



PyPHS: Un module Python pour la modélisation et la simulation à passivité garantie de systèmes multi-physiques

Antoine Falaize, Thomas Hélie, Tristan Lebrun

► To cite this version:

Antoine Falaize, Thomas Hélie, Tristan Lebrun. PyPHS: Un module Python pour la modélisation et la simulation à passivité garantie de systèmes multi-physiques. Journées Jeunes Chercheurs en Audition, Acoustique musicale et Signal audio, Nov 2016, Paris, France. . hal-01470954

HAL Id: hal-01470954

<https://hal.science/hal-01470954>

Submitted on 18 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



PyPHS: UN MODULE PYTHON POUR LA MODÉLISATION ET LA SIMULATION À PASSIVITÉ GARANTIE DE SYSTÈMES MULTI-PHYSIQUES

Antoine Falaize, Thomas Hélie, Tristan Lebrun
Project-team S3 (Sound Signals and Systems)
UMR 9912 (IRCAM-CNRS-UPMC)

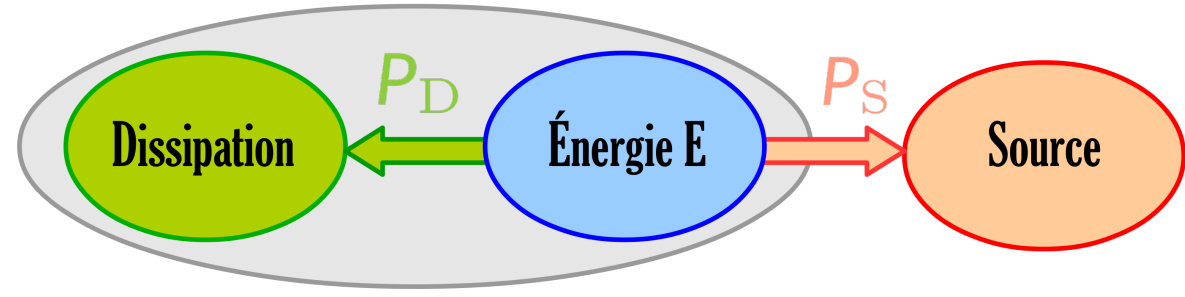


MOTIVATIONS

Constat:

Les systèmes physiques vérifient un **bilan de puissance**:

$$\frac{dE}{dt} = -P_D - P_S$$



Objectif:

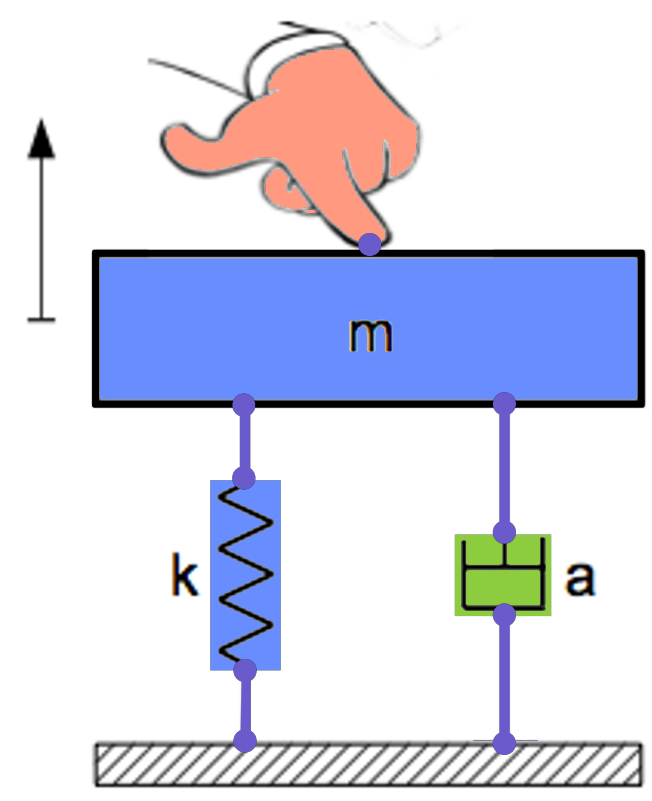
Modéliser et **simuler** en préservant cette propriété.

Approche:

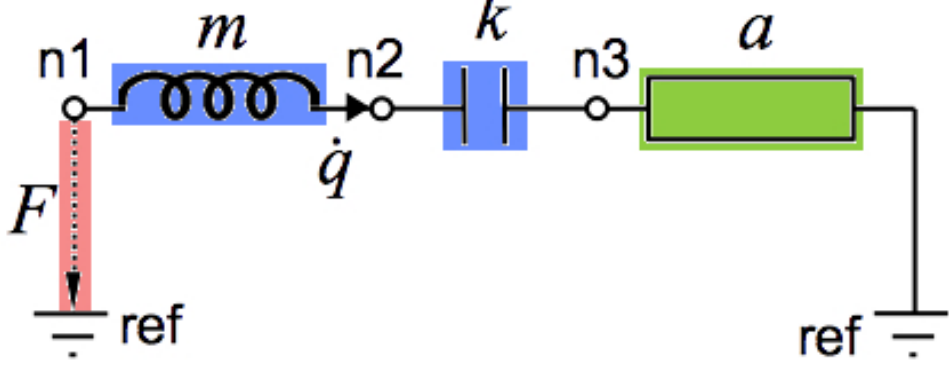
Les **Systèmes Hamiltoniens à Ports**

EXEMPLE

Oscillateur amorti excité par une force



Graphe associé



Dictionnaire

	Var.	Function	Flux	Effort
m	$x_1 = m \dot{q}$	$H_1 = \frac{x_1^2}{2m}$	$v_1 = \frac{dH_1}{dx_1}$	$F_1 = \frac{dx_1}{dt}$
k	$x_2 = q$	$H_2 = \frac{k}{2} x_2^2$	$v_2 = \frac{dH_2}{dx_2}$	$F_2 = \frac{dH_2}{dx_2}$
a	$w_3 = \dot{q}$	$z_3 = Aw_3$	$v_3 = w_3$	$F_3 = z_3$
In	\emptyset	\emptyset	$v_4 = y_4$	$F_4 = u_4$

Netlist

```
mechanics.mass M ('n1','n2'): M=('M',0.1);  
mechanics.stiffness K ('n2','n3'): K=('K',1e6);  
mechanics.damper A ('n3','ref'): A=('A',1);  
mechanics.source IN ('ref','n1'): type='force';
```

Structure retournée par PyPHS

$$\underbrace{\begin{pmatrix} F_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}}_b = \underbrace{\begin{pmatrix} 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}}_J \underbrace{\begin{pmatrix} v_1 \\ F_2 \\ F_3 \\ F_4 \end{pmatrix}}_a$$

DICTIONNAIRE ACTUEL

Mécanique (1D): masses, ressorts lin./nonlin. (cubique, saturants, etc.), amortissements lin./nonlin., visco-élasticité (dérivateur fractionnaire).

électronique: piles, bobines et condensateurs lin./nonlin., résistances, transistors, diodes, triodes.

Magnétique: Aimants, capacités magnétiques lin./nonlin, rÃl'sisto-inductance (intégrateurs fractionnaires).

Thermique: sources de chaleur, capacités thermiques.

Connections: couplages électromagnétiques, couplages électromécaniques, transferts irréversibles, gyrateurs, transformateurs.

I. SYSTÈME HAMILTONIEN À PORTS [1]

n_E Composants stockants :

- État $\mathbf{x} \in \mathbb{R}^{n_E}$
- $E = H(\mathbf{x}) = \sum_{n=1}^{n_E} H_n(x_n) \geq 0$
- $\frac{dE}{dt} = \nabla H^\top \frac{d\mathbf{x}}{dt} = \sum_{n=1}^{n_E} \frac{dH_n}{dx_n} \frac{dx_n}{dt}$

n_D Composants dissipatifs :

- Variable $\mathbf{w} \in \mathbb{R}^{n_D}$
- Fonction $\mathbf{z} : \mathbb{R}^{n_D} \rightarrow \mathbb{R}^{n_D}$
- $P_D = \mathbf{w}^\top \mathbf{z} = \sum_{n=1}^{n_D} w_n z_n \geq 0$

n_S Sources externes :

- Entrée \mathbf{u} , sortie \mathbf{y}
- $P_S = \mathbf{y}^\top \mathbf{u} = \sum_{n=1}^{n_S} y_n u_n$

Interconnexion conservative :

- Puissance reçue:** $P = \mathbf{f} \mathbf{e}$
- flux \mathbf{f} et effort \mathbf{e} (elec: $\mathbf{f} \equiv i$, $\mathbf{e} \equiv v$)
- $N = n_E + n_D + n_S$
- $\sum_{n=1}^N P_n = \sum_{n=1}^N \mathbf{f}_n \mathbf{e}_n = 0$

Représentation d'état

$$\underbrace{\begin{pmatrix} \frac{d\mathbf{x}}{dt} \\ \mathbf{w} \\ \mathbf{y} \end{pmatrix}}_b = \underbrace{\begin{pmatrix} \mathbf{J}_x & -\mathbf{K} & \mathbf{G}_x \\ \mathbf{K}^\top & \mathbf{J}_w & \mathbf{G}_x \\ -\mathbf{G}_x^\top & -\mathbf{G}_w^\top & \mathbf{J}_y \end{pmatrix}}_{\mathbf{J} = -\mathbf{J}^\top} \underbrace{\begin{pmatrix} \nabla H(\mathbf{x}) \\ \mathbf{z}(\mathbf{w}) \\ \mathbf{u} \end{pmatrix}}_a$$

Encode le bilan de puissance

$$\begin{aligned} \frac{dE}{dt} + P_D + P_S &= \nabla H^\top \frac{d\mathbf{x}}{dt} + \mathbf{z}(\mathbf{w})^\top \mathbf{w} + \mathbf{u}^\top \mathbf{y} \\ &= \mathbf{a}^\top \mathbf{b} \\ &= \mathbf{a}^\top \mathbf{J} \mathbf{a} = 0, \quad (\mathbf{J} \text{ antisym.}). \end{aligned}$$

Système passif

- entrée u/état x/contrainte w/sortie y.**
- Énergie $H \equiv$ **Fonction de Lyapunov** (stabilité/contrôle).

III. MÉTHODE NUMÉRIQUE [3]

Objectif: Garantir un **bilan de puissance** à temps discret:

$$\frac{\delta E}{\delta T}[k] = \mathcal{P}[k] - \mathcal{D}[k].$$

Choix:

$$\frac{\delta E[k]}{\delta T} = \frac{E[k+1] - E[k]}{\delta T} = \frac{H(\mathbf{x}[k+1]) - H(\mathbf{x}[k])}{\delta T}$$

Une **Solution:** gradient discret [?]

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &\rightarrow \frac{\delta \mathbf{x}[k]}{\delta T} = \frac{\mathbf{x}[k+1] - \mathbf{x}[k]}{\delta T} \\ \nabla H(\mathbf{x}) &\rightarrow \nabla^d H(\mathbf{x}[k], \delta \mathbf{x}[k]) \end{aligned}$$

avec

$$[\nabla^d H(\mathbf{x}, \delta \mathbf{x})]_n = \frac{H_n([\mathbf{x} + \delta \mathbf{x}]_n) - H_n([\mathbf{x}]_n)}{[\delta \mathbf{x}]_n}.$$

\Rightarrow bilan de puissance discret avec

$$\frac{\delta E}{\delta T}[k] = \nabla^d H^\top(\mathbf{x}[k], \delta \mathbf{x}[k])^\top \frac{\delta \mathbf{x}[k]}{\delta T}$$

CONCLUSIONS

PyPHS: <https://afalaize.github.io/pyphs/>

- Licence CECILL (CEA-CNRS-INRIA).
- PYTHON 2.7 supporté sous Mac OSX, Windows 10 et Linux.

Perspectives Résultats disponible à implémenter: variables d'ondes aller/retour, méthodes numériques d'ordre supérieur, résolution des conflits de réalisabilité, génération de lois de commandes (approche par platitude).

RÉFÉRENCES

- [1] Duijndam, V., Macchelli, A., Stramigioli, S., & Bruyninckx, H. (Eds.). (2009). Modeling and control of complex physical systems: the port-Hamiltonian approach. Springer Science & Business Media.
- [2] Falaize, A., & Hélie, T. (2016). Passive Guaranteed Simulation of Analog Audio Circuits: A Port-Hamiltonian Approach. Applied Sciences, 6(10), 273.
- [3] Lopes, N., Hélie, T., & Falaize, A. (2015). Explicit second-order accurate method for the passive guaranteed simulation of port-Hamiltonian systems. IFAC-PapersOnLine, 48(13), 223-228.

II. ANALYSE DU GRAPHE [2]

Objectif : Générer le système d'équations.

1. Graphe \rightarrow Matrice d'incidence réduite Γ :

$$[\Gamma]_{n,b} = \begin{cases} 1 & \text{si la branche } b \text{ entre le noeud } n, \\ -1 & \text{si la branche } b \text{ sort du noeud } n. \end{cases}$$

2. $\Gamma \rightarrow$ Lois de Kirchhoff généralisées:

- Potentiels sur les noeuds $\mathbf{p} \in \mathbb{R}^{n_N}$.
- Référence N_0 arbitraire.
- $\Gamma^\top \mathbf{p} = \mathbf{e}$, (Loi des mailles).
- $\Gamma \mathbf{f} = 0$, (Loi des noeuds).

3. Séparation suivant le type de contrôle

$$\Gamma = \left(\begin{array}{c|c} \gamma_0 & \gamma_f \end{array} \right)$$

- γ_0 : référence,
- γ_e : effort-ctrl,
- γ_f : flux-ctrl.

4. Critère de réalisabilité:

Forme $\mathbf{b} = \mathbf{J} \cdot \mathbf{a}$ possible si γ_f **inversible**.

Question: Quel contrôle pour les composants dissipatifs bijectifs?

Algorithme:

Données:

Netlist et **dictionnaire** de composants.

Résultat

- Si réalisable:
 - partition de Γ ,
 - structure $\mathbf{b} = \mathbf{J} \cdot \mathbf{a}$.
- Sinon: détection des conflits.