Systèmes cyber-physiques

Les étudier et les enseigner dans les cursus d'informatique à l'UM ?

Philippe.REITZ@LIRMM.FR

février 2016

Contexte

SCP = Systèmes Cyber-Physiques

- Djamel Seriai
  - Initier nos étudiants en informatique à piloter des dispositifs depuis un ordinateur : domotique, robotique, … (systèmes embarqués, téléphones mobiles, …)
  - À terme monter des modules communs EEA/INFO ?
- Jacques Malenfant
  - Exposé du 9/12/15 sur les SCP
- Quelques expériences personnelles
  - Expérimentations avec un kit Arduino
  - Apprentissage de la programmation au niveau lycée via le contrôle de robots simulés (RoboMind, Scratch, tortue Python)

Contexte

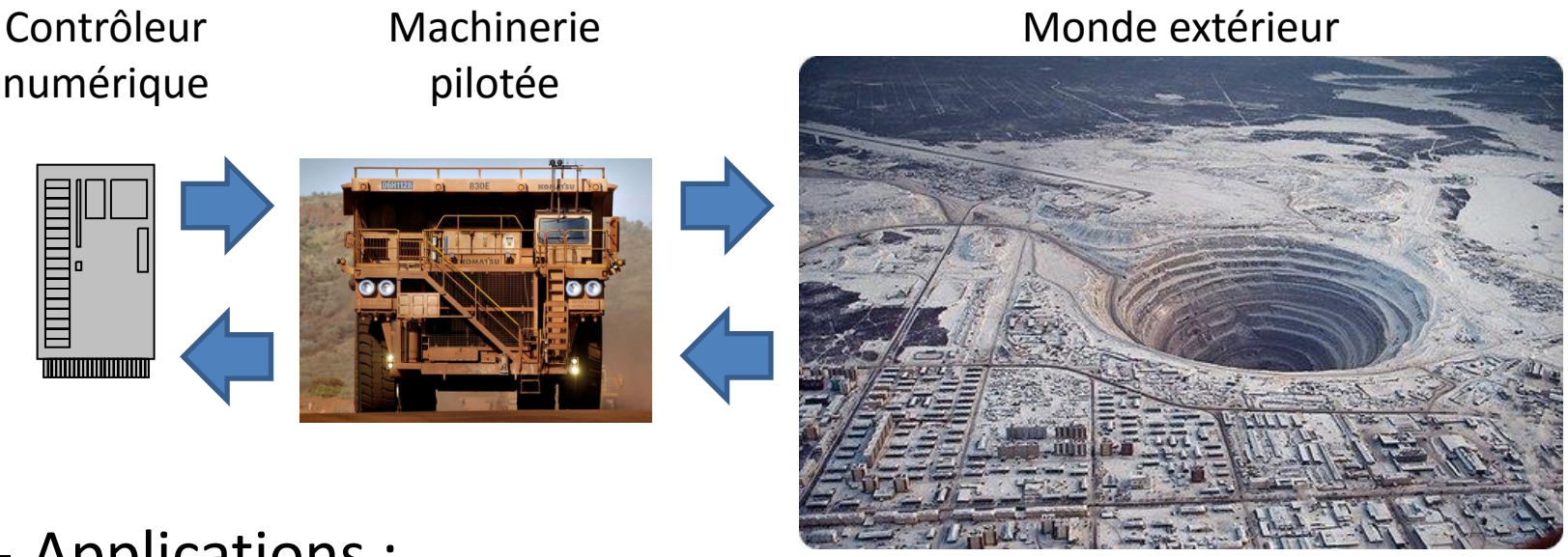
- Objectifs de l'exposé
  - Présenter les SCP avec un regard d'informaticien
    - Aspects théoriques : modèles de calcul
    - Aspects logiciels : structures de données, langages de programmation
    - Aspects scientifiques : modélisation de processus spatialisés
  - Les SCP ont-ils un intérêt :
    - pour nos recherches (Marel, LIRMM)
    - pour nos étudiants en informatique (FdS UM)
  - Existe-t-il des compétences locales

Plan de l'exposé

- Présentation générale des SCP
- Les aspects qui intéressent l'informatique
  - Les automates hybrides
  - Les langages de modélisation/simulation de systèmes
    - Sans la dimension spatiale : Modelica
    - Avec prise en compte de cette dimension : MGS
- Enseigner et étudier les SCP à l'UM
  - Intérêts pour nos recherches
  - Intérêts pour nos étudiants
- Conclusion et perspectives
- Références et sigles

Présentation générale

- Objectif
  - Piloter de façon autonome une machinerie pour qu'elle satisfasse un objectif assigné



  - Applications :
    - Robotique, domotique, systèmes embarqués, automates industriels, …

Présentation générale

- L'existant
  - deux approches de la modélisation de systèmes
    - systèmes dynamiques (dont l'automatique est une branche) ; un système est caractérisé par sa loi de comportement, en général une équation différentielle.
      - Cadre purement continu : état, temps, et espace
    - informatique ; un système est caractérisé par son programme
      - Cadre purement discret : état, temps et espace
- Le diagnostic
  - Question: comment intégrer ces modèles continus et discrets
  - Réponse : les modèles hybrides, incorporant continu et discret ☺
- Les difficultés
  - Modèles fondamentaux adaptés aux SCP ; comment les exploiter (algorithmique, analyse numérique)
  - Quelle chaîne de conception de ces systèmes (conception, simulation, vérification)

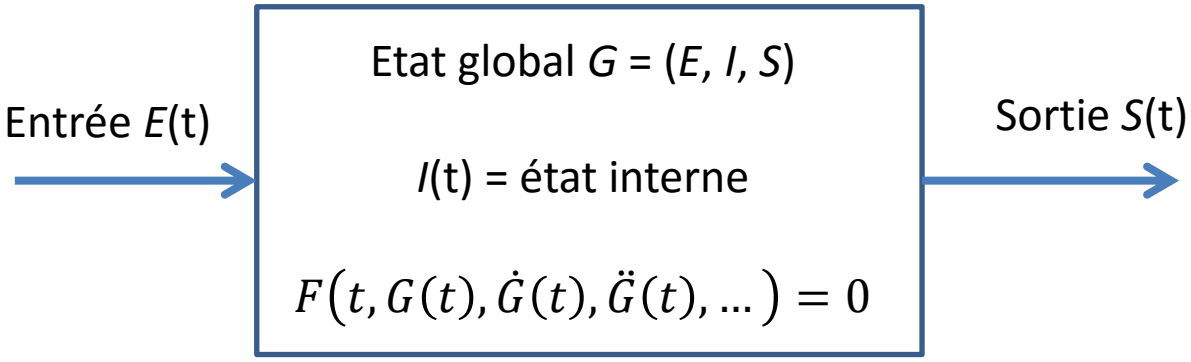
Présentation générale

- Les SCP : un nouveau domaine scientifique ?

  Pas vraiment (au moins depuis les années 50 [cybernétique]) : industrie (contrôle de processus industriels), transports (assistance au pilotage), …
- Pourquoi ce renouveau ?
  - Les coûts quasi-nuls du matériel.
  - Foisonnement de dispositifs numériques (objets connectés) en interaction avec le monde : capteurs en tout genre, actionneurs.
  - Interconnexion, stockage et traitement des données possibles à une échelle jamais vue jusque là.
  - Un intérêt de la communauté purement informatique à ces problèmes jusque-là traités par d'autres (automatique, robotique)
  - Effet de capitalisation/assimilation des connaissances : des domaines perçus comme distincts s'avèrent finalement proches.
  - Effet d'opportunité (marketing scientifique et technique) : comment relancer des thématiques en changeant de terminologie et en feignant d'ignorer le passé.

Approche usuelle EEA

- Les systèmes dynamiques



  - L'espace des états G est un $\mathbb{R}$ ou $\mathbb{C}$ espace vectoriel
  - L'espace des temps t est $\mathbb{R}$
  - F est l'équation différentielle qui caractérise le système

F peut être vue comme une spécification : il n'est pas toujours possible d'exhiber une solution f.

Il est possible d'éliminer le temps en changeant d'espace : espace des états à celui des phases (en physique : hamiltonien), transformations intégrales (Laplace, Fourier, …)

Approche usuelle EEA

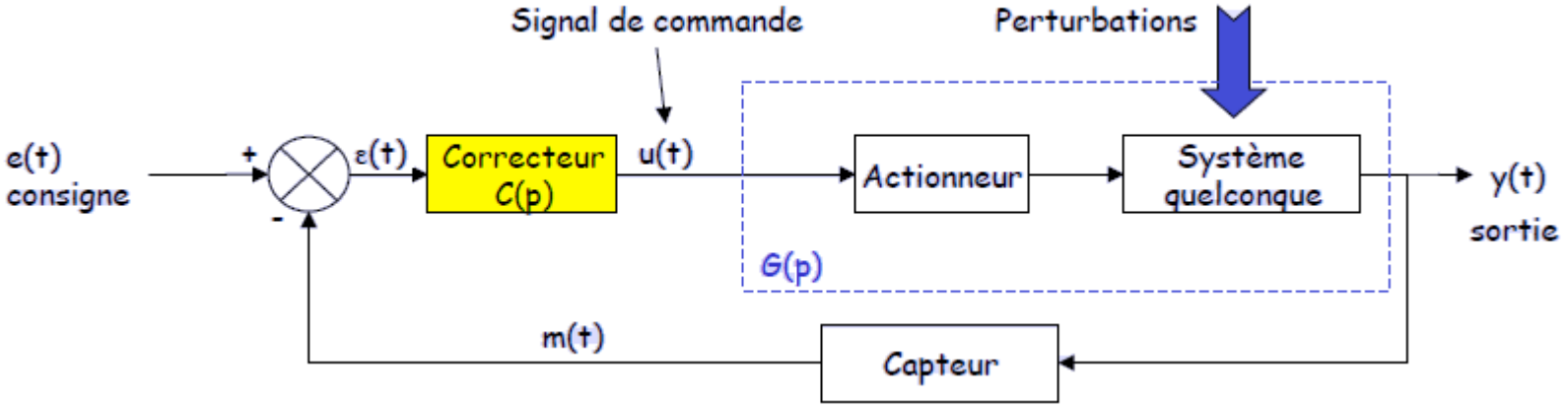
- Détermination de la solution de *F*
  - Soit il existe une solution analytique *f* de *F* (dans la *vraie vie*, c'est rarement le cas) : $G(t) = f(t)$
  - Soit une version approchée *g* est recherchée, via un *solveur* (ou intégrateur) d'équations :
    - Étant donné l'état initial G_0 à t_0 (voire plusieurs points de passage)
    - Le solveur calcule le graphe de *g* pour une série de temps $(t_i)_{i=1..n}$

      $G_k = g(t_k)$ pour tout *k* admissible
    - Le solveur optimise un critère de coût (mesure de la qualité de l'approximation *g* vis-à-vis d'une solution idéale *f*).

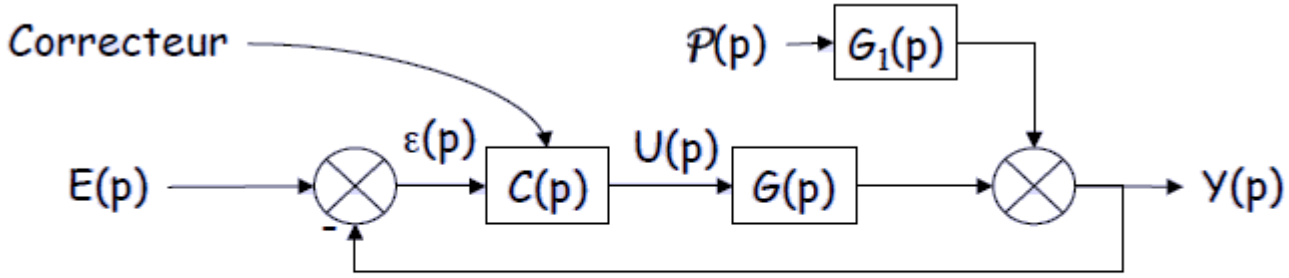
  $\Rightarrow$ *identification* de système

Approche usuelle EEA

- L'automatique [extraits de C. Albéa Sanchez, LAAS]
  - Systèmes dynamiques avec rétroaction



  - L'équation différentielle est transformée en une fraction rationnelle (transformation de Laplace)



$$\Rightarrow Y(p) = \frac{G(p)C(p)}{1+G(p)C(p)}E(p) + \frac{G_1(p)}{1+G(p)C(p)}\mathbf{P}(p)$$

Approche usuelle EEA

- L'automatique
  - Une longue histoire (débuts au XIX^{ème} siècle)
  - Outils de modélisation et de conception du systèmes
  - Prouvabilité de quelques propriétés
    - Commandabilité : il existe une politique de commande en entrée telle que la sortie satisfasse une propriété après un temps donné
    - Stabilité : comment varie la sortie quand l'entrée varie
    - Observabilité : il existe une politique de commande en entrée telle que l'analyse des entrées et sorties permet de déterminer les variables internes

Approche usuelle INFO : système temps réel, calcul concurrent et distribué

- Conception des logiciels de SCP très particulière
  - Langages permettant de spécifier, programmer et prouver (version opérationnelle d'une spécif.)
    - Exemple : VHDL, Verilog
  - Langages permettant une programmation réactive (basée sur les événements)
    - Contraintes de temps-réel
    - Concurrence des calculs, distribution
    - Exemples de langages industriels : Esterel, Ada
  - Dimension liée à la Physique des SCP souvent hors des compétences d'un informaticien

Synthèse des approches de modélisation de processus (spatialisés ou pas)

Etat	Temps	Espace	Modèles associés
Continu	Continu	-	Systèmes dynamiques : EDO (équations différentielles ordinaires) ; calculateurs analogiques ; …
Continu	Continu	Continu	Systèmes dynamiques spatialisés : EDP (équation aux dérivées partielles) ; …
Continu	Discret	-	Systèmes dynamiques à temps discret ; équations aux différences ; automates à alphabet continu (machine de Blum, Shub et Smale [BSS]) ; …
Continu	Discret	Discret	Eléments finis, …
Discret	Continu	-	Modélisation à événements discrets, …
Discret	Discret	Continu	Gaz sur réseau, …
Discret	Discret	Discret	Automates cellulaires, calcul spatialisé, …
Continu + Discret	Continu + Discret	-	Automates hybrides ; EDA (équations différentielles et algébriques hybrides) ; …

Plan de l'exposé

- Présentation générale des SCP
- Les aspects qui intéressent l'informatique
  - Les automates hybrides
  - Les langages de modélisation/simulation de systèmes
    - Sans la dimension spatiale : Modelica
    - Avec prise en compte de cette dimension : MGS
- Enseigner et étudier les SCP à l'UM
  - Intérêts pour nos recherches
  - Intérêts pour nos étudiants
- Conclusion et perspectives
- Références et sigles

Les automates hybrides [Henzinger'96]

- Présentation rapide

  (présentation détaillée juste après)

  – Automate hybride = automate caractérisé par :
    - son état initial
      – état discret + état continu
    - sa loi de transition
      – l'état discret évolue selon un automate
      – l'état continu évolue selon une EDO

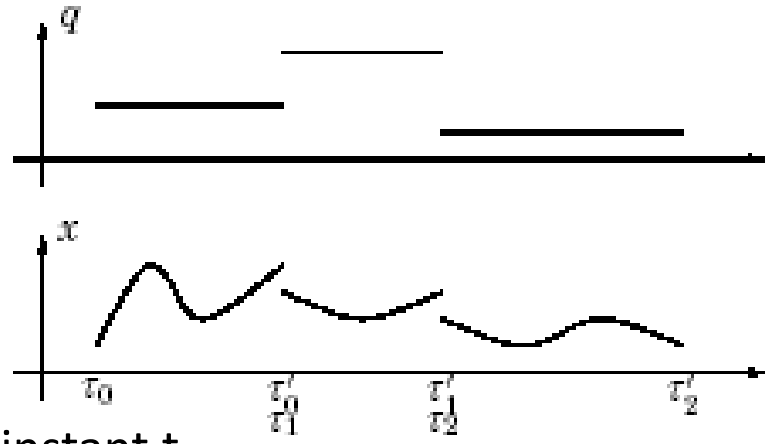
  – L'état évolue selon une loi s'appuyant sur un temps continu
    - alternance de phases continues et de transitions d'état instantanées.

Les automates hybrides

- Le modèle = automate, avec :
  - état = (q, x, τ) avec :
    - état discret q (pris dans un ensemble discret fini Q)
    - état continu x (pris dans un ensemble continu E)
    - temps τ (pris dans un ensemble continu T)
  - loi de transition, décomposée en 4 fonctions :
    - $f(q, x, x', \tau) = 0$: équation différentielle régissant l'évolution de l'état continu en phase stable
    - $g(q, x, x', \tau) \in \{0, 1\}$: prédicat détectant une transition d'état
    - $\delta(q, x, x', \tau) \in$ Q: fonction régissant la transition de l'état discret
    - $i(q, x, x', \tau) \in$ E : fonction régissant la transition de l'état continu (réinitialisation)

Les automates hybrides

- Sémantique
  - état courant = (q, x, τ)
  - phase stable
    - phase stable durant (τ, τ')
      $\Leftrightarrow \forall t \in [\tau, \tau'[, g(q, x, x', t) = 0$
    - durant toute cette phase, à chaque instant t,
      - l'état discret de l'automate reste à q
      - l'état continu x évolue selon l'équation différentielle f
  - phase de transition
    - transition de phase à l'instant t $\Leftrightarrow g(q, x, x', t) = 1$
    - lorsqu'une transition de phase est détectée, alors :
      - l'état discret de l'automate passe à $\delta(q, x, x', t)$
      - l'état continu x est réinitialisé à $i(q, x, x', t)$



Les automates hybrides

Une balle rebondissante

Position de la balle : $x \geq 0$

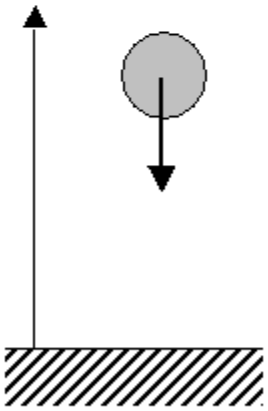
$$\Rightarrow \text{état continu} = (x, v = \dot{x})$$

Accélération : $g > 0$

Coefficient d'amortissement : $c \in [0, 1]$

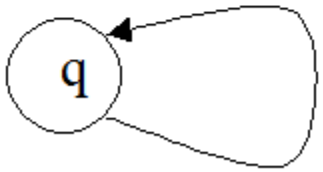
équation différentielle en q :

$$\begin{cases} v = \dot{x} \\ \dot{v} = -g \end{cases}$$



condition de la transition :

$$x = 0 \text{ et } v < 0$$



réinitialisation :

$$v := -cv$$

Les automates hybrides

- Principaux résultats
  - dans le cas général, ils sont tous dramatiques (questions indécidables, au mieux NP)
    - Atteignabilité : quelle entrée e pour obtenir une sortie s
    - Contrôlabilité : atteignabilité en temps borné
    - Stabilité : si e change, comment change s
  - dans quelques cas particuliers (automates hybrides linéaires), les automaticiens ont réussi à trouver des algorithmes raisonnables pour résoudre leurs problèmes (cf. vérification).
    - système HYTECH d'Henzinger.
    - système TEMPO de N. Lynch

Plan de l'exposé

- Présentation générale des SCP
- Les aspects qui intéressent l'informatique
  - Les automates hybrides
  - Les langages de modélisation/simulation de systèmes
    - Sans la dimension spatiale : Modelica
    - Avec prise en compte de cette dimension : MGS
- Enseigner et étudier les SCP à l'UM
  - Intérêts pour nos recherches
  - Intérêts pour nos étudiants
- Conclusion et perspectives
- Références et sigles

Le langage Modelica (1/2)

- Un langage déclaratif, orienté objet, né fin 90 dans les pays scandinaves (Simula).
- Un langage spécialement adapté à la modélisation/simulation de systèmes
  - applicable à tous les domaines (voir bémols après)
  - Bibliothèques multi-domaines
  - Modelica = UML pour les modélisateurs ?
- Un langage pour décrire des modèles hybrides

Le langage Modelica (2/2)

- Un langage de modélisation normalisé
  - Les spécifications sont normalisées
  - Les outils informatiques (compilateurs) ont pour seule obligation de se conformer à ces spécifications
- Domaines d'application à ce jour
  - Mécatronique, Electronique, Systèmes de puissance, Hydraulique, Chimie, Biologie, Aérodynamique, …

Modéliser / simuler avec Modelica : processus général (1/2)

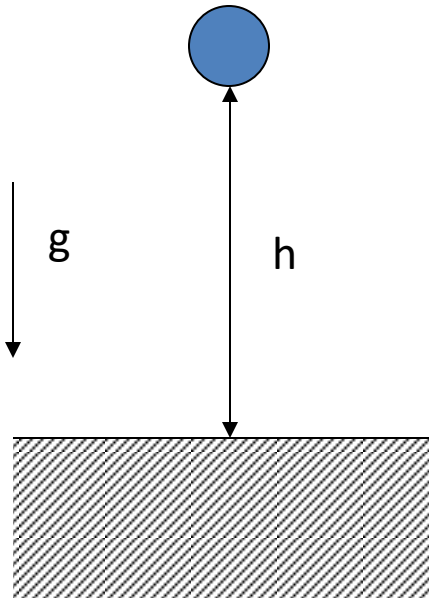
1. définition d'un modèle du système
   - orienté composant
     - variables d'état discrètes et/ou continues
     - dynamique (temps continu) décrite par
       - équations différentielles
       - programmes
     - héritage entre composants
       - composants réutilisables (bibliothèques de composants)
   - modèle = assemblage de composants
     - éditeurs graphiques

Modéliser / simuler avec Modelica : processus général (2/2)

2. définition des paramètres d'une simulation
3. compilation du modèle paramétré
   - mise à plat du modèle objet
   - production d'un système d'EDA hybride

     EDA = équation différentielle et algébrique ; formalisme équivalent aux automates hybrides
4. exécution du programme de simulation
   - exploitation de solveurs d'EDA hybrides adaptés au problème

Modelica : balle rebondissante




```
model BalleRebondissante
  parameter Real c = 0.7;
  parameter Real g = 9.81;
            Real h (start=1);
            Real v;
equation
  der(h) = v;
  der(v) = -g;
  when h <= 0 then
    reinit(v, -c*pre(v));
    reinit(h, 0);
  end when;
end BalleRebondissante;
```

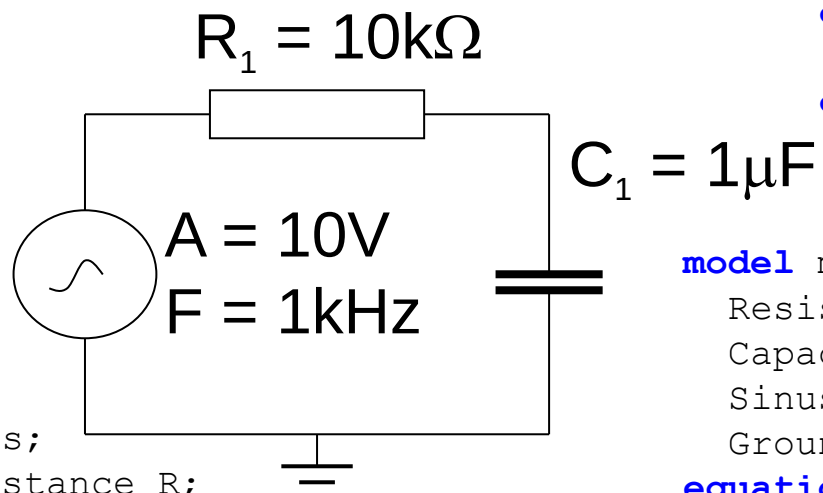
Modelica : circuit électronique

```
type Voltage = Real(quantity="Voltage", unit="V");
type Current = Real(quantity="Current", unit="A");

connector Pin
  Voltage      v;
  flow Current i;
end Pin;

partial model TwoPins
  Pin     p, n;
  Voltage u;
  Current i;
equation
  0 = p.i + n.i;
  u = p.v - n.v;
  i = p.i;
end TwoPins;

model Resistor
  extends TwoPins;
  parameter Resistance R;
equation
  u = R*i;
end Resistor;
```




```
model Capacitor
  extends TwoPins;
  parameter Capacitance C;
equation
  C*der(u) = i;
end Capacitor;

model SinusSource
  extends   TwoPins;
  parameter Frequency F;
  parameter Voltage   A;
protected
  constant Real PI = 3.1415927;
equation
  u = A * sin(time * 2*PI*F);
end SinusSource;
```

```
model monCircuitTest
  Resistor    R1(R=1e4);
  Capacitor   C1(C=1e-6);
  SinusSource S(A=10, F=1000);
  Ground      G;
equation
  connect(S.p, R1.p);
  connect(R1.n, C1.p);
  connect(S.n, G);
  connect(C1.n, G);
end monCircuitTest;
```

Alternatives/implémentations de Modelica

- Scilab / Scicos / Xcos [Inria]
- Labview
  - mention particulière : le plus proche des besoins de programmation des SCP ; connectable à des appareils, qui peuvent être pilotés.
- MapleSim (Maple)
- SystemModeler (Mathematica)
- Dymola (Catia)
- logiciels industriels de CAO

Et la dimension spatiale ?

- La dimension spatiale est en général ignorée
  - Théorie des systèmes dynamiques, automatique
  - Modelica et assimilés
- Prendre en compte la dimension spatiale
  - On passe des EDO à des EDP
  - Des solveurs existent (éléments finis + EDA), mais soucis de stabilité (les transitions discrètes ne font pas bon ménage avec les solveurs classiques $\Rightarrow$ pb d'analyse numérique)
  - La modélisation de l'environnement spatial du système est moins triviale qu'il n'y paraît
    - Au final, un tel solveur peut servir de moteur à un jeu 3D intégrant des effets physiques : le langage doit donc être capable de décrire complètement un monde virtuel…
    - Quelques tentatives dans Modelica, sans résultat probant.

Modélisation informatique de processus spatialisés

- Beaucoup de modèles de calcul spatialisé
  - Calcul chimique abstrait (métaphore chimique)
    - Langages Gamma, HOCL, Jo-Caml,
  - P-systems, L-systems (métaphore biologique)
  - Automates cellulaires
  - Agents situés / Calcul ambiant
  - Calcul amorphe, BLOB-calculus
  - Processus spatialisés
- Exposé d'une approche cohérente avec les SCP
  - Le projet MGS

Le langage MGS – J.L. Giavitto & al.

- 1991 : thèse sur le langage 8,5
  - Programme = spécification d'une contrainte sur une collection (espace) de flots (temps) = notion de tissu.
  - Pas de référence à des positions absolues (temps, espace) : accès aux voisins via des déplacements élémentaires.
- Période 1992 à 2000
  - Généralisation des concepts de collection et de flot à celui de champ de données (DF)
    - Tableau = fonction totale de $I_1 \times I_2 \cdots \times I_n \to V$, avec I_k inclus dans $\mathbb{Z}$
      - accès à une composante via un indice absolu
    - Champ de donnée = fonction partielle de $I_1 \times I_2 \cdots \times I_n \to V$, avec I_k idem
      - accès via un déplacement par rapport à un indice courant (implicite)
  - Focus sur les champs de données basés sur des groupes (GBF)
    - GBF = fonction partielle de $G \to V$, où *G* muni d'une structure de groupe
- A partir de 2001 : MGS (Modèle Général de Simulation)

MGS en quelques lignes

- MGS : un langage de récriture parallèle
  - Unificateur de nombreux modèles de calcul parallèles
- Principe général
  - une **collection** est un espace ayant une certaine topologie (entièrement décrite par son groupe des déplacements)
    - chaque point de la collection porte une donnée
  - il doit être possible d'identifier des sous-collections
    - chaque **sous-collection** est délimitée par un **bord** dans la collection
  - une **transformation** remplace une sous-collection par une autre dont le bord est compatible avec celui de la première
    - une transformation est décrite par un ensemble de **règles**

      si motif sous-collection

      tel que condition

      alors sous-collection de substitution
  - un **calcul** sur une collection est une application itérée de transformations sur cette collection

MGS

- Un exemple : recherche du minimum

```
T = { (x, y / x<=y) => x }
X = Multiset(4, 3, 3, 1, 2, 5, 1, 4, 2, 5)
T['fixpoint](X)
```

  renvoie `Multiset(1)`

- Autre exemple : tri d'un tableau

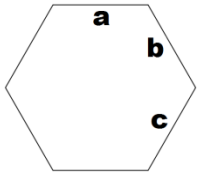
```
T = { (x, y / x>y) => y, x }
X = Seq(4, 3, 3, 1, 2, 5, 1, 4, 2, 5)
T['fixpoint](X)
```

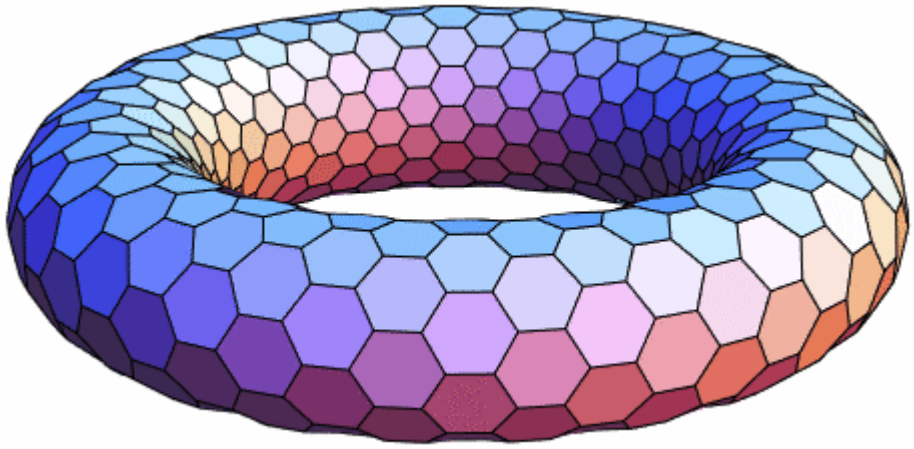
  renvoie `Seq(1, 1, 2, 2, 3, 3, 4, 4, 5, 5)`

MGS

- Spécification d'un espace

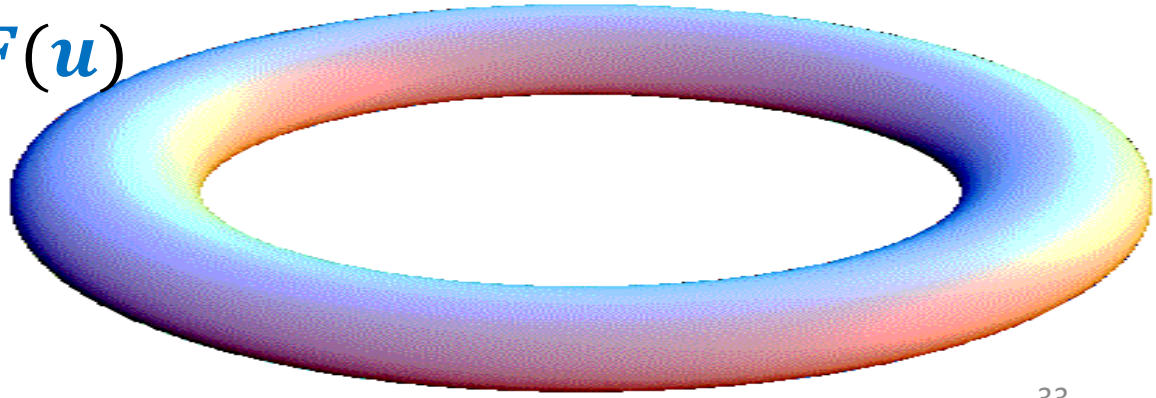
```
gbf hex_torus = <
  a, b, c;
  a+c = b;
  20 (a-c+a+b) = 0;
  10 (b+c) = 0;
>;;
```





- Simulation de diffusion 2D (Turing)

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{k} \cdot \nabla^2 \boldsymbol{u} + \boldsymbol{F}(\boldsymbol{u})$$



pour finir sur MGS

- notion de GBF : généralise la plupart des structures de données connues
  - Tuples, tableaux (au sens large), (multi-)ensembles, séquences (listes, files, piles), arbres, graphes, …
- Slogans de MGS
  - Programmer, c'est assembler des structures de données
    - Toute modification sur une structure de donnée doit résulter d'une transformation locale (données voisines)
    - Plusieurs transformations locales peuvent être appliquées en parallèle (parallélisme implicite).
    - Le programme est d'un style équationnel
  - Quelles sont les structures de données adaptées à la modélisation / simulation

Plan de l'exposé

- Présentation générale des SCP
- Les aspects qui intéressent l'informatique
  - Les automates hybrides
  - Les langages de modélisation/simulation de systèmes
    - Sans la dimension spatiale : Modelica
    - Avec prise en compte de cette dimension : MGS
- Enseigner et étudier les SCP à l'UM
  - Intérêts pour nos recherches
  - Intérêts pour nos étudiants
- Conclusion et perspectives
- Références et sigles

Intérêt des SCP pour nos recherches

- Aspects fondamentaux
  - Modèles de calcul étendus
    - Sortir du cadre purement discret : calculs sur $\mathbb{R}$
    - Intégrer des aspects non traditionnels : l'espace et sa géométrie, sa dynamique, …
- Aspects logiciels
  - Quels langages pour programmer les SCP
    - Modelica a une approche *composants*
  - Comment augmenter la puissance d'expression des langages (en général)
    - Style équationnel des EDA séduisant
  - Comment les compiler, les exécuter

Intérêt des SCP pour nos étudiants

- Vision unifiée de domaines a priori distincts
  - Programmes (informatique) et équations différentielles (automatique) sont deux façons de décrire des processus.
- Formation aux métiers des SCP
  - Robotique, domotique
  - Contrôle de processus industriels

Quelles compétences à l'UM

- Du coté UM/EEA
  - Aucun cours sur les modèles hybrides (niveau L ou M)
  - Une ou deux thèses de robotique font référence aux modèles hybrides, sans toutefois en exploiter les techniques
- Du coté UM/INFO
  - Aucun cours non plus
- Sur Montpellier
  - À l'UM
    - Enseignement de Scilab
  - Au Cirad
    - Equipe de Jean-Chistophe Soulié (DEVS)
    - Langage dédié à la modélisation de Pascal Degenne

Conclusion

- Intérêts évidents des SCP
  - Thème dans le champ de l'informatique
  - Besoin de personnels formés dans le monde professionnel
  - Champ d'application de recherches actuelles
  - Champ d'inspiration de recherches à venir
- Quelques difficultés à surmonter
  - Des enseignants à former
  - Repenser l'enseignement initial EEA/INFO
    - Mise en commun d'un bagage de connaissances minimal

Perspectives

- Programmer, c'est contrôler des trajectoires dans un espace totalement construit
  - Programmer, c'est définir un espace de données
    - Chaque point est un environnement
    - Chaque instruction permet de relier deux (classes de) points
      - Environnement avant exécution
      - Environnement après exécution
    - Une composition d'instructions définit donc des (classes de) chemins
  - Exécuter un programme, c'est :
    - Définir un point de départ
    - Dérouler le chemin de calcul que le programme induit

  $\Rightarrow$ cf. notion de *trace* de calcul
- Analogie programme / champ (au sens physique)
  - Un champ oriente localement les trajectoires en chaque point

Perspectives

- La majorité des technologies sont des branches de la Physique ; l'informatique n'est pas perçue comme telle.
  - Les SCP illustrent ce rapprochement
  - Le mouvement existe depuis longtemps
    - Physique discrète de E. Fredkin (années 60) ou E. Tonti (1990)
    - Automates cellulaires et physique ; K Suze (années 70)

Références

- Langage Modelica :
  http://modelica.org
- Les travaux de l'équipe de J.L. Giavitto
  https://www.ibisc.univ-evry.fr/~mgs/
- Les modèles de calculs spatialisés
  Une synthèse sur http://lucacardelli.name
  - Calcul chimique abstrait (Boudol et Berry)
    - Calcul joint, calcul bleu, …
  - Calcul ambiant, processus spatialisés (Cardelli)
  - Agents situés, automates cellulaires
  - Calcul amorphe, BLOB-calculus, …
- Les systèmes dynamiques
  - Plein d'ouvrages ; privilégier ceux liés à l'automatique
  - Extraits d'un cours d'automatique de C. Albéa Sanchez au LAAS

Sigles

EDA = Equation Différentielle et Algébrique [*DAE**]

EDO = Equation Différentielle Ordinaire [*ODE**]

EDP = Equation aux Dérivées Partielles [*PDE**]

EEA = Electronique, Electrotechnique et Automatique

SCP = Systèmes Cyber-Physiques [*CPS**]

* : sigle anglo-saxon