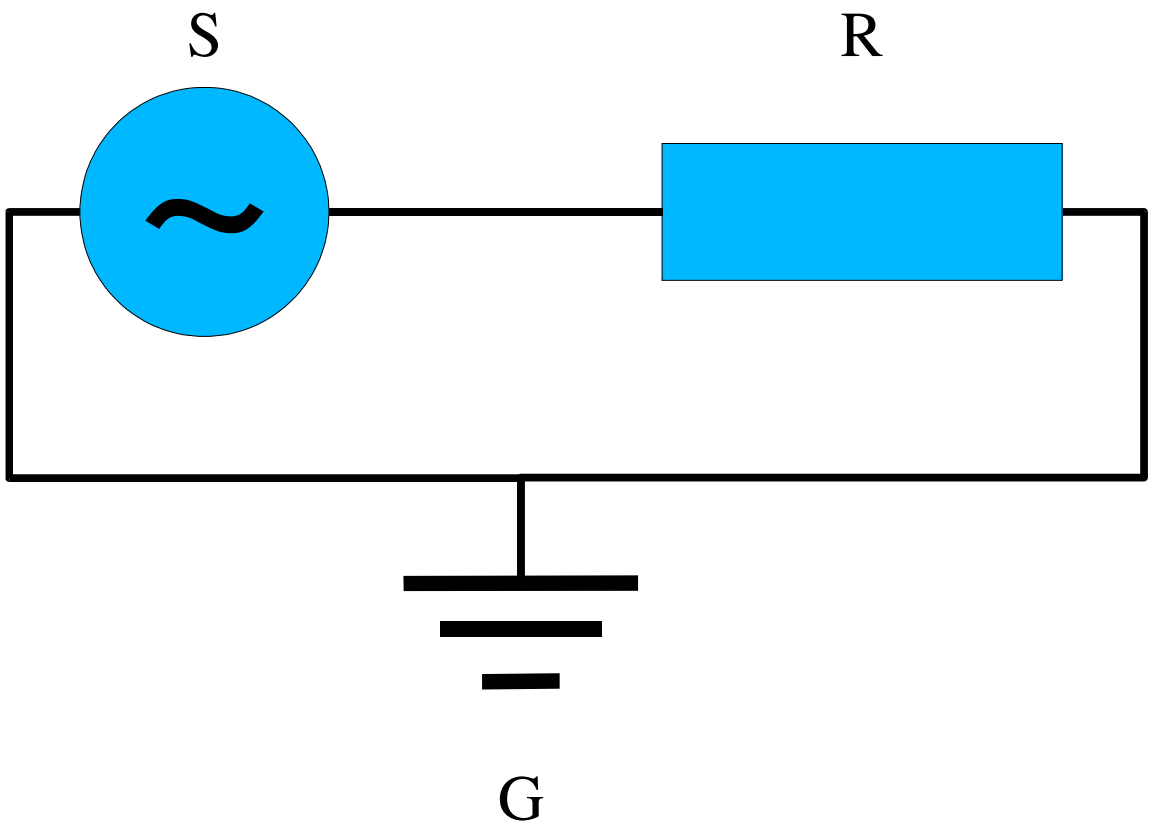
Niveaux d'abstraction



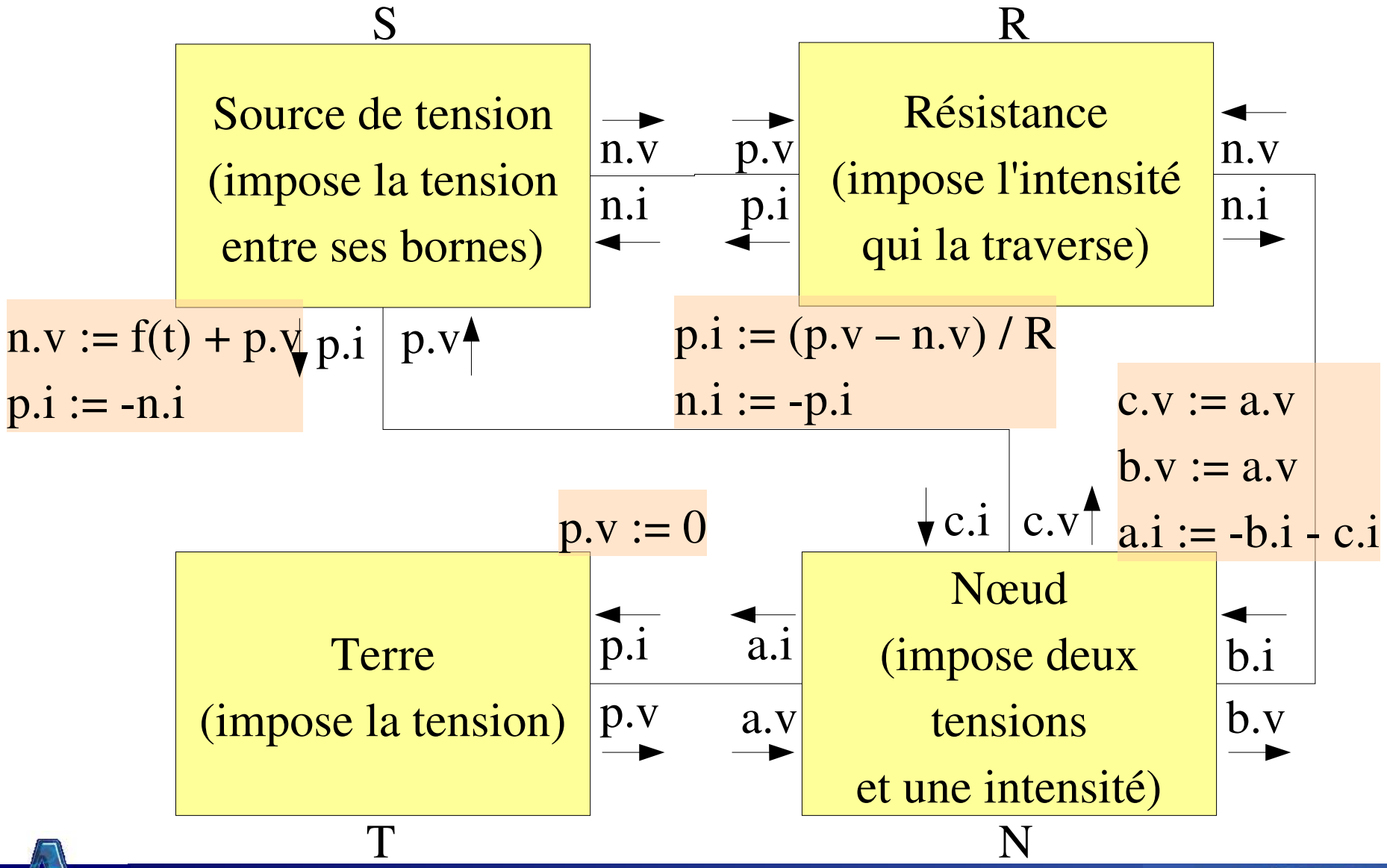
L'approche causale

- L'approche causale ou « orientée » consiste à construire explicitement le modèle en représentation d'état par assemblage de formes « pré-résolues » d'équations réparties dans les sous-modèles
- L'outil de modélisation peut assister l'utilisateur en lui proposant des orientations de modèles appropriées pour un assemblage de sous-modèles donnés

Exemple de système



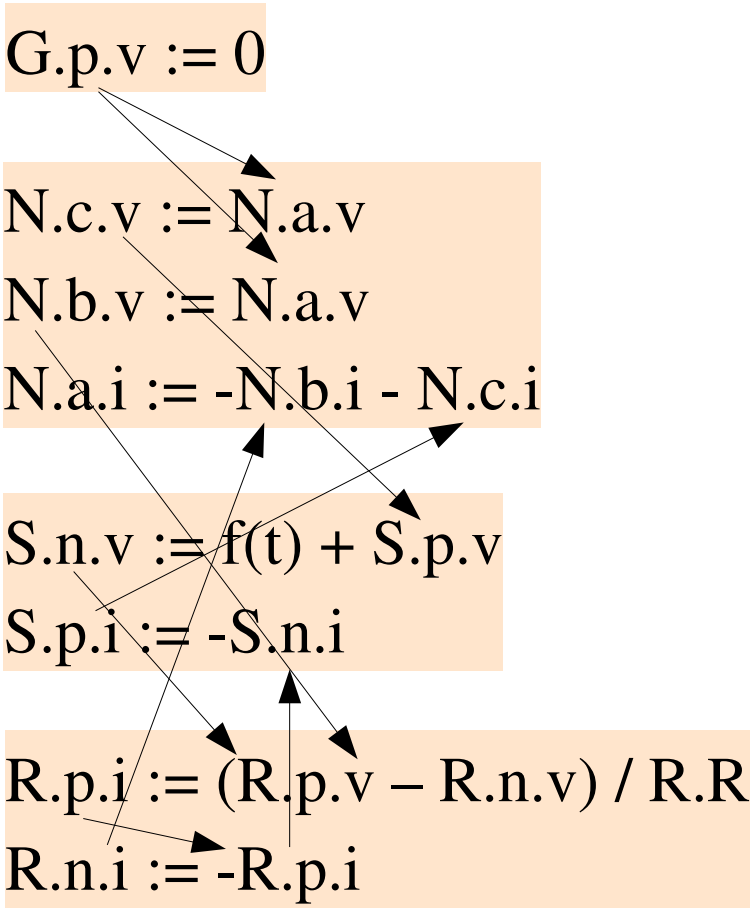
Exemple de modèle causal



Principe de la construction de l'abstraction finale

- Les affectations présentes dans les sous-modèles sont triées topologiquement suivant un ordre basé sur le fait qu'une affectation fournissant une entrée doit se trouver avant les affectations consommant cette entrée
- La relation d'ordre n'est pas totale ; il existe de plus des groupes non triables (« boucles algébriques »)
- Les boucles algébriques sont typiquement résolues par un solveur algébrique

Exemple



Propagation
des valeurs
entre groupes d'affectations

G.p.v := 0

N.a.v := G.p.v

N.c.v := N.a.v

N.b.v := N.a.v

S.p.v := N.c.v

S.n.v := f(t) + S.p.v

R.p.v := S.n.v

R.n.v := N.b.v

R.p.i := (R.p.v – R.n.v) / R.R

R.n.i := -R.p.i

S.n.i := R.p.i

S.p.i := -S.n.i

N.b.i := R.n.i

N.c.i := S.p.i

N.a.i := -N.b.i - N.c.i

Remarques

- L'assemblage des sous-modèles aboutit à une forme proche de ce que l'on doit fournir au solveur
- On suppose qu'un sous-modèle « calcule » toujours les mêmes variables au cours de la simulation, quel que soit l'environnement de ce sous-modèle
- La librairie de sous-modèles est extrêmement redondante car on doit construire toute une famille de sous-modèles orientés correspondant à un même système de contraintes

L'approche acausale

- L'approche acausale ou « non orientée » consiste à partir d'un niveau d'abstraction élevé dans lequel un sous-modèle contient un ensemble de contraintes (équations)
- L'assemblage de sous-modèles acausaux aboutit à un système d'équations (et non une série d'affectations) dont un outil de manipulation symbolique devra extraire une séquence calculable

Exemple

```
model Ground
  Pin p;
equation
  p.v = 0.0;
end Ground;

model Resistor extends TwoPin;
  parameter Real R;
equation
  v = R * i;
end Resistor;

model VoltageSource extends TwoPin;
  parameter Real VMax, f = 1.0;
equation
  v = VMax * sin(6.28 * f * time);
end VoltageSource;
```

```
model Circuit;
  VoltageSource S(VMax = 5.0);
  Resistor R(R=1000.0);
  Ground G;
equation
  connect(G.p, S.p);
  connect(G.p, R.n);
  connect(S.n, R.p);
end Circuit;
```

Construction du système d'équations

- Le système d'équations est construit, comme dans le cas causal, par réunion des « contenus » des sous-modèles
- Le résultat n'est pas directement « triable » contrairement au cas causal car il n'existe pas d'ordre préétabli pour le calcul des variables
  - Il faut « découvrir » les affectations
  - Il faut générer le code de ces affectations

Exemple (1)

```
model Circuit;
  VoltageSource S(VMax = 5.0);
  Resistor R(R=1000.0);
  Ground G;
equation
  connect(G.p, S.p);
  connect(G.p, R.n);
  connect(S.n, R.p);
end Circuit;
```

$S.v = S.p.v - S.n.v$

$S.i = S.p.i$

$S.i = -S.n.i$

$S.v = 5.0 * \sin(6.28 * time)$

$R.v = R.p.v - R.n.v$

$R.i = R.p.i$

$R.i = -R.n.i$

$R.v = 1000.0 * R.i$

$G.p.v = 0.0$

$G.p.i + S.p.i + R.n.i = 0.0$

$S.n.i + R.p.i = 0.0$

$G.p.v = S.p.v$

$G.p.v = R.n.v$

$S.n.v = R.p.v$

Exemple (2)

Résultat de l'analyse causale

```
S.p.i := S.i
S.i := -S.n.i
S.p.v := G.p.v
R.n.v := G.p.v
R.p.v := S.n.v
R.p.i := R.i
R.n.i := -R.i
G.p.v := 0.0
G.p.i := -S.p.i – R.n.i
S.n.i := -R.p.i
R.v := R.p.v - R.n.v
R.i := R.v / 1000.0
S.v := 5.0 * sin(6.28 * time)
S.n.v := S.p.v - S.v
```

tri →

```
G.p.v := 0.0
S.v := 5.0 * sin(6.28 * time)
S.p.v := G.p.v
R.n.v := G.p.v
S.n.v := S.p.v - S.v
R.p.v := S.n.v
R.v := R.p.v – R.n.v
R.i := R.v / 1000.0
R.p.i := R.i
R.n.i := -R.i
S.n.i := -R.p.i
S.i := -S.n.i
S.p.i := S.i
G.p.i := -S.p.i – R.n.i
```

Exemple (3)

Élimination des variables inutiles

```
G.p.v := 0.0
S.v := 5.0 * sin(6.28 * time)
S.p.v := G.p.v
R.n.v := G.p.v
S.n.v := S.p.v - S.v
R.p.v := S.n.v
R.v := R.p.v – R.n.v
R.i := R.v / 1000.0
R.p.i := R.i
R.n.i := -R.i
S.n.i := -R.p.i
S.i := -S.n.i
S.p.i := S.i
G.p.i := -S.p.i – R.n.i
```

```
G.p.v := 0.0
S.v := 5.0 * sin(6.28 * time)
S.p.v := 0.0
R.n.v := 0.0
S.n.v := -5.0 * sin(6.28 * time)
R.p.v := -5.0 * sin(6.28 * time)
R.v := -5.0 * sin(6.28 * time)
R.i := -0.005 * sin(6.28 * time)
R.p.i := -0.005 * sin(6.28 * time)
R.n.i := -0.005 * sin(6.28 * time)
S.n.i := 0.005 * sin(6.28 * time)
S.i := -0.005 * sin(6.28 * time)
S.p.i := -0.005 * sin(6.28 * time)
G.p.i := 0.0
```

Exemple (4)

```
G.p.v := 0.0
S.v := 5.0 * sin(6.28 * time)
S.p.v := 0.0
R.n.v := 0.0
S.n.v := -5.0 * sin(6.28 * time)
R.p.v := -5.0 * sin(6.28 * time)
R.v := -5.0 * sin(6.28 * time)
R.i := -0.005 * sin(6.28 * time)
R.p.i := -0.005 * sin(6.28 * time)
R.n.i := -0.005 * sin(6.28 * time)
S.n.i := 0.005 * sin(6.28 * time)
S.i := -0.005 * sin(6.28 * time)
S.p.i := -0.005 * sin(6.28 * time)
G.p.i := 0.0
```

factorisation

```
v1 := sin(6.28 * time)
v2 := 5.0 * v1
v3 := -v2
v4 := 0.005 * v1
v5 := -v4
```

```
G.p.v := 0.0
S.v := v2
S.p.v := 0.0
R.n.v := 0.0
S.n.v := v3
R.p.v := v3
R.v := v3
R.i := v5
R.p.i := v5
R.n.i := v5
S.n.i := v4
S.i := v5
S.p.i := v5
G.p.i := 0.0
```

Avantages de l'approche

- À partir d'un sous-modèle unique on peut dériver automatiquement plusieurs affectations selon les contraintes globales imposées par le modèle
- La génération du code du modèle se faisant plus tard que dans le cas causal, il y a plus d'information disponible que l'on peut exploiter pour réduire le modèle
- Les librairies ne sont plus redondantes