PyPHS Documentation
Version 0.1.9b2

Antoine Falaize

17 novembre 2016

1 Introduction

The python package `pyphs` is dedicated to the treatment of *passive multiphysical systems* in the *Port-Hamiltonian Systems* (PHS) formalism. This formalism structures physical systems into
— energy conserving parts,
— power dissipating parts and
— source parts.
This guarantees a *power balance* is fulfilled, including for numerical *simulations* based on an adapted *numerical method*.

1. Systems are described by *directed multi-graphs* (`networkx.MultiDiGraph`).
2. The time-continuous port-Hamiltonian structure is build from an *automated graph analysis*.
3. The discrete-time port-Hamiltonian structure is derived from a *structure preserving numerical method*.
4. LaTeXdescription code and C++ simulation code are automatically generated.

1.1 Installation

Notice only python 2.7 is supported.

It is recommanded to install `pyphs` using PyPI (the Python Package Index). In terminal :

```
pip install pyphs
```

Mac OSX only : An installation for *Anaconda* users is also available. In terminal :

```
conda install -c afalaize pyphs
```

1.2 The PHS formalism

Below is a recall of the Port-Hamiltonian Systems (PHS) formalism. For details, the reader is referred to the *e.g.* the acaemic reference [Falaize and Hélie, 2016].

We consider systems that can be described by the following time-continuous non-linear state-space representation :

$$\underbrace{\begin{pmatrix} \frac{d\mathbf{x}}{dt} \\ \mathbf{w} \\ \mathbf{y} \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xx}} & \mathbf{M}_{\mathtt{xw}} & \mathbf{M}_{\mathtt{xy}} \\ \mathbf{M}_{\mathtt{wx}} & \mathbf{M}_{\mathtt{ww}} & \mathbf{M}_{\mathtt{wy}} \\ \mathbf{M}_{\mathtt{yx}} & \mathbf{M}_{\mathtt{yw}} & \mathbf{M}_{\mathtt{yy}} \end{pmatrix}}_{\mathbf{M}} \cdot \underbrace{\begin{pmatrix} \nabla \mathrm{H}(\mathbf{x}) \\ \mathbf{z}(\mathbf{w}) \\ \mathbf{u} \end{pmatrix}}_{\mathbf{a}} \tag{1}$$

where

$$\mathbf{M} = \underbrace{\begin{pmatrix} \mathbf{J}_{\mathtt{xx}} & \mathbf{J}_{\mathtt{xw}} & \mathbf{J}_{\mathtt{xy}} \\ \mathbf{J}_{\mathtt{wx}} & \mathbf{J}_{\mathtt{ww}} & \mathbf{J}_{\mathtt{wy}} \\ \mathbf{J}_{\mathtt{yx}} & \mathbf{J}_{\mathtt{yw}} & \mathbf{J}_{\mathtt{yy}} \end{pmatrix}}_{\mathbf{J}} - \underbrace{\begin{pmatrix} \mathbf{R}_{\mathtt{xx}} & \mathbf{R}_{\mathtt{xw}} & \mathbf{R}_{\mathtt{xy}} \\ \mathbf{R}_{\mathtt{wx}} & \mathbf{R}_{\mathtt{ww}} & \mathbf{R}_{\mathtt{wy}} \\ \mathbf{R}_{\mathtt{yx}} & \mathbf{R}_{\mathtt{yw}} & \mathbf{R}_{\mathtt{yy}} \end{pmatrix}}_{\mathbf{R}} \tag{2}$$

and

- $\mathbf{J} : \mathbf{x} \mapsto \mathbf{J}(\mathbf{x})$ is a skew-symmetric matrix :

$$\mathbf{J}_{\alpha\beta} = -\mathbf{J}_{\beta\alpha}^{\intercal} \quad \text{for} \quad (\alpha, \beta) \in \{\mathtt{x}, \mathtt{w}, \mathtt{y}\}^2,$$

- $\mathbf{R} : \mathbf{x} \mapsto \mathbf{R}(\mathbf{x}) \succeq 0$ is a positive definite matrix,
- $\mathbf{x} : t \mapsto \mathbf{x}(t) \in \mathbb{R}^{n_{\mathtt{x}}}$ is the *state vector*,
- $\mathrm{H} : \mathbf{x} \mapsto \mathrm{H}(\mathbf{x}) \in \mathbb{R}_+$ is a *storage function* (convex and positive-definite scalar function with $\mathrm{H}(0) = 0$),
- $\nabla \mathrm{H} : \mathbf{x} \mapsto \nabla \mathrm{H}(\mathbf{x}) \in \mathbb{R}^{n_{\mathtt{x}}}$ denote the gradient of the storage function with the *storage power*

$$\mathrm{P}_{\mathbf{x}} = \frac{d\mathbf{x}}{dt} \cdot \nabla \mathrm{H}(\mathbf{x}),$$

- $\mathbf{w} : t \mapsto \mathbf{w}(t) \in \mathbb{R}^{n_{\mathtt{w}}}$ is the *dissipation vector variable*,
- $\mathbf{z} : \mathbf{w} \mapsto \mathbf{z}(\mathbf{w}) \in \mathbb{R}^{n_{\mathtt{w}}}$ is a *dissipation function* (with positive definite jacobian matrix and $\mathbf{z}(0) = 0$) for the *dissipated power*

$$\mathrm{P}_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{z}(\mathbf{w}) + \mathbf{a} \cdot \mathbf{R} \cdot \mathbf{a},$$

- $\mathbf{u} : t \mapsto \mathbf{u}(t) \in \mathbb{R}^{n_{\mathtt{y}}}$ is the *input vector*,
- $\mathbf{y} : t \mapsto \mathbf{y}(t) \in \mathbb{R}^{n_{\mathtt{y}}}$ is the *output vector*,
- **the power received *by* the sources *from* the system is**

$$\mathrm{P} = \mathbf{u} \cdot \mathbf{y}.$$

The state is split according to $\mathbf{x} = (\mathbf{x}_{\mathtt{l}}^{\intercal}, \mathbf{x}_{\mathtt{nl}}^{\intercal})^{\intercal}$ with

$\mathbf{x}_{\mathtt{l}} = (x_1, \cdots, x_{n_{\mathtt{xl}}})^{\intercal}$ the states associated with the quadratic components of the storage function $\mathrm{H}_{\mathtt{l}}(\mathbf{x}_{\mathtt{l}}) = \frac{\mathbf{x}_{\mathtt{l}} \cdot \mathbf{Q} \cdot \mathbf{x}_{\mathtt{l}}}{2}$

$\mathbf{x}_{\mathtt{nl}} = (x_{n_{\mathtt{xl}}+1}, \cdots, x_{n_{\mathtt{x}}})^{\intercal}$ the states associated with the non-quadratic components of the storage function with $n_{\mathtt{x}} = n_{\mathtt{xl}} + n_{\mathtt{xnl}}$ and

$$\mathrm{H}(\mathbf{x}) = \mathrm{H}_{\mathtt{l}}(\mathbf{x}_{\mathtt{l}}) + \mathrm{H}_{\mathtt{nl}}(\mathbf{x}_{\mathtt{nl}})$$

.

The set of dissipative variables is split according to $\mathbf{w} = (\mathbf{x}_{\mathtt{l}}^{\intercal}, \mathbf{w}_{\mathtt{nl}}^{\intercal})^{\intercal}$ with

$\mathbf{w}_{\mathtt{l}} = (w_1, \cdots, w_{n_{\mathtt{wl}}})^\intercal$ the variables associated with the linear components of the dissipative relation $\mathbf{z}_{\mathtt{l}}(\mathbf{w}_{\mathtt{l}}) = \mathbf{Z}_{\mathtt{l}}\,\mathbf{w}_{\mathtt{l}}$

$\mathbf{w}_{\mathtt{nl}} = (w_{n_{\mathtt{wl}}+1}, \cdots, w_{n_{\mathtt{w}}})^\intercal$ the variables associated with the nonlinear components of the dissipative relation $\mathbf{z}_{\mathtt{nl}} : \mathbf{w}_{\mathtt{nl}} \mapsto \mathbf{z}_{\mathtt{nl}}(\mathbf{w}_{\mathtt{nl}}) \in \mathbb{R}^{n_{\mathtt{wnl}}}$ with $n_{\mathtt{w}} = n_{\mathtt{wl}} + n_{\mathtt{wnl}}$ and

$$\mathbf{z}(\mathbf{w}) = \begin{pmatrix} \mathbf{Z}_{\mathtt{l}}\,\mathbf{w}_{\mathtt{l}} \\ \mathbf{z}_{\mathtt{nl}}(\mathbf{w}_{\mathtt{nl}}) \end{pmatrix}.$$

Accordingly, the structure matrices are split as

$$\underbrace{\left(\begin{array}{c} \frac{\mathrm{d}\mathbf{x}_{\mathtt{l}}}{\mathrm{d}t} \\ \frac{\mathrm{d}\mathbf{x}_{\mathtt{nl}}}{\mathrm{d}t} \\ \hline \mathbf{w}_{\mathtt{l}} \\ \mathbf{w}_{\mathtt{nl}} \\ \hline \mathbf{y} \end{array}\right)}_{\mathbf{b}} = \underbrace{\left(\begin{array}{cc|cc|c} \mathbf{M}_{\mathtt{xlxl}} & \mathbf{M}_{\mathtt{xlxnl}} & \mathbf{M}_{\mathtt{xlwl}} & \mathbf{M}_{\mathtt{xlwnl}} & \mathbf{M}_{\mathtt{xly}} \\ \mathbf{M}_{\mathtt{xnlxl}} & \mathbf{M}_{\mathtt{xnlxnl}} & \mathbf{M}_{\mathtt{xnlwl}} & \mathbf{M}_{\mathtt{xnlwnl}} & \mathbf{M}_{\mathtt{xnly}} \\ \hline \mathbf{M}_{\mathtt{wlxl}} & \mathbf{M}_{\mathtt{wlxnl}} & \mathbf{M}_{\mathtt{wlwl}} & \mathbf{M}_{\mathtt{wlwnl}} & \mathbf{M}_{\mathtt{wly}} \\ \mathbf{M}_{\mathtt{wnlwl}} & \mathbf{M}_{\mathtt{wnlxnl}} & \mathbf{M}_{\mathtt{wnlwl}} & \mathbf{M}_{\mathtt{wnlwnl}} & \mathbf{M}_{\mathtt{wnly}} \\ \hline \mathbf{M}_{\mathtt{yxl}} & \mathbf{M}_{\mathtt{yxnl}} & \mathbf{M}_{\mathtt{ywl}} & \mathbf{M}_{\mathtt{ywnl}} & \mathbf{M}_{\mathtt{yy}} \end{array}\right)}_{\mathbf{M}} \cdot \underbrace{\left(\begin{array}{c} \mathbf{Q}\cdot\mathbf{x} \\ \nabla \mathrm{H}_{\mathtt{nl}}(\mathbf{x}_{\mathtt{nl}}) \\ \hline \mathbf{Z}_{\mathtt{l}}\cdot\mathbf{w}_{\mathtt{l}} \\ \mathbf{z}_{\mathtt{nl}}(\mathbf{w}_{\mathtt{nl}}) \\ \hline \mathbf{u} \end{array}\right)}_{\mathbf{a}} \tag{3}$$

Table des matières

2 Structure of the `pyphs.PortHamiltonianObject`

Below is a list of each module of practical use in the object `pyphs.PortHamiltonianObject`, along with a short description. We consider the following instantiation :

```python
# import of (pre-installed) pyphs package:
import pyphs

# instantiate the PortHamiltonianObject:
phs = pyphs.PortHamiltonianObject(label='mylabel')
```

2.1 The symbs module

Container for all the SymPy symbolic variables (`sympy.Symbol`).

Attributes are ordered *list of symbols* associated with the system's vectors components.

`phs.symbs.x` : state vector symbols $\mathbf{x} \in \mathbb{R}^{n_x}$,

`phs.symbs.w` : dissipative vector variable symbols $\mathbf{w} \in \mathbb{R}^{n_w}$,

`phs.symbs.u` : input vector symbols $\mathbf{u} \in \mathbb{R}^{n_y}$,

`phs.symbs.y` : output vector symbols $\mathbf{y} \in \mathbb{R}^{n_y}$,

`phs.symbs.cu` : input vector symbols for connectors $\mathbf{c_u} \in \mathbb{R}^{n_y}$,

`phs.symbs.cy` : output vector symbols for connectors $\mathbf{c_y} \in \mathbb{R}^{n_y}$,

`phs.symbs.p` : Time-varying parameters symbols $\mathbf{p} \in \mathbb{R}^{n_y}$.

Methods :

`phs.symbs.dx()` : Returns the symbols associated with the state differential $\mathrm{d}\mathbf{x}$ formed by appending the prefix d to each symbol in `x`.

`phs.symbs.args()` : Return the list of symbols associated with the vector of all arguments of the symbolic expressions (`expr` module).

2.2 The exprs module

Container for all the SymPy symbolic expressions `sympy.Expr`associated with the system's functions.

Attributes : For scalar function (e.g. the storage function H), arguments of `phs.exprs` are SymPy expressions (`sympy.Expr`) ; for vector functions (e.g. the disipative function $\mathbf{z}$), arguments are ordered lists of SymPy expressions ; for matrix functions (e.g. the Jacobian matrix of disipative function $\mathbf{z}$), arguments are `sympy.Matrix` objects. Notice the expressions arguments [1] must belong either to (i) the elements of `phs.symbs.args()`, or (ii) the keys of the dictionary `phs.symbs.subs`.

`phs.exprs.H` : storage function $\mathrm{H} \in \mathbb{R}$,

`phs.exprs.z` : dissipative function $\mathbf{z} \in \mathbb{R}^{n_w}$,

`phs.exprs.g` : input/output gains vector function $\mathbf{g} \in \mathbb{R}^{n_g}$,

The following expression are computed from the `exprs.build()` method (see below) :

`phs.exprs.dxH` : the continuous gradient vector of storage scalar function $\nabla\mathrm{H}(\mathbf{x}) \in \mathbb{R}^{n_x}$,

`phs.exprs.dxHd` : the discrete gradient vector of storage scalar function $\overline{\nabla}\mathrm{H}(\mathbf{x}, \delta\mathbf{x}) \in \mathbb{R}^{n_x}$,

`phs.exprs.hessH` : the continuous hessian matrix of storage scalar function (computed as $\nabla\nabla\mathrm{H}(\mathbf{x}) \in \mathbb{R}^{n_x \times n_x}$),

`phs.exprs.jacz` : the continuous jacobian matrix of dissipative vector function $\nabla\mathbf{z}(\mathbf{w}) \in \mathbb{R}^{n_w \times n_w}$.

1. Accessed through the `sympy.Expr.free_symbols` (*e.g.* `phs.exprs.H.free_symbols` to recover the arguments of the Storage function H).

`phs.exprs.y` : the expression of the continuous output vector function $\mathbf{y}(\nabla \mathrm{H}, \mathbf{z}, \mathbf{u}) \in \mathbb{R}^{n_{\mathrm{y}}}$,

`phs.exprs.yd` : the expression of the discrete output vector function $\overline{\mathbf{y}}(\overline{\nabla} \mathrm{H}, \mathbf{z}, \mathbf{u}) \in \mathbb{R}^{n_{\mathrm{y}}}$,

Methods :

`phs.exprs.build()` : Build the following system functions as SymPy expressions and append them as attributes to the `phs.exprs` module : `phs.exprs.dxH`, `phs.exprs.dxHd`, `phs.exprs.hessH`, `phs.exprs.jacz`, `phs.exprs.y`, and `phs.exprs.yd`.

`phs.exprs.setexpr(name, expr)` : Add the SymPy expression `expr` to the `phs.exprs` module, with argument `name`, and add `name` to the set of `phs.exprs._names`.

`phs.exprs.freesymbols()` : Return a python set of all the free symbols (`sympy.Symbol`) that appear at least once in all expressions with names in `phs.exprs._names`.

2.3 The `dims` module

Container for accessors to the system's dimensions. No attributes should be changed manually. To split the system into its linear and nonlinear part, use `phs.split_linear()` which organize the system vectors as

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{\mathtt{l}} \\ \mathbf{x}_{\mathtt{nl}} \end{pmatrix}, \quad \dim(\mathbf{x}_{\mathtt{l}}) = \tag{4}$$

Attributes : `phs.dims.xl` : Number of state vector components associated with a quadratic storage function : $\mathrm{H}_{\mathtt{l}}(\mathbf{x}_{\mathtt{l}}) = \mathbf{x}_{\mathtt{l}}^{\mathsf{T}} \cdot \frac{\mathbf{Q}}{2} \cdot \mathbf{x}_{\mathtt{l}}$, and `phs.dims.x()` is equal to `phs.dims.xl + phs.dims.xnl()`.

`phs.dims.wl` : Number of dissipative vector variable components associated with a linear dissipative function : $\mathbf{z}_{\mathtt{l}}(\mathbf{w}_{\mathtt{l}}) = \mathbf{Z}_{\mathtt{l}} \cdot \mathbf{w}_{\mathtt{l}}$, and `phs.dims.w()` is equal to `phs.dims.wl + phs.dims.wnl()`.

Methods :

`phs.dims.x()` : Return the dimension of state vector `len(phs.symbs.x)`.

`phs.dims.xnl()` : Return the number of state vector components associated with a nonlinear storage function
`len(phs.symbs.x)`.

`setexpr(name, expr)` : Add the SymPy expression `expr` to the `exprs` module, with argument `name`, and add `name` to the set of `exprs._names`.

`freesymbols()` : Retrun a python set of all the free symbols (`sympy.symbols`) that appear at least once in all expressions with names in `exprs._names`.

3 Algorithms

This section details the algorithms actually implemented for

1. the graph analysis and

2. the different simulation methods

3.1 Graph analysis

The graph analysis method that derives the port-Hamiltonian system's differential-algebraic equations from with a given netlist is detailed in the reference [Falaize and Hélie, 2016]. The algorithm implemented in PyPHS is exactly that in [Falaize and Hélie, 2016, algorithm 1].

3.2 Simulation methods

The discrete gradient method is used in conjunction with the port-Hamiltonian structure to produce a passive-guaranteed numerical scheme (see [Falaize and Hélie, 2016] for details). In the sequel, quantities are defined on the current time step $\mathbf{x} \equiv \mathbf{x}(t_k)$, with $k \in \mathbb{N}_+^*$.

3.2.1 Split of the linear part from the nonlinear part

The dicrete gradient for the quadratic part of the Hamiltonian is $\nabla \mathrm{H}_{\mathtt{l}} = \frac{1}{2}\,\mathbf{Q}\,(2\mathbf{x}_{\mathtt{l}} + \delta\mathbf{x}_{\mathtt{l}})$ and the discret linear subsystem is

$$
\begin{aligned}
\mathbf{D}_{\mathtt{l}}^{-1} = \mathbf{iD}_{\mathtt{l}} \quad &= \quad \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} - \begin{pmatrix} \mathbf{M}_{\mathtt{xlxl}} & \mathbf{M}_{\mathtt{xlwl}} \\ \mathbf{M}_{\mathtt{wlxl}} & \mathbf{M}_{\mathtt{wlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z}_{\mathtt{l}} \end{pmatrix}, \\
\underbrace{\begin{pmatrix} \delta\mathbf{x}_{\mathtt{l}} \\ \mathbf{w}_{\mathtt{l}} \end{pmatrix}}_{\mathbf{v}_{\mathtt{l}}} \quad &= \quad \underbrace{\mathbf{D}_{\mathtt{l}} \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xlxl}} \\ \mathbf{M}_{\mathtt{wlxl}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{lxl}}}}}_{\mathbf{N}_{\mathtt{lxl}}} \mathbf{Q}\;\mathbf{x}_{\mathtt{l}} + \underbrace{\mathbf{D}_{\mathtt{l}} \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xlxnl}} & \mathbf{M}_{\mathtt{xlwnl}} \\ \mathbf{M}_{\mathtt{wlxnl}} & \mathbf{M}_{\mathtt{wlwnl}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{lnl}}}}}_{\mathbf{N}_{\mathtt{lnl}}} \underbrace{\begin{pmatrix} \nabla\mathrm{H}_{\mathtt{nl}} \\ \mathbf{z}_{\mathtt{nl}} \end{pmatrix}}_{\mathbf{f}_{\mathtt{nl}}} + \underbrace{\mathbf{D}_{\mathtt{l}} \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xly}} \\ \mathbf{M}_{\mathtt{wly}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{ly}}}}}_{\mathbf{N}_{\mathtt{ly}}} \mathbf{u}
\end{aligned} \tag{5}
$$

and the nonlinear subsystem is

$$
\begin{aligned}
\begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} \underbrace{\begin{pmatrix} \delta\mathbf{x}_{\mathtt{nl}} \\ \mathbf{w}_{\mathtt{nl}} \end{pmatrix}}_{\mathbf{v}_{\mathtt{nl}}} \quad = \quad & \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xnlxnl}} & \mathbf{M}_{\mathtt{xnlwnl}} \\ \mathbf{M}_{\mathtt{wnlxnl}} & \mathbf{M}_{\mathtt{wnlwnl}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{nlnl}}}} \mathbf{f}_{\mathtt{nl}} + \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xnly}} \\ \mathbf{M}_{\mathtt{wnly}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{nly}}}} \mathbf{u} \\
& + \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xnlxl}} & \mathbf{M}_{\mathtt{xnlwl}} \\ \mathbf{M}_{\mathtt{wnlxl}} & \mathbf{M}_{\mathtt{wnlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z}_{\mathtt{l}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{nll}}}} \mathbf{v}_{\mathtt{l}} + \underbrace{\begin{pmatrix} \mathbf{M}_{\mathtt{xnlxl}} \\ \mathbf{M}_{\mathtt{wnlxl}} \end{pmatrix}}_{\overline{\mathbf{N}_{\mathtt{nlxl}}}} \mathbf{Q}\;\mathbf{x}_{\mathtt{l}}
\end{aligned} \tag{6}
$$

3.2.2 Presolve numerical nonlinear subsystem

$$
\begin{aligned}
\begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} \mathbf{v}_{\mathtt{nl}} \quad = \quad & \underbrace{\left(\overline{\mathbf{N}_{\mathtt{nlxl}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\,\mathbf{N}_{\mathtt{lxl}}\right)}_{\mathbf{N}_{\mathtt{nlxl}}} \mathbf{x}_{\mathtt{l}} + \underbrace{\left(\overline{\mathbf{N}_{\mathtt{nlnl}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\,\mathbf{N}_{\mathtt{lnl}}\right)}_{\mathbf{N}_{\mathtt{nlnl}}} \mathbf{f}_{\mathtt{nl}} \\
& \underbrace{\left(\overline{\mathbf{N}_{\mathtt{nly}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\,\mathbf{N}_{\mathtt{ly}}\right)}_{\mathbf{N}_{\mathtt{nly}}} \mathbf{u}
\end{aligned} \tag{7}
$$

3.2.3 Algorithm

Inputs

$$
\begin{aligned}
\mathbf{iD}_{\mathtt{l}} &= \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} - \begin{pmatrix} \mathbf{M}_{\mathtt{xlxl}} & \mathbf{M}_{\mathtt{xlwl}} \\ \mathbf{M}_{\mathtt{wlxl}} & \mathbf{M}_{\mathtt{wlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z}_{\mathtt{l}} \end{pmatrix} \\
\overline{\mathbf{N}_{\mathtt{lxl}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xlxl}} \\ \mathbf{M}_{\mathtt{wlxl}} \end{pmatrix} \mathbf{Q} \\
\overline{\mathbf{N}_{\mathtt{lnl}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xlxnl}} & \mathbf{M}_{\mathtt{xlwnl}} \\ \mathbf{M}_{\mathtt{wlxnl}} & \mathbf{M}_{\mathtt{wlwnl}} \end{pmatrix} \\
\overline{\mathbf{N}_{\mathtt{ly}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xly}} \\ \mathbf{M}_{\mathtt{wly}} \end{pmatrix} \\
\overline{\mathbf{N}_{\mathtt{nlnl}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xnlxnl}} & \mathbf{M}_{\mathtt{xnlwnl}} \\ \mathbf{M}_{\mathtt{wnlxnl}} & \mathbf{M}_{\mathtt{wnlwnl}} \end{pmatrix} \\
\overline{\mathbf{N}_{\mathtt{nll}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xnlxl}} & \mathbf{M}_{\mathtt{xnlwl}} \\ \mathbf{M}_{\mathtt{wnlxl}} & \mathbf{M}_{\mathtt{wnlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z}_{\mathtt{l}} \end{pmatrix} \\
\overline{\mathbf{N}_{\mathtt{nlxl}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xnlxl}} \\ \mathbf{M}_{\mathtt{wnlxl}} \end{pmatrix} \mathbf{Q} \\
\overline{\mathbf{N}_{\mathtt{nly}}} &= \begin{pmatrix} \mathbf{M}_{\mathtt{xnly}} \\ \mathbf{M}_{\mathtt{wnly}} \end{pmatrix} \\
\mathcal{J}_{\mathbf{fnl}}(\mathbf{v}_{\mathtt{nl}}) &= \begin{pmatrix} \mathcal{J}_{\nabla \mathrm{H}_{\mathtt{nl}}} & 0 \\ 0 & \mathcal{J}_{\mathbf{z}_{\mathtt{nl}}} \end{pmatrix} \\
\mathbf{I}_{\mathtt{nl}} &= \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix}
\end{aligned} \tag{8}
$$

Process

$$
\begin{aligned}
\mathbf{D}_{\mathtt{l}} &= \mathbf{iD}_{\mathtt{l}}^{-1} \\
\mathbf{N}_{\mathtt{lxl}} &= \mathbf{D}_{\mathtt{l}}\, \overline{\mathbf{N}_{\mathtt{lxl}}} \\
\mathbf{N}_{\mathtt{lnl}} &= \mathbf{D}_{\mathtt{l}}\, \overline{\mathbf{N}_{\mathtt{lnl}}} \\
\mathbf{N}_{\mathtt{ly}} &= \mathbf{D}_{\mathtt{l}}\, \overline{\mathbf{N}_{\mathtt{ly}}} \\
\mathbf{N}_{\mathtt{nlxl}} &= \overline{\mathbf{N}_{\mathtt{nlxl}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\, \mathbf{N}_{\mathtt{lxl}} \\
\mathbf{N}_{\mathtt{nlnl}} &= \overline{\mathbf{N}_{\mathtt{nlnl}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\, \mathbf{N}_{\mathtt{lnl}} \\
\mathbf{N}_{\mathtt{nly}} &= \overline{\mathbf{N}_{\mathtt{nly}}} + \overline{\mathbf{N}_{\mathtt{nll}}}\, \mathbf{N}_{\mathtt{ly}} \\
\mathbf{c} &= \mathbf{N}_{\mathtt{nlxl}}\, \mathbf{x}_{\mathtt{l}} + \mathbf{N}_{\mathtt{nly}}\, \mathbf{u} \\
\text{Iterate} &: \mathbf{F}_{\mathtt{nl}}(\mathbf{v}_{\mathtt{nl}}) = \mathbf{I}_{\mathtt{nl}}\, \mathbf{v}_{\mathtt{nl}} - \mathbf{N}_{\mathtt{nlnl}}\, \mathbf{f}_{\mathtt{nl}} - \mathbf{c} \\
&\ \ \mathcal{J}_{\mathbf{F}_{\mathtt{nl}}}(\mathbf{v}_{\mathtt{nl}}) = \mathbf{I}_{\mathtt{nl}} - \mathbf{N}_{\mathtt{nlnl}}\, \mathcal{J}_{\mathbf{fnl}}(\mathbf{v}_{\mathtt{nl}}) \\
&\ \ \mathbf{v}_{\mathtt{nl}} = \mathbf{v}_{\mathtt{nl}} - \mathcal{J}_{\mathbf{F}_{\mathtt{nl}}}^{-1}(\mathbf{v}_{\mathtt{nl}})\, \mathbf{F}_{\mathtt{nl}}(\mathbf{v}_{\mathtt{nl}}) \\
\mathbf{v}_{\mathtt{l}} &= \mathbf{N}_{\mathtt{lxl}}\, \mathbf{x}_{\mathtt{l}} + \mathbf{N}_{\mathtt{lnl}}\, \mathbf{f}_{\mathtt{nl}} + \mathbf{N}_{\mathtt{ly}}\, \mathbf{u} \\
\mathbf{y} &= \mathbf{M}_{\mathrm{yxl}}\, \nabla \mathrm{H}_{\mathtt{l}} + \mathbf{M}_{\mathrm{yxnl}}\, \nabla \mathrm{H}_{\mathtt{nl}} \mathbf{M}_{\mathrm{ywl}}\, \mathbf{Z}_{\mathtt{l}}\, \mathbf{w}_{\mathtt{l}} + \mathbf{M}_{\mathrm{ywnl}}\, \mathbf{z}_{\mathtt{nl}} + \mathbf{M}_{\mathrm{yy}}\, \mathbf{u} \\
\mathbf{x} &= \mathbf{x} + \delta \mathbf{x}
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
\mathbf{y} &= \mathbf{M}_{\mathrm{yxl}}\, \nabla \mathrm{H}_{\mathtt{l}} + \mathbf{M}_{\mathrm{yxnl}}\, \nabla \mathrm{H}_{\mathtt{nl}} \mathbf{M}_{\mathrm{ywl}}\, \mathbf{Z}_{\mathtt{l}}\, \mathbf{w}_{\mathtt{l}} + \mathbf{M}_{\mathrm{ywnl}}\, \mathbf{z}_{\mathtt{nl}} + \mathbf{M}_{\mathrm{yy}}\, \mathbf{u} && (10) \\
&= \mathbf{M}_{\mathrm{yxl}}\, \nabla \mathrm{H}_{\mathtt{l}} + \mathbf{M}_{\mathrm{yxnl}}\, \nabla \mathrm{H}_{\mathtt{nl}} \mathbf{M}_{\mathrm{ywl}}\, \mathbf{Z}_{\mathtt{l}}\, \mathbf{w}_{\mathtt{l}} + \mathbf{M}_{\mathrm{ywnl}}\, \mathbf{z}_{\mathtt{nl}} + \mathbf{M}_{\mathrm{yy}}\, \mathbf{u} && (11) \\
& && (12)
\end{aligned}
$$

3.3 Realizability solver

Connections

Serial $\sum_{n=1}^{N} e_n = 0$,
$f_1 = \cdots = f_N = \phi$ (variable commune)

parallel $\sum_{n=1}^{N} f_n = 0$,
$e_1 = \cdots = e_N = \phi$ (variable commune)
Storage

Realizable

$$\begin{cases} \phi = u = \frac{\mathrm{d}}{\mathrm{d}t}x & = & \frac{\mathrm{d}x_1}{\mathrm{d}t} = \cdots = \frac{\mathrm{d}}{\mathrm{d}t}x_N \\ y = \nabla \mathrm{H}_(x) & = & \sum_{i=1}^{N} \nabla \mathrm{H} i(x_i) \end{cases}$$

alors $x = x_1 = x_2$ et $\mathrm{H}(x) = \left(\sum_{i=1}^{N} \mathrm{H}_i\right)(x)$

Non-Realizable

$$\begin{cases} \phi = u = \nabla \mathrm{H}_(x) & = & \nabla \mathrm{H}1(x_1) = \cdots = \nabla \mathrm{H}_N(x_N), \\ y = \frac{\mathrm{d}}{\mathrm{d}t}x & = & \sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}t}x_i \end{cases}$$

alors $x = \sum_{i=1}^{N} x_i$ et $\mathrm{H}(x) = \left(\sum_{i=1}^{N} H_i \nabla \mathrm{H}_i^{-1} G\right)(x)$
avec $G^{-1}(x) = \sum_{i=1}^{N} \nabla \mathrm{H}_i(x_i)$
Dissipatives

Realizable

$$\begin{cases} \phi = u = w & = & w_1 = \cdots = w_N, \\ y = z(w) & = & \sum_{i=1}^{N} z_i(x_i) \end{cases}$$

Non-Realizable

$$\begin{cases} \phi = u = z(w) & = & z_1(w_1) = \cdots = z_N(w_N), \\ y = w & = & \sum_{i=1}^{N} w_i \end{cases}$$

alors $w = \sum_{i=1}^{N} w_i = \left(\sum_{i=1}^{N} z_i^{-1}\right)(\phi)$ et $z^{-1}(\phi) = w \Rightarrow z(w) = \left(\sum_{i=1}^{N} z_i\right)^{-1}(w)$

Références

[Falaize and Hélie, 2016] Falaize, A. and Hélie, T. (2016). Passive guaranteed simulation of analog audio circuits : A port-hamiltonian approach. *Applied Sciences*, 6(10) :273.