PyPHS Documentation
Version 0.1.9b2

Antoine Falaize

6 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

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 [**?**]. We consider

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

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

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*

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

— $\mathbf{w} : t \mapsto \mathbf{w}(t) \in \mathbb{R}^{n_{\mathtt{w}}}$ 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 :

```
# 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 $d\mathbf{x}$ 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 $\mathbf{g} \in \mathbb{R}^{n_g}$,

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

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 $\nabla \mathbf{z}(\mathbf{w}) \in \mathbb{R}^{n_w \times n_w}$.

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

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

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`.

`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

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

Attributes : `phs.dims.xl` : Number of state vector components associated with a quadratic storage function : $\mathrm{H}_\mathrm{l}(\mathbf{x}_\mathrm{l}) = \mathbf{x}_\mathrm{l}^\intercal \cdot \frac{\mathbf{Q}}{2} \cdot \mathbf{x}_\mathrm{l}$, 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 : $\mathbf{z}_\mathrm{l}(\mathbf{w}_\mathrm{l}) = \mathbf{Z}_\mathrm{l} \cdot \mathbf{w}_\mathrm{l}$, 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 [**?**]. The algorithm implemented in PyPHS is exactly that in [**?**, 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 [**?**] 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

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

and the nonlinear subsystem is

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

3.2.2 Presolve numerical nonlinear subsystem

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

3.2.3 Algorithm

Inputs

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

Process

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

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

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

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

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

Non-Realizable

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

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

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

Non-Realizable

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

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

4 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.

4.1 Gyrator

$$\begin{array}{rcl} \mathfrak{e}_2 & = & \alpha\, \mathfrak{f}_1 \\ \mathfrak{e}_1 & = & -\alpha\, \mathfrak{f}_1 \end{array}$$

or equivalently :

$$\begin{array}{rcl} \mathfrak{f}_2 & = & -\alpha^{-1}\, \mathfrak{e}_1 \\ \mathfrak{f}_1 & = & \alpha\, \mathfrak{e}_2 \end{array}$$

4.2 Transformer

$$\begin{array}{rcl} \mathfrak{f}_2 & = & \alpha\, \mathfrak{f}_1 \\ \mathfrak{e}_1 & = & -\alpha\, \mathfrak{e}_2 \end{array}$$

or equivalently :

$$\begin{array}{rcl} \mathfrak{f}_2 & = & -\alpha^{-1}\, \mathfrak{e}_1 \\ \mathfrak{f}_1 & = & \alpha\, \mathfrak{e}_2 \end{array}$$

5 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).

5.1 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] :

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

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 :

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

The system (14) 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 $z_\xi(w_\xi) = r_\xi w_\xi$:

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

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 (15), with $\tilde{\mu}_{\beta,p}(\xi) = \frac{\mu_\beta(\xi)}{p}$.

5.2 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 (14) 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 (16) is

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

5.3 Finite order approximation

For implementation purpose, a finite approximation of diffusive representation (15) 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 $\widehat{\mathcal{I}}_\beta$:

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

The poles ξ_n's are chosen as $\xi_n = 10^{\ell_n} \in \mathcal{C}$, for $0 \le n \le n_\xi+1$, 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 (19) 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 $0 \le k \le n_\omega$. 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 (18), where $\omega_{k-\frac{1}{2}} = \sqrt{\omega_{k-1}\omega_k}$ denotes the mean of ω_{k-1} and ω_k for $1 \le k \le n_\omega$. 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 $[\mathbf{W}]_{k,k} = (\ln\omega_k - \ln\omega_{k-1}) \,/\, \left|[\boldsymbol{\mathcal{I}}]_k\right|^2$. The minimization of (20) 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 1 with :

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

According to section 5.2, the finite dimensional approximation of the weighted

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 : $\mathbf{J}_{\mathtt{xx}} = \mathbb{0}_{n_\xi \times n_\xi}$, $\mathbf{J}_{\mathtt{xw}} = -\mathbf{I}_{\mathbf{d}n_\xi}$, $\mathbf{J}_{\mathtt{xy}} = \mathbb{1}_{n_\xi \times 1}$, $\mathbf{J}_{\mathtt{ww}} = \mathbb{0}_{n_\xi \times n_\xi}$, $\mathbf{J}_{\mathtt{wy}} = \mathbb{0}_{n_\xi \times 1}$, $\mathbf{J}_{\mathtt{yy}} = 0$.	

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 (22) based on the minimization of (20). 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.

fractional differentiator with transfer function $\mathcal{D}_{\alpha,p} = ps^\alpha$ is obtained from the minimization of (20) for the transfer function $\mathcal{I}_{1-\alpha}$. The corresponding pH formulation is given in table 2 with :

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

State : $\mathbf{x}_\alpha = \big(x_1, \cdots, x_{n_\xi}\big)^\intercal$	Energy : $\mathrm{H}_\alpha(\mathbf{x}_\alpha) = \frac{1}{2}\,\mathbf{x}_\alpha^\intercal\ \mathrm{diag}(p_1, \cdots, p_{n_\xi})\,\mathbf{x}_\alpha$
Dissipation variable : $\mathbf{w}_\alpha = \big(w_1, \cdots, w_{n_\xi}\big)^\intercal$	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 : $\mathbf{J}_{\mathtt{xx}} = \mathbb{0}_{n_\xi \times n_\xi}$, $\mathbf{J}_{\mathtt{xw}} = -\,\mathrm{diag}(\widehat{\boldsymbol{\mu}})^{-1}$, $\mathbf{J}_{\mathtt{xy}} = \mathbb{0}_{n_\xi \times 1}$, $\mathbf{J}_{\mathtt{ww}} = \mathbb{0}_{n_\xi \times n_\xi}$, $\mathbf{J}_{\mathtt{wy}} = -\mathbb{1}_{n_\xi \times 1}$, $\mathbf{J}_{\mathtt{yy}} = 0$.	

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 (23) based on the minimization of (20) for the transfer function $\mathcal{I}_{1-\alpha}$. The interpretation is less intuitive than for the fractional integrator of table 1 due to the coefficients in $\mathbf{J}_{\mathtt{xw}}$ that involves transformers.