PyPHS Version 0.2
Documentation

Antoine Falaize

28 janvier 2017

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

1.1.1 Using `pip`

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

```
pip install pyphs
```

1.1.2 Using `setuptools`

1. Download the sources from the package repository
   `https://github.com/afalaize/pyphs`

2. In a terminal, navigate to the folder that contains the `setup.py` script

3. Install with :

   ```
   python setup.py install
   ```

4. To run tests :

   ```
   python setup.py test
   ```

1.1.3 Using `Anaconda` (Mac OSX only)

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 academic reference [?]. We consider systems that can be described by the following time-continuous non-linear state-space representation :

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

where

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

and

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

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

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

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

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

$$P_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{z}(\mathbf{w}) + \mathbf{a} \cdot \mathbf{R} \cdot \mathbf{a},$$

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

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

The state is split according to $\mathbf{x} = (\mathbf{x}_1^{\mathsf{T}}, \mathbf{x}_{nl}^{\mathsf{T}})^{\mathsf{T}}$ with

$\mathbf{x}_1 = (x_1, \cdots, x_{n_{x1}})^{\mathsf{T}}$ the states associated with the quadratic components of the storage function $H_1(\mathbf{x}_1) = \frac{\mathbf{x}_1 \cdot \mathbf{Q} \cdot \mathbf{x}_1}{2}$

$\mathbf{x_{nl}} = (x_{n_{x1}+1}, \cdots, x_{n_x})^{\mathsf{T}}$ the states associated with the non-quadratic components of the storage function with $n_{\mathtt{x}} = n_{\mathtt{x1}} + n_{\mathtt{xnl}}$ and

$$\mathrm{H}(\mathbf{x}) = \mathrm{H_1}(\mathbf{x_1}) + \mathrm{H_{nl}}(\mathbf{x_{nl}})$$

.

The set of dissipative variables is split according to $\mathbf{w} = (\mathbf{x_1^{\mathsf{T}}}, \mathbf{w_{nl}^{\mathsf{T}}})^{\mathsf{T}}$ with

$\mathbf{w_1} = (w_1, \cdots, w_{n_{w1}})^{\mathsf{T}}$ the variables associated with the linear components of the dissipative relation $\mathbf{z_1}(\mathbf{w_1}) = \mathbf{Z_1}\,\mathbf{w_1}$

$\mathbf{w_{nl}} = (w_{n_{w1}+1}, \cdots, w_{n_w})^{\mathsf{T}}$ the variables associated with the nonlinear components of the dissipative relation $\mathbf{z_{nl}} : \mathbf{w_{nl}} \mapsto \mathbf{z_{nl}}(\mathbf{w_{nl}}) \in \mathbb{R}^{n_{wnl}}$ with $n_{\mathtt{w}} = n_{\mathtt{w1}} + n_{\mathtt{wnl}}$ and

$$\mathbf{z}(\mathbf{w}) = \begin{pmatrix} \mathbf{Z_1}\,\mathbf{w_1} \\ \mathbf{z_{nl}}(\mathbf{w_{nl}}) \end{pmatrix}.$$

Accordingly, the structure matrices are split as

$$\underbrace{\begin{pmatrix} \frac{\mathrm{d}\mathbf{x_1}}{\mathrm{d}t} \\ \frac{\mathrm{d}\mathbf{x_{nl}}}{\mathrm{d}t} \\ \hline \mathbf{w_1} \\ \mathbf{w_{nl}} \\ \hline \mathbf{y} \end{pmatrix}}_{\mathbf{b}} = \underbrace{\left(\begin{array}{cc|cc|c} \mathbf{M_{x1x1}} & \mathbf{M_{x1xnl}} & \mathbf{M_{x1w1}} & \mathbf{M_{x1wnl}} & \mathbf{M_{x1y}} \\ \mathbf{M_{xnlx1}} & \mathbf{M_{xnlxnl}} & \mathbf{M_{xnlw1}} & \mathbf{M_{xnlwnl}} & \mathbf{M_{xnly}} \\ \hline \mathbf{M_{w1x1}} & \mathbf{M_{w1xnl}} & \mathbf{M_{w1w1}} & \mathbf{M_{w1wnl}} & \mathbf{M_{w1y}} \\ \mathbf{M_{wnlw1}} & \mathbf{M_{wnlxnl}} & \mathbf{M_{wnlw1}} & \mathbf{M_{wnlwnl}} & \mathbf{M_{wnly}} \\ \hline \mathbf{M_{yx1}} & \mathbf{M_{yxnl}} & \mathbf{M_{yw1}} & \mathbf{M_{ywnl}} & \mathbf{M_{yy}} \end{array} \right)}_{\mathbf{M}} \cdot \underbrace{\begin{pmatrix} \mathbf{Q} \cdot \mathbf{x} \\ \nabla \mathrm{H_{nl}}(\mathbf{x_{nl}}) \\ \hline \mathbf{Z_1} \cdot \mathbf{w_1} \\ \mathbf{z_{nl}}(\mathbf{w_{nl}}) \\ \hline \mathbf{u} \end{pmatrix}}_{\mathbf{a}}$$

$$(3)$$

Table des matières

2 Overview

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

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

```python
# 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 $d\mathbf{x}$ formed by appending the prefix d to each symbol in $\mathbf{x}$.

`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 $H \in \mathbb{R}$,

`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 $\nabla H(\mathbf{x}) \in \mathbb{R}^{n_x}$,

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

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

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

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

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

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`.

---

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.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

$$\mathbf{x} = \left(\begin{array}{c} \mathbf{x_l} \\ \mathbf{x_{nl}} \end{array} \right), \quad \dim(\mathbf{x_l}) = \tag{4}$$

Attributes : `phs.dims.xl` : Number of state vector components associated with a quadratic storage function : $H_1(\mathbf{x_1}) = \mathbf{x_1^\mathsf{T}} \cdot \frac{\mathbf{Q}}{2} \cdot \mathbf{x_1}$, 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 : $\mathbf{z_1}(\mathbf{w_1}) = \mathbf{Z_1} \cdot \mathbf{w_1}$, 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 Recall on Port-Hamiltonian Systems

3.1 PHSCORE

3.1.1 Attributes

- •

3.1.2 Methods

4 Finite dimensionnal dynamical network

4.1 PHSNETLIST

4.2 PHSGRAPH

5 Numerical Methods

5.1 PHSNumericalMethod

5.2 PHSNumericalMethodStandard

A Algorithms

This section details the algorithms actually implemented for

1. the graph analysis and

2. the different simulation methods

A.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 [?]. The algorithm implemented in PyPHS is exactly that in [?, algorithm 1].

A.2 Simulation methods

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

A.2.1 Split of the linear part from the nonlinear part

The dicrete gradient for the quadratic part of the Hamiltonian is $\nabla H_l = \frac{1}{2} \mathbf{Q} \left(2\mathbf{x}_l + \delta\mathbf{x}_l \right)$ and the discret linear subsystem is

$$
\mathbf{D}_l^{-1} = \mathbf{iD}_l = \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} - \begin{pmatrix} \mathbf{M}_{xlxl} & \mathbf{M}_{xlwl} \\ \mathbf{M}_{wlxl} & \mathbf{M}_{wlwl} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z}_l \end{pmatrix},
$$

$$
\underbrace{\begin{pmatrix} \delta\mathbf{x}_l \\ \mathbf{w}_l \end{pmatrix}}_{\mathbf{v}_l} = \mathbf{D}_l \underbrace{\underbrace{\begin{pmatrix} \mathbf{M}_{xlxl} \\ \mathbf{M}_{wlxl} \end{pmatrix}}_{\overline{\mathbf{N}}_{lxl}} \mathbf{Q}}_{\mathbf{N}_{lxl}} \mathbf{x}_l + \mathbf{D}_l \underbrace{\underbrace{\begin{pmatrix} \mathbf{M}_{xlxnl} & \mathbf{M}_{xlwnl} \\ \mathbf{M}_{wlxnl} & \mathbf{M}_{wlwnl} \end{pmatrix}}_{\overline{\mathbf{N}}_{lnl}}}_{\mathbf{N}_{lnl}} \underbrace{\begin{pmatrix} \nabla H_{nl} \\ \mathbf{z}_{nl} \end{pmatrix}}_{\mathbf{f}_{nl}} + \mathbf{D}_l \underbrace{\underbrace{\begin{pmatrix} \mathbf{M}_{xly} \\ \mathbf{M}_{wly} \end{pmatrix}}_{\overline{\mathbf{N}}_{ly}}}_{\mathbf{N}_{ly}} \mathbf{u}
$$

$$(5)$$

and the nonlinear subsystem is

$$
\begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} \underbrace{\begin{pmatrix} \delta\mathbf{x_{nl}} \\ \mathbf{w_{nl}} \end{pmatrix}}_{\mathbf{v_{nl}}} = \underbrace{\begin{pmatrix} \mathbf{M_{xnlxnl}} & \mathbf{M_{xnlwnl}} \\ \mathbf{M_{wnlxnl}} & \mathbf{M_{wnlwnl}} \end{pmatrix}}_{\overline{\mathbf{N_{nlnl}}}} \mathbf{f_{nl}} + \underbrace{\begin{pmatrix} \mathbf{M_{xnly}} \\ \mathbf{M_{wnly}} \end{pmatrix}}_{\overline{\mathbf{N_{nly}}}} \mathbf{u}
$$
$$
+ \underbrace{\begin{pmatrix} \mathbf{M_{xnlxl}} & \mathbf{M_{xnlwl}} \\ \mathbf{M_{wnlxl}} & \mathbf{M_{wnlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z_l} \end{pmatrix}}_{\overline{\mathbf{N_{nll}}}} \mathbf{v_l} + \underbrace{\begin{pmatrix} \mathbf{M_{xnlxl}} \\ \mathbf{M_{wnlxl}} \end{pmatrix} \mathbf{Q}}_{\overline{\mathbf{N_{nlxl}}}} \mathbf{x_l}
$$

$$(6)$$

A.2.2 Presolve numerical nonlinear subsystem

$$
\begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} \mathbf{v_{nl}} = \underbrace{\left(\overline{\mathbf{N_{nlxl}}} + \overline{\mathbf{N_{nll}}}\,\mathbf{N_{lxl}} \right)}_{\mathbf{N_{nlxl}}} \mathbf{x_l} + \underbrace{\left(\overline{\mathbf{N_{nlnl}}} + \overline{\mathbf{N_{nll}}}\,\mathbf{N_{lnl}} \right)}_{\mathbf{N_{nlnl}}} \mathbf{f_{nl}}
$$
$$
\underbrace{\left(\overline{\mathbf{N_{nly}}} + \overline{\mathbf{N_{nll}}}\,\mathbf{N_{ly}} \right)}_{\mathbf{N_{nly}}} \mathbf{u}
$$

$$(7)$$

A.2.3 Algorithm

Inputs

$$
\begin{aligned}
\mathbf{iD_l} &= \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix} - \begin{pmatrix} \mathbf{M_{xlxl}} & \mathbf{M_{xlwl}} \\ \mathbf{M_{wlxl}} & \mathbf{M_{wlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z_l} \end{pmatrix} \\
\overline{\mathbf{N_{lxl}}} &= \begin{pmatrix} \mathbf{M_{xlxl}} \\ \mathbf{M_{wlxl}} \end{pmatrix} \mathbf{Q} \\
\overline{\mathbf{N_{lnl}}} &= \begin{pmatrix} \mathbf{M_{xlxnl}} & \mathbf{M_{xlwnl}} \\ \mathbf{M_{wlxnl}} & \mathbf{M_{wlwnl}} \end{pmatrix} \\
\overline{\mathbf{N_{ly}}} &= \begin{pmatrix} \mathbf{M_{xly}} \\ \mathbf{M_{wly}} \end{pmatrix} \\
\overline{\mathbf{N_{nlnl}}} &= \begin{pmatrix} \mathbf{M_{xnlxnl}} & \mathbf{M_{xnlwnl}} \\ \mathbf{M_{wnlxnl}} & \mathbf{M_{wnlwnl}} \end{pmatrix} \\
\overline{\mathbf{N_{nll}}} &= \begin{pmatrix} \mathbf{M_{xnlxl}} & \mathbf{M_{xnlwl}} \\ \mathbf{M_{wnlxl}} & \mathbf{M_{wnlwl}} \end{pmatrix} \begin{pmatrix} \frac{1}{2}\mathbf{Q} & 0 \\ 0 & \mathbf{Z_l} \end{pmatrix} \\
\overline{\mathbf{N_{nlxl}}} &= \begin{pmatrix} \mathbf{M_{xnlxl}} \\ \mathbf{M_{wnlxl}} \end{pmatrix} \mathbf{Q} \\
\overline{\mathbf{N_{nly}}} &= \begin{pmatrix} \mathbf{M_{xnly}} \\ \mathbf{M_{wnly}} \end{pmatrix} \\
\mathcal{J}_{\mathbf{fnl}}(\mathbf{v_{nl}}) &= \begin{pmatrix} \mathcal{J}_{\nabla H_{nl}} & 0 \\ 0 & \mathcal{J}_{\mathbf{z_{nl}}} \end{pmatrix} \\
\mathbf{I_{nl}} &= \begin{pmatrix} \frac{\mathbf{I_d}}{\delta t} & 0 \\ 0 & \mathbf{I_d} \end{pmatrix}
\end{aligned}
$$

$$(8)$$

Process

$$
\begin{aligned}
\mathbf{D_1} &= \mathbf{i}\overline{\mathbf{D_1^{-1}}} \\
\mathbf{N_{1x1}} &= \mathbf{D_1}\,\overline{\mathbf{N_{1x1}}} \\
\mathbf{N_{1n1}} &= \mathbf{D_1}\,\overline{\mathbf{N_{1n1}}} \\
\mathbf{N_{1y}} &= \mathbf{D_1}\,\overline{\mathbf{N_{1y}}} \\
\mathbf{N_{n1x1}} &= \overline{\mathbf{N_{n1x1}}} + \overline{\mathbf{N_{n11}}}\,\mathbf{N_{1x1}} \\
\mathbf{N_{n1n1}} &= \overline{\mathbf{N_{n1n1}}} + \overline{\mathbf{N_{n11}}}\,\mathbf{N_{1n1}} \\
\mathbf{N_{n1y}} &= \overline{\mathbf{N_{n1y}}} + \overline{\mathbf{N_{n11}}}\,\mathbf{N_{1y}} \\
\mathbf{c} &= \mathbf{N_{n1x1}}\,\mathbf{x_1} + \mathbf{N_{n1y}}\,\mathbf{u} \\
\text{Iterate} \quad : \quad & \mathbf{F_{n1}(v_{n1})} = \mathbf{I_{n1}}\,\mathbf{v_{n1}} - \mathbf{N_{n1n1}}\,\mathbf{f_{n1}} - \mathbf{c} \\
& \mathcal{J}_{\mathbf{F_{n1}}}(\mathbf{v_{n1}}) = \mathbf{I_{n1}} - \mathbf{N_{n1n1}}\,\mathcal{J}_{\mathbf{fn1}}(\mathbf{v_{n1}}) \\
& \mathbf{v_{n1}} = \mathbf{v_{n1}} - \mathcal{J}_{\mathbf{F_{n1}}}^{-1}(\mathbf{v_{n1}})\,\mathbf{F_{n1}(v_{n1})} \\
\mathbf{v_1} &= \mathbf{N_{1x1}}\,\mathbf{x_1} + \mathbf{N_{1n1}}\,\mathbf{f_{n1}} + \mathbf{N_{1y}}\,\mathbf{u} \\
\mathbf{y} &= \mathbf{M_{yx1}}\,\nabla\mathrm{H_1} + \mathbf{M_{yxn1}}\,\nabla\mathrm{H_{n1}}\mathbf{M_{yw1}}\,\mathbf{Z_1}\,\mathbf{w_1} + \mathbf{M_{ywn1}}\,\mathbf{z_{n1}} + \mathbf{M_{yy}}\,\mathbf{u} \\
\mathbf{x} &= \mathbf{x} + \delta\mathbf{x}
\end{aligned}
\tag{9}
$$

$$
\begin{aligned}
\mathbf{y} &= \mathbf{M_{yx1}}\,\nabla\mathrm{H_1} + \mathbf{M_{yxn1}}\,\nabla\mathrm{H_{n1}}\mathbf{M_{yw1}}\,\mathbf{Z_1}\,\mathbf{w_1} + \mathbf{M_{ywn1}}\,\mathbf{z_{n1}} + \mathbf{M_{yy}}\,\mathbf{u} \tag{10} \\
&= \mathbf{M_{yx1}}\,\nabla\mathrm{H_1} + \mathbf{M_{yxn1}}\,\nabla\mathrm{H_{n1}}\mathbf{M_{yw1}}\,\mathbf{Z_1}\,\mathbf{w_1} + \mathbf{M_{ywn1}}\,\mathbf{z_{n1}} + \mathbf{M_{yy}}\,\mathbf{u} \tag{11} \\
& \tag{12}
\end{aligned}
$$

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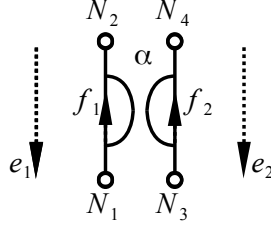avec $G^{-1}(x) = \sum_{i=1}^{N} \nabla\mathrm{H}_i(x_i)$

Dissipatives

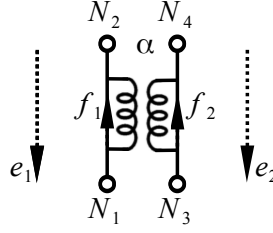FIGURE 1 – Scematic and conventions for the gyrator.

FIGURE 2 – Scematic and conventions for the transformer.

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

B Connectors : Gyrators and Transformers

A connector is set of two edges (e_1, e_2), the efforts and flux of which are coupled depending on the connector's type.

B.1 Gyrator

Constitutive relations

$$\begin{pmatrix} \mathfrak{e}_1 \\ \mathfrak{e}_2 \end{pmatrix} = \begin{pmatrix} 0 & -\alpha \\ +\alpha & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{f}_1 \\ \mathfrak{f}_2 \end{pmatrix}$$

B.2 Transformer

Constitutive relations

$$\begin{pmatrix} \mathfrak{e}_1 \\ \mathfrak{f}_2 \end{pmatrix} = \begin{pmatrix} 0 & -\alpha \\ +\alpha & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathfrak{f}_1 \\ \mathfrak{e}_2 \end{pmatrix}$$

C   Fractional calculus

The diffusive process in loudspeakers suspension (creep phenomenon **??**) and loudspeakers ferromagnetic path (eddy current phenomenon **??**) can be described by linear models that include fractional order dynamics (see [**?**, **?**, **?**, **?**] for fractional modeling of viscoelasticity, and [**?**, **?**, **?**] for fractional modeling of eddy currents). A well established formalism for the realization of fractional transfer functions is the so called *diffusive representations*, recalled thereafter (see detailed developements in [**?**, **?**], and [**?**] for a port-Hamiltonian formulation).

C.1   Fractional integrator

Defining $s = \rho.e^{i\theta}$, with $\rho \geq 0$ and $\theta \in [-\pi, \pi[$, the transfer function of the fractional integrator $\mathcal{I}_\beta(s) = s^{-\beta}$ exhibits a cut $\mathcal{C} = \mathbb{R}_-$. The residue theorem gives the realization of $\mathcal{I}_\beta$ as the continuous aggregation of linear damping along the cut $\mathcal{C}$. This leads to the following *diffusive representation* [**?**, §2] :

$$\begin{aligned} \mathcal{I}_\beta(s) : \mathbb{C} \setminus \mathbb{R}_- &\rightarrow \mathbb{C} \\ s &\mapsto \int_0^\infty \mu_\beta(\xi) \frac{1}{s+\xi} \mathrm{d}\xi \end{aligned} \tag{13}$$

where the weights $\mu_\beta(\xi) = \frac{\mathcal{I}_\beta(-\xi-i0^+) - \mathcal{I}_\beta(-\xi+i0^+)}{2i\pi} = \frac{sin(\beta\pi)}{\pi}\xi^{-\beta}$ correspond to the jump of $\mathcal{I}_\beta$ across $\mathcal{C} \equiv \{-\xi \in \mathbb{R}^-\}$). A state-space representation with output $y_\beta(s) = \mathcal{I}_\beta(s)u_\beta(s)$ is :

$$\begin{cases} \frac{\mathrm{d}x_\xi}{\mathrm{d}t} &= -\xi x_\xi + u_\beta, \quad x_\xi(0) = 0, \\ y_\beta &= \int_0^{+\infty} \mu_\beta(\xi) x_\xi \mathrm{d}\xi. \end{cases} \tag{14}$$

The system (**??**) is recast as an infinite dimensional pH system (**??**), defining the *hamiltonian density* $H_\xi(x_\xi) = \mu_\beta(\xi)\frac{x_\xi^2}{2}$ and the *resistance density* $r_\xi = \frac{\xi}{\mu_\beta(\xi)}$ with $z_\xi(w_\xi) = r_\xi w_\xi$:

$$\begin{pmatrix} \frac{\mathrm{d}x_\xi}{\mathrm{d}t} w_\xi \\ y_\beta \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ \mathbb{1}_\infty & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial H_\xi}{\partial x_\xi} \\ z_\xi(w_\xi) \\ u_\beta \end{pmatrix} \tag{15}$$

where $\mathbb{1}_\infty$ denotes an infinite dimensional row vector of 1, that is $y_\beta = \int_0^\infty \frac{\partial H_\xi}{\partial x_\xi} \mathrm{d}\xi$. Notice the total energy is $H_\beta(\mathbf{x}_\beta) = \int_{\xi \in \mathcal{C}} H_\xi(x_\xi) \mathrm{d}\xi$ with infinite dimensional state $\mathbf{x}_\beta \in \mathbb{R}^{\mathbb{R}_+}$. The realization of the dynamical element with parameter p and transfer function $\mathcal{I}_{p,\beta}(s) = (ps^\beta)^{-1}$ is given by (**??**), with $\tilde{\mu}_{\beta,p}(\xi) = \frac{\mu_\beta(\xi)}{p}$.

C.2   Fractional differentiator

Fractional damping can be modeled as combination of fractional integrators and differentiators (see [**?**, **?**, **?**, **?**]). The realization of fractional differentiator of order α with input u_α, transfer function $\mathcal{D}_\alpha(s) = s^\alpha$ and output $y_\alpha = \mathcal{D}_\alpha u_\alpha$, is built on the diffusive representation (**??**) as follows [**?**, **?**] :

$$\begin{cases} \frac{\mathrm{d}x_\xi}{\mathrm{d}t} &= -\xi \cdot x_\xi + u_\alpha, \quad x_\xi(0) = 0, \\ y_\alpha &= \int_0^{+\infty} \mu_{1-\alpha}(\xi)\big(u_\alpha - \xi \cdot x_\xi\big)\mathrm{d}\xi. \end{cases} \tag{16}$$

Defining the *hamiltonian density* $H_\xi(x_\xi) = \mu_{1-\alpha}(\xi)\xi\frac{x_\xi^2}{2}$, the *resistance density* $r_\xi = \mu_{1-\alpha}(\xi)$ and $z_\xi(w_\xi) = r_\xi w_\xi$, the pH formulation of the fractional differentiator (??) is

$$
\begin{pmatrix} \frac{\mathrm{d}x_\xi}{\mathrm{d}t} \\ w_\xi \\ y_\alpha \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{\mu_{1-\alpha}(\xi)} & 0 \\ \frac{1}{\mu_{1-\alpha}(\xi)} & 0 & -1 \\ 0 & -\mathbb{1}_\infty & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial H_\xi}{\mathrm{d}x_\xi} \\ z_\xi(w_\xi) \\ u_\alpha \end{pmatrix}. \tag{17}
$$

C.3 Finite order approximation

For implementation purpose, a finite approximation of diffusive representation (??) is built based on a finite set of n_ξ poles $(\xi_1, \ldots, \xi_{n_\xi})$ localized on the cut $\mathcal{C}$. The weights $\boldsymbol{\mu} = (\mu_1 \cdots \mu_{n_\xi})^T$ are obtained from a least square optimization as detailed in [?, sec. 5.1.2], by minimizing an appropriate distance between $\mathcal{I}_\beta$ and its discretisation $\widehat{\mathcal{I}}_\beta$:

$$
\widehat{\mathcal{I}}_\beta(s) = \sum_{n=1}^{n_\xi} \frac{\mu_n}{s + \xi_n} = \mathbf{E}(s) \cdot \boldsymbol{\mu} \quad \text{with} \quad \mathbf{E}(s) = \left(\begin{array}{ccc} \frac{1}{s+\xi_1} & \cdots & \frac{1}{s+\xi_{n_\xi}} \end{array} \right)^{\mathsf{T}}. \tag{18}
$$

The poles ξ_n's are chosen as $\xi_n = 10^{\ell_n} \in \mathcal{C}$, for $0 \leq n \leq n_\xi + 1$, where the ℓ_n's are equally spaced, with step $\delta = \frac{\ell_{n_\xi+1}-\ell_0}{n_\xi+1}$, from ℓ_0 to ℓ_{N+1} . Since gain deviations are perceived relatively to the reference gains on the audio range, the weights $\boldsymbol{\mu}$ are optimized with respect to the objective function

$$
\mathcal{O}(\boldsymbol{\mu}) = \int_{\omega_-}^{\omega_+} \left| 1 - \frac{\widehat{\mathcal{I}}_\beta(s=i\omega)}{\mathcal{I}_\beta(s=i\omega)} \right|^2 \mathrm{d}\ln\omega. \tag{19}
$$

where $\omega_- = 2\pi f_-$, $\omega_+ = 2\pi f_+$ for $[f_-, f_+] = [20\text{Hz}, 20\text{kHz}]$. In practice, the integral in (??) is approximated by a finite sum on a frequency grid, here, $\ln\omega_k = \ln\omega_- + \frac{k}{n_\omega}\ln\frac{\omega_+}{\omega_-}$ for $0 \leq k \leq n_\omega$. This yields the following practical objective function

$$
\widehat{\mathcal{O}}(\boldsymbol{\mu}) = \overline{(\mathbf{M}\boldsymbol{\mu} - \mathbf{T})}^{\mathsf{T}} \mathbf{W}(\mathbf{M}\boldsymbol{\mu} - \mathbf{T}), \tag{20}
$$

where matrix $\mathbf{M}$ is composed of the rows $[\mathbf{M}]_{k,:} = \mathbf{E}(s=i\omega_{k-\frac{1}{2}})^T$ defined in (??), where $\omega_{k-\frac{1}{2}} = \sqrt{\omega_{k-1}\omega_k}$ denotes the mean of ω_{k-1} and ω_k for $1 \leq k \leq n_\omega$. Vector $\boldsymbol{\mathcal{I}}$ is composed of $[\boldsymbol{\mathcal{I}}]_k = \mathcal{I}_\beta(s=i\omega_{k-\frac{1}{2}})$ and the diagonal matrix $\mathbf{W}$ is defined by $[\mathbf{W}]_{k,k} = (\ln\omega_k - \ln\omega_{k-1}) / |[\boldsymbol{\mathcal{I}}]_k|^2$. The minimization of (??) is achieved by off-the-shelf optimization algorithm, imposing the weights to be positive :

$$
\widehat{\boldsymbol{\mu}} = \{\min_{\boldsymbol{\mu}} \widehat{\mathcal{O}}(\boldsymbol{\mu}) : \boldsymbol{\mu} > 0\} \tag{21}
$$

The finite dimensional pH system realizing the weighted fractional integrator with transfer function $\mathcal{I}_{p,\beta} = (ps^\beta)^{-1}$ is given in table ?? with :

$$
\begin{cases} p_n &= \frac{\widehat{\boldsymbol{\mu}}_n}{p}, \\ r_n &= \frac{\xi_n}{p_n}, \end{cases} \qquad n \in (1, \cdots n_\xi). \tag{22}
$$

According to section ??, the finite dimensional approximation of the weighted fractional differentiator with transfer function $\mathcal{D}_{\alpha,p} = ps^\alpha$ is obtained from

11

State : $\mathbf{x}_\beta = \left(x_1, \cdots, x_{n_\xi}\right)^\mathsf{T}$	Energy : $\mathrm{H}_\beta(\mathbf{x}_\beta) = \frac{1}{2}\,\mathbf{x}_\beta^\mathsf{T}\,\mathrm{diag}(p_1, \cdots, p_{n_\xi})\,\mathbf{x}_\beta$
Dissipation variable : $\mathbf{w}_\beta = \left(w_1, \cdots, w_{n_\xi}\right)^\mathsf{T}$	Dissipation law : $\mathbf{z}_\beta(\mathbf{w}_\beta) = \mathrm{diag}(r_1, \cdots, r_{n_\xi})\,\mathbf{w}_\beta$
Input : u_β	Output : $\widehat{y}_\beta$
Structure : $\mathbf{J_{xx}} = \mathbb{0}_{n_\xi \times n_\xi},\ \mathbf{J_{xw}} = -\mathbf{Id}_{n_\xi},\ \mathbf{J_{xy}} = \mathbb{1}_{n_\xi \times 1},$ $\mathbf{J_{ww}} = \mathbb{0}_{n_\xi \times n_\xi},\ \mathbf{J_{wy}} = \mathbb{0}_{n_\xi \times 1},\ \mathbf{J_{yy}} = 0.$	

TABLE 1 – Port-Hamiltonian formulation (??) for the approximation of the fractional integrator $y_\beta(s) = (ps^\beta)^{-1}u_\beta(s)$ on a finite set of n_ξ poles. The parameters p_n, r_n for $n \in (1, \cdots n_\xi)$ are defined in (??) based on the minimization of (??). As an example, if $u_\beta \equiv i$ and $y_\beta \equiv v$, this structure corresponds to the serial connection of n_ξ parallel RC cells ; if $y_\beta \equiv i$ and $u_\beta \equiv v$, this structure corresponds to the parallel connection of n_ξ serial LC cells.

the minimization of (??) for the transfer function $\mathcal{I}_{1-\alpha}$. The corresponding pH formulation is given in table ?? with :

$$\begin{cases} p_n & = & p\,\widehat{\boldsymbol{\mu}}_n\,\xi_n, \\ r_n & = & p_n\widehat{\boldsymbol{\mu}}_n, \end{cases} \qquad n \in (1, \cdots n_\xi). \qquad (23)$$

State : $\mathbf{x}_\alpha = \left(x_1, \cdots, x_{n_\xi}\right)^\mathsf{T}$	Energy : $\mathrm{H}_\alpha(\mathbf{x}_\alpha) = \frac{1}{2}\,\mathbf{x}_\alpha^\mathsf{T}\,\mathrm{diag}(p_1, \cdots, p_{n_\xi})\,\mathbf{x}_\alpha$
Dissipation variable : $\mathbf{w}_\alpha = \left(w_1, \cdots, w_{n_\xi}\right)^\mathsf{T}$	Dissipation law : $\mathbf{z}_\alpha(\mathbf{w}_\alpha) = \mathrm{diag}(r_1, \cdots, r_{n_\xi})\,\mathbf{w}_\alpha$
Input : u_α	Output : $\widehat{y}_\alpha$
Structure : $\mathbf{J_{xx}} = \mathbb{0}_{n_\xi \times n_\xi},\ \mathbf{J_{xw}} = -\mathrm{diag}(\widehat{\boldsymbol{\mu}})^{-1},\ \mathbf{J_{xy}} = \mathbb{0}_{n_\xi \times 1},$ $\mathbf{J_{ww}} = \mathbb{0}_{n_\xi \times n_\xi},\ \mathbf{J_{wy}} = -\mathbb{1}_{n_\xi \times 1},\ \mathbf{J_{yy}} = 0.$	

TABLE 2 – Port-Hamiltonian formulation (??) for the approximation of the fractional differentiator $y_\alpha(s) = ps^\alpha u_\alpha(s)$ on a finite set of n_ξ poles. The parameters p_n, r_n for $n \in (1, \cdots n_\xi)$ are defined in (??) based on the minimization of (??) for the transfer function $\mathcal{I}_{1-\alpha}$. The interpretation is less intuitive than for the fractional integrator of table ?? due to the coefficients in $\mathbf{J_{xw}}$ that involves transformers.