INTRODUCTION TO "RORPack" FOR MATLAB

A MATLAB SOFTWARE PACKAGE FOR ROBUST OUTPUT REGULATION

LASSI PAUNONEN

Saturday 5th March, 2022

Abstract. This document contains the mathematical introduction to RORPack — a software package for robust output tracking and disturbance rejection for linear PDE systems. The RORPack library is open-source and freely available at https://github.com/lassipau/rorpack/ and https://github.com/lassipau/rorpack-matlab/ The package contains functionality for automated construction of robust internal model based controllers, simulation of the controlled systems, visualisation of the results, as well as a collection of example cases on robust output regulation of controlled heat and wave equations on 1D and 2D spatial domains.

Contents

Download links 1
1. General Description 2
1.1. Controller Design and Simulation Workflow 2
2. Introduction to Robust Output Regulation 3
3. Using the Software (Matlab version) 5
4. Implemented Controller Types 10
4.1. Comments on the Controller Parameters 14
5. PDE Models of the Example Cases 15
5.1. The 1D Heat Equations 15
5.2. The 2D Heat Equations on Rectangular Domains 18
5.3. The 1D Wave Equations 20
5.4. The 2D Wave Equation on an Annulus 22
5.5. The Controlled Beam Equations 23
6. Contributors 26
Appendix A. External Links 26
References 27

Download links

Both the Python and Matlab versions of **RORPack** software are distributed as open source under the GNU General Public License version 3 (see `LICENSE.txt` for detailed license and copyright information) and can freely downloaded at the addresses

https://github.com/lassipau/rorpack/ (Python version)
https://github.com/lassipau/rorpack-matlab/ (Matlab version)

The project version information can be found in the `CHANGELOG.md` files, and installation instructions are included in the `README.md` files.

1. General Description

RORPack is a software package for controller design and simulation of robust output tracking and disturbance rejection for linear partial differential equation (PDE) models. The package is available as a Python library and a Matlab package. The software contains a number of complete examples on robust controller design and simulation of PDE models of the following types:

- one-dimensional (1D) and two-dimensional (2D) diffusion equations with control and observation either at the boundary or inside the spatial domain.
- 1D wave equations with either boundary or in-domain control and observation.
- A 2D wave equation on an annulus with boundary control and observation
- 1D beam equations with boundary or in-domain control and observation.

New examples will also be added in the future versions of the package.

The purpose of this document is to give a general introduction to the background theory of robust output regulation for linear PDE systems (Section 2), to document the usage of the software package on a general level (Sections 3 and Section 4), and to describe the mathematical models that are included as the example cases (Section 5).

The purpose of **RORPack** is to serve as a tool to *illustrate* the theory of robust output regulation for distributed parameter systems and it should not (yet) be considered as a serious controller design software. The developers of the software do not take any responsibility and are not liable for any damage caused through use of this software. In addition, at its present stage, the library is not optimized in terms of numerical accuracy. Any helpful comments on potential improvements of the numerical aspects will be greatly appreciated!

This documentation is also published in arXiv.org. The website of the project is located at the address (hosted by GitHub Pages)

https://lassipau.github.io/rorpack/

All comments and suggestions for improvements are welcome! These can be sent directly to `lassi.paunonen@tuni.fi`.

1.1. Controller Design and Simulation Workflow.

Basic workflow for robust controller design and simulation for a given system is that the user creates a "main simulation file" with the following parts:

(1) Calling of a user-defined Matlab function that returns a numerical approximation of the linear PDE model. In the provided examples this function is in a separate file with the prefix "Constr".
(2) Defining the reference and disturbance signals to be considered.
(3) Calling of a **RORPack** routine for construction of a "robust controller" that is suitable for the type of the PDE model. This also involves defining the parameters of the controller construction.

(4) Calling of **RORPack** routines for construction and simulation of the closed-loop system. The routines return numerical data describing the output, the error, the control signal, and the closed-loop state.
(5) Calling of **RORPack** routines for visualising the behaviour of the output and the output error, as well as user-defined routines for illustrating the behaviour of the state of the controlled PDE system (multidimensional plots or animations).

The example cases included in **RORPack** are built around main files that follow the above structure, and the users are encouraged to implement their own simulations by modifying the example codes.

It is also possible to combine other Python software packages and Matlab toolboxes in the study of robust controller design for PDE models in order to employ ready-made numerical approximations or additional numerical methods in the computation of required controller parameters. This approach is illustrated the PDE example cases "`Heat1DCase2_Main`" (see Section 5.1) and "`BeamKelvinVoigt_Main`" (see Section 5.5) both of which utilise the **Chebfun package** [15] (https://chebfun.org/), which provides flexible tools for numerical approximation with Chebyshev polynomials and solution of differential equations with extreme accuracy using spectral methods. In `Heat1DCase2_Main` the **Chebfun** package is used to compute high-accuracy approximations of the *transfer function* of the controlled PDE system, and in `BeamKelvinVoigt_Main` the package is used to implement a spectral Galerkin approximation [51] of an Euler–Bernoulli beam with nonlocal basis based on Chebyshev polynomials.

2. Introduction to Robust Output Regulation

In this section we give a general introduction to the mathematical theory of *robust output regulation*. The purpose of the **RORPack** package is to illustrate controller design for linear distributed parameter systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) + B_d w_{dist}(t), \qquad x(0) = x_0 \in X \tag{2.1a}$$

$$y(t) = C_\Lambda x(t) + Du(t) + D_d w_{dist}(t) \tag{2.1b}$$

on a Banach or a Hilbert space X. Controlled linear PDE models describing diffusion-convection phenomena, waves and vibrations and elastic deformations can be written in this form with a suitable differential operator A [11, 52, 29]. In our main control problem the goal is to design a dynamic error feedback controller in such a way that

> "the output $y(t)$ of the system converges to a given reference signal $y_{ref}(t)$ despite the external disturbance signal $w_{dist}(t)$".

In addition, the controller needs to be *robust* in the sense that it achieves the output tracking and disturbance rejection even if the parameters (A, B, B_d, C, D, D_d) are perturbed or contain small uncertainties.

Our main emphasis is on robust output regulation of *diffusion-convection equations*, *wave equations*, and *beam and plate equations*. However, the **RORPack** package can also be used for construction of controllers for finite-dimensional systems with given matrices (A, B, B_d, C, D, D_d).

The full robust output regulation problem is defined in the following way.

The Robust Output Regulation Problem. *Given a reference signal $y_{ref}(t)$, design a dynamic error feedback controller such that the output $y(t)$ of the system converges to the reference signal asymptotically, i.e.,*

$$\lim_{t\to\infty} \|y(t) - y_{ref}(t)\|_Y = 0 \tag{2.2}$$

despite the disturbance signal $w_{dist}(t)$. Moreover, the controller is required to be robust *in the sense that it achieves the convergence of the output (2.2) even under small uncertainties and changes in the parameters (A, B, B_d, C, D, D_d) of the system.*

The reference signal $y_{ref}(t)$ and the disturbance signals $w_{dist}(t)$ are assumed to be of the form

$$y_{ref}(t) = a_0^1 + \sum_{k=1}^{q} (a_k^1 \cos(\omega_k t) + b_k^1 \sin(\omega_k t)) \tag{2.3a}$$

$$w_{dist}(t) = a_0^2 + \sum_{k=1}^{q} (a_k^2 \cos(\omega_k t) + b_k^2 \sin(\omega_k t)) \tag{2.3b}$$

for some **known frequencies** $\{\omega_k\}_{k=0}^q \subset \mathbb{R}$ with $0 = \omega_0 < \omega_1 < \ldots < \omega_q$ and unknown amplitudes $\{a_k^j\}_{k,j}, \{b_k^j\}_{k,j} \subset \mathbb{R}$ (some of which may zero).

Dynamic feedback is essential for achieving robust output regulation, and the control problem can indeed be solved with a dynamic error feedback controller (see Figure 1). The classical *internal model principle* gives a characterization for the controllers that solve the robust output regulation problem. This fundamental result was first introduced for finite-dimensional linear systems in the 1970's by Francis and Wonham [16] and Davison [12] (see [21, Ch. 1] for an excellent overview). The internal model principle was later extended for infinite-dimensional linear systems by the Systems Theory Research Group at Tampere, Finland in the references [25, 42, 36, 44, 37][1].

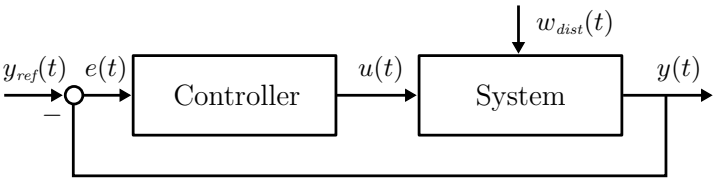

FIGURE 1. Dynamic error feedback control scheme.

The internal model principle [42, Thm. 6.9] is also the most important tool in designing controllers for robust output regulation. The result states that a controller solves the robust output regulation problem if and only if the following conditions are satisfied.

- The error feedback controller incorporates "an internal model" of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3).
- The closed-loop system is exponentially or strongly stable.

In controller design, the internal model property can be guaranteed by choosing a suitable *structure* for the controller. The rest of parameters are subsequently chosen so that the closed-loop system becomes stable.

[1]Our focus is on linear systems, but there is also an extensive literature on the internal model principle for nonlinear systems, see, e.g., [21, 7, 8] and references therein.

The detailed description of the theory of robust output regulation problem and the internal model principle can be found in the references listed below. The main emphasis in the list is (shamelessly) on the publications by the Systems Theory Research Group at Tampere University, Finland. All preprints are available at https://sysgrouptampere.wordpress.com/publications/

- [47, 53, 17, 30, 18, 48, 31, 27, 26, 14, 5, 19, 20, 43, 54]: Robust controller design in various forms for infinite-dimensional linear systems (including PI-control for PDE models).
- [42] and [44]: The Internal Model Principle for infinite-dimensional linear systems with bounded and unbounded, respectively, input and output operators B and C.
- [37, 38]: Robust controller design for *regular linear systems*.
- [24, 23, **?**]: Robust controller design for *port-Hamiltonian systems* and other boundary controlled partial differential equations.
- [48, 39, 40, **?**]: Robust controller design *impedance passive systems*.
- [35, 45, 22, 41]: Robust finite-dimensional low-order controller design for parabolic systems using Galerkin approximations and model reduction.
- [46, 50, 9, 13, 34, 1, 55, 3, 2]: References on the output regulation *without* the robustness requirement.

The considered *dynamic error feedback controllers* are of the form

$$\dot{z}(t) = \mathcal{G}_1 z(t) + \mathcal{G}_2(y(t) - y_{ref}(t)), \qquad z(0) = z_0 \in Z \tag{2.4a}$$

$$u(t) = Kz(t) + D_c(y(t) - y_{ref}(t)). \tag{2.4b}$$

Here $y(t) - y_{ref}(t)$ is the *regulation error*. The constructions of the robust controllers in **RORPack** are based on the references [48, 37, 39, 35] with certain modifications and improvements (see Section 4). In particular, the internal models are defined in their "real forms", making the controller real whenever the parameters of the plant (2.1) are real. The same controllers also achieve robust output tracking also for reference and disturbance signals with complex amplitudes $\{a_k^j\}_{k,j}, \{b_k^j\}_{k,j} \subset \mathbb{C}$.

The constructions of the robust controllers use the knowledge of the frequencies $\{\omega_k\}_k$ of the reference and disturbance signals, the number of outputs $p := \dim Y$ of the system (2.1), and certain knowledge of the system. In particular, the "Low-Gain Controller" requires knowledge of the values $P(i\omega_k)$ of the transfer function $P(\lambda) = C_\Lambda R(\lambda, A)B + D$ of the system (2.1) at the complex frequencies $\{i\omega_k\}_k \subset i\mathbb{R}$ of the reference and disturbance signals (2.3). The other controller structures also use knowledge of the parameters (A, B, C, D) of the system as they involve designing an observer for (2.1), and require the user to provide stabilizing state feedback and output injection operators. See Section 4 for the controller-specific requirements.

3. Using the Software (Matlab version)

The `rorpack` Matlab library can be installed for Matlab R2019a and later[2] as instructed in the `README.md` file included in the software package (this mainly involves copying the package on your hard drive and adding the

[2]Earlier versions should be fine, but have not been tested. Let us know of any issues!

"RORPack" folder and its subfolders to the Matlab PATH). The subdirectory `examples/` contains the included PDE examples and simulation files (documented in detail in Section 5).

The following commented example file explains the typical structure and workflow of the controller construction and simulation with **RORPack**. The considered example cases are included in the files `Heat1DCase3_Main.m` and `ConstrHeat1DCase3.m` and are documented in Section 5.1. Due to the properties of Matlab the constructor routine used in the main simulation file is in its own separate Matlab function file with the prefix "`Constr`".

The main file `Heat1DCase3_Main.m` begins with general comment lines.

```
%% Heat equation on the interval [0,1] with
% Neumann boundary control and Dirichlet boundary observation
% Approximation with a Finite differences scheme

% Two distributed control inputs and two distributed outputs y(t),
% Neumann boundary disturbance at x=0. The controls act on the
% intervals 'IB1' and 'IB2' (Default 'IB1' = [0.3,0.4] and
% 'IB2' = [0.6,0.7]) and the measurements are the average
% temperatures on the intervals 'IC1'  and 'IC2' (Default
% 'IC1' = [0.1,0.2] and 'IC2' = [0.8,0.9]).
```

The next part defines the parameters N, `cfun` and `x0fun` used in the construction of (A, B, B_d, C, D, D_d) with routine `ConstrHeat1DCase3.m` (detailed below). The parameter N is the size of the approximation and `cfun` is a function describing the spatially varying thermal diffusivity of the material. `x0fun` describes the initial heat profile of the plant. In this example, the parameters `IB1` and `IB2` describe the two intervals where the control is applied to the heat equation, and `IC1` and `IC2` describe the internvals of the measurements from the heat equation.

```
% Parameters for this example.
% Size of the numerical approximation.
N = 100;

% Initial state of the plant
%x0fun = @(x) zeros(size(x));
x0fun = @(x) 1*(1+cos(pi*(1-x)));
%x0fun = @(x) 3*(1-x)+x;

% The spatially varying thermal diffusivity of the material
% cfun = @(xi) ones(size(xi));
cfun = @(xi) 0.3-0.6*xi.*(1-xi);

IB1 = [.3, .4];  IB2 = [.6, .7];  IC1 = [.1, .2];  IC2 = [.8, .9];

[x0,Sys,spgrid,BCtype] = ...
    ConstrHeat1DCase3(cfun,x0fun,N,IB1,IB2,IC1,IC2);
```

The function `ConstrHeat1DCase3.m` used in constructing the approximation of the PDE system begins with comment lines, checking if `IB1`, `IB2`, `IC1` and `IC2` are given as parameters (or if these should be set to default values), and a consistency check for the parameters.

```
function [x0,Sys,spgrid,BCtype] = Constr1DHeatCase3(...
    cfun,x0fun,N,IB1,IB2,IC1,IC2)
% Finite Differences approximation of a 1D Heat equation with
% different types of distributed/boundary control and observation.
%
% Case 3: Two distributed control inputs and two distributed
% outputs y(t), Neumann boundary disturbance at x=0. The controls
% affect the intervals 'IB1' = [a_1,b_1] and 'IB2' = [a_2,b_2],
% and the measurements are the averages of the temperatures on the
% intervals 'IC1' = [c_1,d_1] and 'IC2' = [c_2,d_2]. If these
% parameters are not given, then defaults IB1 = [.3, .4],
% IB2 = [.6, .7], IC1 = [.1, .2], and IC2 = [.8, .9] are used.

if nargin <= 3
  IB1 = [.3, .4]; IB2 = [.6, .7]; IC1 = [.1, .2]; IC2 = [.8, .9];
end

% Check that the intervals are defined correctly
if ~isempty(find([IB1(:);IB2(:);IC1(:);IC2(:)]<0))...
        || ~isempty(find([IB1(:);IB2(:);IC1(:);IC2(:)]>1))
  error('All limits of the intervals IB1, IB2, IC1, and IC2 ...
      need to be on [0,1].')
elseif IB1(2)<=IB1(1) || IB2(2)<=IB2(1) ||...
        IC1(2)<=IC1(1) || IC2(2)<=IC2(1)
  error('All intervals IB1, IB2, IC1, and IC2 need to be of ...
      positive lengths.')
end
```

The rest of the code constructs a Finite Difference approximation of the heat equation on $[0,1]$ with the two distributed inputs (on the invervals `IB1` and `IB2`) and the two measured outputs, which are the temperature averages over the intervals `IC1` and `IC2`. The second order differential operator is constructed using the general-purpose **RORPack** routine `DiffOP1d` for approximation of a diffusion operator with a spatially varying diffusion coefficient. Also the Neumann boundary disturbance operator B_d and the initial state x_0 are approximated according to the same Finite Difference approximation. At the end of the code, all parameters of the system are saved to the main variable `Sys`, which is a Matlab structure with fields `A`, `B`, `Bd`, `C`, `D`, `Dd` corresponding to the parameters (A, B, B_d, C, D, D_d) of (2.1).

```
% The point x=1 has a Dirichlet boundary condition
% and is not chosen as a state variable
spgrid = linspace(0,1,N+1);
h = 1/N;
% Case 3 has a Neumann boundary condition at x=0
% and a Dirichlet condition at x=1
BCtype = 'ND';

[A,spgrid] = DiffOp1d(cfun,spgrid,BCtype);

% Two distributed inputs on the intervals IB1 and IB2
B1 = 1/(IB1(2)-IB1(1))*(spgrid(1:N)>=IB1(1) & ...
    spgrid(1:N)<=IB1(2));
B1 = B1(:);
```

```
B2 = 1/(IB2(2)-IB2(1))*(spgrid(1:N)>=IB2(1) & ...
    spgrid(1:N)<=IB2(2));
B2 = B2(:);

% Neumann boundary disturbance at x=0, sign is based on the
% outwards normal derivative
Bd = sparse([2/h;zeros(N-1,1)]);

% Two distributed outputs on the intervals IC1 and IC2
C1 = h/(IC1(2)-IC1(1))*(spgrid(1:N)>=IC1(1) & ...
    spgrid(1:N)<=IC1(2));
C2 = h/(IC2(2)-IC2(1))*(spgrid(1:N)>=IC2(1) & ...
    spgrid(1:N)<=IC2(2));

x0 = x0fun(spgrid(1:N)).';

Sys.A = A;
Sys.B = [B1,B2];
Sys.Bd = Bd;
Sys.C = [C1;C2];
Sys.D = zeros(2,2);
Sys.Dd = zeros(2,1);
```

The next part of the main file defines the reference signal $y_{ref}(t)$ (in `yref`) and the disturbance signal $w_{dist}(t)$ (in `wdist`) and lists the (real) frequencies $\{\omega_k\}_{k=1}^q$ in the variable `freqsReal`. Alternative reference and disturbance signals are commented out in the code for further simulation experiments. Finally, a consistency check of `Sys`, `freqsReal`, `yref`, and `wdist` is completed using the routine `SysConsistent`. *If disturbance inputs are not present in the studied PDE model, $w_{dist}(t)$ should be defined as a zero signal with* `wdist = @(t) zeros(size(t))`. *Also* `Sys.Bd` *and* `Sys.Dd` *should be defined as* $\mathrm{dimX} \times 1$ *and* $\mathrm{dimY} \times 1$ *zero matrices, and this can be done automatically using the function* `SysConsistent`.

```
% Define the reference and disturbance signals, and list the
% required frequencies in 'freqsReal'

% Case 1:
yref = @(t) [sin(2*t);2*cos(3*t)];
%wdist = @(t) zeros(size(t));
wdist = @(t) sin(6*t);
freqsReal = [1, 2, 3, 6];

% Check the consistency of the system definition
Sys = SysConsistent(Sys,yref,wdist,freqsReal);
```

The next part constructs the chosen controller structure with the corresponding function from the **RORPack** "controllers" folder. Alternative controller structures are commented out in the code for easy comparison of controller performances. The use of the controller construction routines is documented in detail in Section 4 as well as in the the comments of the example files.

```
%% Construct the controller
```

```
% A Reduced Order Observer Based Robust Controller
%
% The construction of the controller uses a Galerkin approximation
% of the heat system.
% The Galerkin approximation used in the controller design
% is a lower dimensional numerical approximation of the PDE model.
Nlow = 50;
[~,Sys_Nlow,~,~] = Constr1DHeatCase3(cfun,x0fun,Nlow,IB1,IB2,...
                                     IC1,IC2);

% Store the numerical approximation in "SysApprox".
SysApprox.AN = Sys_Nlow.A;
SysApprox.BN = Sys_Nlow.B;
SysApprox.CN = Sys_Nlow.C;
SysApprox.D = Sys_Nlow.D;

% Parameters for the stabilization step of the controller design
alpha1 = 1;
alpha2 = 0.5;
% Size of Q0 = dimension of the internal model
Q0 = eye(IMdim(freqsReal,size(SysApprox.CN,1)));
Q1 = eye(size(SysApprox.AN,1)); % Size = dim(V_N)
Q2 = eye(size(SysApprox.AN,1)); % Size = dim(V_N)
R1 = eye(size(SysApprox.CN,1)); % Size = dim(Y)
R2 = eye(size(SysApprox.BN,2)); % Size = dim(U)

% Size of the final reduced-order observer part of the controller
ROMorder = 3;

ContrSys = ObserverBasedROMRC(freqsReal,SysApprox,alpha1,...
                              alpha2,R1,R2,Q0,Q1,Q2,ROMorder);
```

The next part constructs and simulates the behaviour of the closed-loop system. In particular, `CLSys` is a Matlab structure which stores the main operators (A_e, B_e, C_e, D_e) of the closed-loop system in the fields `CLSys.Ae`, `CLSys.Be`, `CLSys.Ce`, and `CLSys.De`[3].

```
%% Closed-loop construction and simulation

% Construct the closed-loop system
CLSys = ConstrCLSys(Sys,ContrSys);

% Print an approximate stability margin of the closed-loop system
stabmarg = CLStabMargin(CLSys)

% Define the initial state of the closed-loop system
% (the controller has zero initial state by default).
xe0 = [x0;zeros(size(ContrSys.G1,1),1)];

% Set the simulation length and define the plotting grid
Tend = 8;
tgrid = linspace(0,Tend,300);
```

[3]There are different conventions in the literature for definition of the "inputs" of the closed-loop system (and the definition of B_e). **RORPack** uses the version where the input of the closed-loop system is $(w_{dist}(t), y_{ref}(t))^T$ (like in [35]), instead of the state of *an exosystem*.

```
% Simulate the closed-loop system
CLsim = SimCLSys(CLSys,xe0,yref,wdist,tgrid,[]);
```

Finally, the results of the simulation are plotted in separate figures and the behaviour of the controlled state is animated using the user-defined routines. Code for saving the animation as an MPEG-4 or AVI file is commented out.

```
%% Visualization

% Plot the (approximate) eigenvalues of the closed-loop system
figure(1)
PlotEigs(CLSys.Ae,[-30 .3 -6 6])

% Choose whether or not to print titles of the figures
PrintFigureTitles = true;

% Plot the controlled outputs, the tracking error norm, and
% the control inputs
figure(2)
subplot(3,1,1)
PlotOutput(tgrid,yref,CLsim,PrintFigureTitles)
subplot(3,1,2)
PlotErrorNorm(tgrid,CLsim,PrintFigureTitles)
subplot(3,1,3)
PlotControl(tgrid,CLsim,PrintFigureTitles)

% In plotting and animating the state,
% fill in the homogeneous Dirichlet boundary condition at x=1
spgrid = [spgrid, 1];

figure(3)
Plot1DHeatSurf(CLsim.xesol(1:N,:),spgrid,tgrid,BCtype)

figure(4)
% No movie recording
[~,zlims] = Anim1DHeat(CLsim.xesol(1:N,:),spgrid,tgrid, ...
                       BCtype,0.03,0);
% Movie recording
% [MovAnim,zlims] = Anim1DHeat(CLsim.xesol(1:N,:),spgrid,tgrid,...
%                              BCtype,0.03,1);

figure(5)
tt = linspace(0,16,500);
plot(tt,yref(tt),'Color',1.1*[0 0.447 0.741],'Linewidth',3);
title('Reference $y_{ref}$','Interpreter','latex','Fontsize',16)
```

4. Implemented Controller Types

In this section we list the concrete controllers implemented in **RORPack**. The documentation of the code includes additional information on the usage of the construction routines. The controllers are the following:

- The "Low-Gain (Minimal) Robust Controller" (including only the internal model), based on references [18, 48], [37, Sec. IV]. Stabilization of the closed-loop system is based on selection of a suitable

low-gain parameter $\varepsilon > 0$. This controller can be used for systems which are exponentially stable (or stabilizable with output feedback).

Calling sequence for the construction:

```
LowGainRC(freqsReal,Pvals,epsgain,Sys,Dc)
```

where `Sys` contains the parameters of the plant, `freqsReal` contains the frequencies $\{\omega_k\}_{k=0}^q$ of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3), `epsgain` is the value of the low-gain parameter $\varepsilon > 0$. The (optional) parameter `Dc` is the feedthrough matrix D_c of the controller (2.4). Since the controller feedthrough is equivalent to a "preliminary" output feedback $u(t) = D_c y(t) + \tilde{u}(t)$ to the system (plus an additional disturbance term), for unstable systems the parameter `Dc` should be an output feedback matrix which stabilizes the system exponentially. If `Dc` is not given, the controller will by default have zero feedthrough.

The parameter `Pvals` is a is a Matlab cell array of length $q+1$ containing transfer function values $P_{D_c}(i\omega_k) \in \mathbb{C}^{p\times m}$ at the complex frequencies $\{i\omega_k\}_{k=0}^q$ of $w_{dist}(t)$ and $y_{ref}(t)$. Here $P_{D_c}(\lambda)$ is the transfer function of the system $(A^{D_c}, B^{D_c}, C^{D_c}, D^{D_c})$ which is obtained from (A, B, C, D) with output feedback $u(t) = D_c y(t) + \tilde{u}(t)$. The values have the theoretical formula

$$P_{D_c}(i\omega_k) = C_\Lambda^{D_c} R(i\omega_k, A^{D_c}) B^{D_c} + D^{D_c},$$

but they can also be computed via a solution of a boundary value problem (see, e.g., `Heat1DCase2_Main.m` in Section 5.1), or measured from the response of a physical model. If the transfer function $P(\lambda)$ of (A, B, C, D) is well-defined at iw_k, then $P_{D_c}(i\omega_k)$ can be computed with the formula $P_{D_c}(i\omega_k) = (I - P(i\omega_k)Dc)^{-1}P(i\omega_k)$, and if $D_c = 0$, then $P_{Dc}(i\omega_k) = P(i\omega_k)$. See Section 4.1 for additional comments.

The parameter `epsgain` can alternatively be a vector of length 2 providing minimal and maximal values for ε. The controller construction has a naive functionality for finding an ε to optimize stability margin of the numerically approximated closed-loop system (simply by starting from the minimal value and increasing ε in steps).

- The "Observer Based Robust Controller", based on references [19, Sec. 7], [37, Sec. VI], [38, Sec. 5]. The closed-loop stability is achieved using an observer for the state of the system (2.1). This controller can be used for both stable and unstable systems.

  Calling sequence for the construction:

  ```
  ObserverBasedRC(freqsReal,Sys,K21,L,
                  IMstabtype,IMstabmarg)
  ```

  where `Sys` contains the parameters of the plant and `freqsReal` contains the frequencies $\{\omega_k\}_{k=0}^q$ of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3). The parameter `L` describes an operator L such that $A + LC_\Lambda$ generates an exponentially stable semigroup.

  Instead of the approach used in [37], the "internal model", i.e., the pair (G_1, B_1), in the controller is stabilized using either LQR-based design (`IMstabtype = 'LQR'`) or pole placement (`IMstabtype`

`= 'poleplacement'`) with a stability margin `IMstabmarg`. In these two cases the parameter `K21` describes an operator K_{21} such that $A + BK_{21}^{\Lambda}$ generates an exponentially stable semigroup. Note that the variable `IMstabmarg` only determines the stability margin of the internal model, and the stability margin of the closed-loop system also depends on the stability margins of the semigroups generated by $A + BK_2^{\Lambda}$ and $A + LC_{\Lambda}$.

Alternatively, when `IMstabtype = 'full_K'`, the parameter `K21` describes an operator K_{21} such that the semigroup generated by

$$\begin{bmatrix} G_1 & G_2C_{\Lambda} \\ 0 & A \end{bmatrix} + \begin{bmatrix} G_2D \\ B \end{bmatrix} K_{21}^{\Lambda}$$

is exponentially stable.

- The "The Reduced Order Observer Based Robust Controller", for *parabolic* PDE systems, especially convection-diffusion-reaction equations. The controller design is based on Galerkin approximations and model reduction via Balanced Truncation, and was introduced in the reference [35] (see also [45] for systems with boundary control).

  Calling sequence for the construction:

  ```
  ObserverBasedROMRC(freqsReal,sysApprox,alpha1,alpha2,
                     R1,R2,Q0,Q1N,Q2N,ROMorder)
  ```

  where `sysApprox` contains the parameters of the Galerkin approximation (A^N, B^N, C^N, D) of the plant parameters (A, B, C, D) (a struct with fields `sysApprox.AN`, `sysApprox.BN`, `sysApprox.CN`, `sysApprox.D`) and `freqsReal` contains the (real) frequencies $\{\omega_k\}_{k=0}^{q}$ of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3). The parameters `alpha1`, `alpha2`, `R1`, `R2`, `Q0`, `Q1N`, `Q2N` are parameters for the LQR/LQG-based stabilization of the closed-loop system (details below). Finally, `ROMorder` is the desired dimension of the reduced order "observer part" of the controller. The final controller has dimension which is equal to the dimension of the internal model plus `ROMorder`.

  The parameters `alpha1` and `alpha2` correspond to $\alpha_1, \alpha_2 \geq 0$ in [35, Sec. III.A], and are used as design parameters to determine the closed-loop stability margin (the choice $\alpha_1 = \alpha_2 = 0$ is possible). In particular, $(A + \alpha_k I, B, C)$ should be exponentially stabilizable and detectable for $k = 1, 2$. The parameters `R1`, `R2`, and `Q0` correspond to $R_1 \in \mathcal{L}(Y)$, $R_2 \in \mathcal{L}(U)$, and $Q_0 \in \mathbb{R}^{n_{IM} \times n_{IM}}$ (where $n_{IM} \in \mathbb{N}$ is the dimension of the internal model). All these matrices should be chosen to be invertible and positive definite, and for example choices $R_1 = r_1 I$, $R_2 = r_2 I$, $Q_0 = q_0 I$ for some $r_1, r_2, q_0 > 0$ are possible (the dimension n_{IM} of Q_0 can be computed with the routine `IMdim(freqsReal,p)`). Finally, `Q1N` and `Q2N` correspond to the Galerkin approximations (compatible with `sysApprox`) of the parameters $Q_1 \in \mathcal{L}(U_0, X)$ and $Q_2 \in \mathcal{L}(X, Y_0)$ in [35, Sec. III.A]. They can be chosen to correspond to the Galerkin approximations of $Q_1 = q_1 I$ and $Q_2 = q_2 I$ for some $q_1, q_2 > 0$.

The model reduction dimension parameter `ROMorder` corresponds to r in [35, Sec. III.A] and it determines the final reduced order dimension of the observer part of the controller. The model reduction step can be skipped completely by setting `ROMorder=NaN`. The size of the parameter should be determined to be sufficiently high so that the closed-loop system is exponentially stable. The result [35, Thm. III.1] guarantees that closed-loop stability for the original PDE plant and the reduced order controller is achieved if both the Galerkin approximation order N and the model reduction parameter $r \geq N$ are sufficiently high, but (at the moment) there are no concrete lower bounds for these values. Typically, if the decay of the Hankel singular values of (A^N, B^N, C^N) are sufficiently fast, it is possible to achive the closed-loop stability with a fairly small model reduction parameter r.

- The "Dual Observer Based Robust Controller", based on references [37, Sec. V], [38, Sec. 4]. The closed-loop stability is achieved using a complementary controller structure which (formally) coincides with observer based stabilization of the dual of the closed-loop system.

  Calling sequence for the construction:

  ```
  DualObserverBasedRC(freqsReal,Sys,K2,L1,
                                  IMstabtype,IMstabmarg)
  ```

  where `Sys` contains the parameters of the plant and `freqsReal` contains the frequencies $\{\omega_k\}_{k=0}^q$ of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3). The parameters `K2` and `L1` describe operators K_2 and L_1 such that $A + BK_2^\Lambda$ and $A + L_1 C_\Lambda$ generate exponentially stable semigroups.

  Instead of the approach in [37], the "internal model", i.e., the pair (C_1, G_1), in the controller is stabilized either using LQR-based design (`IMstabtype = 'LQR'`) or alternatively using pole placement (`IMstabtype = 'poleplacement'`) with a predefined stability margin `IMstabmarg`. Note that the variable `IMstabmarg` only determines the stability margin of the internal model, and the stability margin of the closed-loop system also depends on the stability margins of the semigroups generated by $A + BK_2^\Lambda$ and $A + L_1 C_\Lambda$.

- The "Passive (Minimal) Robust Controller" based on [48, Thm. 1.2], [39, Sec. 5.1], [**?**]. The controller can be used for *impedance passive* linear systems which are either exponentially or strongly stable, or stabilizable with output feedback. The controller design is based on forming a *power preserving interconnection* between the passive system and the controller to achieve closed-loop stability.

  Calling sequence for the construction:

  ```
  PassiveRC(freqsReal,dimY,epsgain,Sys,Dc)
  ```

  where `freqsReal` contains the (real) frequencies $\{\omega_k\}_{k=0}^q$ of the signals $y_{ref}(t)$ and $w_{dist}(t)$ in (2.3), `dimY` is the dimension of the output space Y (i.e., the number of outputs of the system), and `epsgain` is

a scalar parameter $\varepsilon > 0$ which (linearly) scales the norms of K and G_2.

The parameter `epsgain` can alternatively be a vector of length 2 providing minimal and maximal values for ε. The controller construction has a naive functionality for finding an ε to optimize stability margin of the numerically approximated closed-loop system (simply by starting from the minimal value and increasing ε in steps). In this case, the parameter `Sys` (containing the parameters of the plant) is used for this optimization.

The optional parameter `Dc`, which is either $\dim Y \times \dim Y$-matrix or a scalar, is the feedthrough matrix D_c for the controller. The closed-loop structure implies that a nonzero feedthrough term D_c is equivalent to a "preliminary" output feedback $u(t) = D_c y(t) + \tilde{u}(t)$ to the system (plus an additional disturbance term). For unstable systems the parameter `Dc` should be an output feedback matrix which stabilizes the system either exponentially or strongly, and in particular, for exactly or approximately controllable unstable passive systems `Dc` can be any negative definite matrix. If `Dc` is not given, the controller will by default have zero feedthrough. If `Dc` is a scalar, the parameter is interpreted as "`Dc` times identity".

It should be noted that the controller construction routine does not test passivity, and for a non-passive system the resulting closed-loop system will typically be unstable.

4.1. **Comments on the Controller Parameters.** In this section we make some remarks on the choices of the controller parameters.

The gain parameter $\varepsilon > 0$ in `LowGainRC`. The theory in [18, 48] guarantees that for a stable system (A, B, C, D) there exists $\varepsilon^* > 0$ such that for any $0 < \varepsilon < \varepsilon^*$ the closed-loop system is exponentially stable. The stability margin of the closed-loop system (which directly determines the convergence rate of the regulation error $e(t)$) can be optimized with a suitable choice of $\varepsilon > 0$. Finding such an optimal value of $\varepsilon > 0$ for a PDE system is a challenging task, but in numerical simulations one can use a naive approach of tracking the spectrum of the finite-dimensional closed-loop system matrix $A_e(\varepsilon)$. This is the approach taken in the example cases in **RORPack**, though it should be noted that for general PDE systems there is no guarantee that the value $\varepsilon > 0$ obtained this way would optimize the closed-loop system for the original PDE system. In the example cases we are mainly interested in finding an $\varepsilon > 0$ which achieves a reasonable rate of convergence rate of the error $\|e(t)\|$.

The gain in the passive robust controller `PassiveRC` can analogously be adjusted with a choice of a parameter $\varepsilon > 0$ to optimize the closed-loop stability margin.

The transfer function values. The controllers make use of the values $P_{D_c}(i\omega_k)$ (or $P(i\omega_k)$) of the transfer functions associated to the original or stabilized versions of the PDE system. Once the system `Sys` is defined, the simplest (and most tempting) approach is to compute these values as

```
SysFB.C*((1i*wk*eye(dimX)-SysFB.A)\SysFB.B) + SysFB.D
```

where the approximation `SysFB` of $(A^{D_c}, B^{D_c}, C^{D_c}, D^{D_c})$ is computed from `Sys` as an output feedback with `SysFB = SysOutputFeedback(Sys,Dc)`, and `dimX=size(SysFB.A,1)`. However, using the same approximation of the PDE for both controller design and simulation corresponds to essentially controlling the approximation *as a finite-dimensional system*. If possible, to avoid unrealistically positive results, it is therefore better to use two different approximations for controller construction and simulation. This is especially the case if the validity of the approximation for the computation of the parameters can not be guaranteed with absolute certainty. However, there are of course cases, such as in Galerkin approximations of parabolic systems [33], where the values of the controller parameters can be shown to converge with the order of the approximation, and the above concerns are at least for the most part unnecessary. In the Matlab version, the observer based controllers do not (at least currently) use the transfer functions for the construction of the controllers. For simplicity, several built-in examples in **RORPack** still use `Sys` for computing `Pvals`, and the user is encouraged to replace these computations with other approximation methods!

For some special PDEs, such as 1D heat or wave equations with constant coefficients, the values $P(i\omega_k)$ may have explicit expressions, but these are very limited special cases. For PDEs with spatially varying parameters there are powerful computational methods that can be used to determine $P(i\omega_k)$, such as the **Chebfun** package (https://chebfun.org/) employed in `Heat1DCase1_Main.m` in Section 5.1 ("Case 2").

5. PDE Models of the Example Cases

In this section we introduce the PDE models considered in the example cases and review their fundamental properties. The considered reference and disturbance signals are in each case combinations of trigonometric functions of the form (2.3) with a given set of frequencies. The precise choices of the signals can be seen from the main files of the examples. Similarly, the chosen initial states are visible from the source files, and in several files alternative initial states are provided (these can be used by uncommenting the corresponding lines of code).

5.1. The 1D Heat Equations.

This collection of examples consists of 1D heat equations with spatially varying thermal diffusivity on $\Omega = [0, 1]$ with different configurations of control inputs and measured outputs. Cases 1, 2, 4, and 5 consider control, disturbance, and observation on the boundary, and Case 3 considers a system with two distributed inputs and outputs. In general, any combination of the above types of inputs and outputs is possible. The main properties from the point of view of robust output regulation is whether or not the uncontrolled system is exponentially stable (the "minimal" Low-Gain Robust Controller can be used) or impedance passive (the Passive Robust Controller can be used).

In each of the cases, the spatial discretization of the PDE is completed using Finite Differences.

Case 1: Neumann boundary input at $\xi = 0$, disturbance and output at $\xi = 1$.

Main file name: `Heat1DCase1_Main.m`

The model on $\Omega = [0,1]$ is

$$\begin{aligned}\frac{\partial x}{\partial t}(\xi,t) &= \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial x}{\partial \xi})(\xi,t), \qquad x(\xi,0) = x_0(\xi)\\ -\frac{\partial x}{\partial \xi}(0,t) &= u(t), \qquad \frac{\partial x}{\partial \xi}(1,t) = w_{dist}(t),\\ y(t) &= x(1,t),\end{aligned}$$

where $c(\cdot) \geq c_0 > 0$ is the spatially varying *thermal diffusivity* of the material. The uncontrolled system is unstable due to the eigenvalue $\lambda = 0$. The example uses the "Observer Based Robust Controller" and "Dual Observer Based Robust Controller" to achieve robust output tracking and disturbance rejection.

Case 2: Input, output, and disturbance at $\xi = 0$, Dirichlet at $\xi = 1$.

Main file name: `Heat1DCase2_Main.m`

The model on $\Omega = [0,1]$ is

$$\begin{aligned}\frac{\partial x}{\partial t}(\xi,t) &= \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial x}{\partial \xi})(\xi,t), \qquad x(\xi,0) = x_0(\xi)\\ -\frac{\partial x}{\partial \xi}(0,t) &= u(t) + w_{dist}(t), \qquad x(1,t) = 0,\\ y(t) &= x(0,t),\end{aligned}$$

where $c(\cdot) \geq c_0 > 0$ is the spatially varying thermal diffusivity of the material. The uncontrolled system is exponentially stable due to the homogeneous Dirichlet boundary condition at $\xi = 1$, and therefore the Low-Gain Robust Controller can be used. The system is also impedance passive since the control input and measured output are collocated, and because of this the robust output regulation problem can alternatively be solved using the Passive Robust Controller.

In the example, the construction of the Low-Gain Robust Controller employs the **Chebfun** Matlab package in computing the required transfer function values $P(i\omega_k) = C_\Lambda R(i\omega_k, A)B + D$. This can be done by noting that for each k the desired transfer function value is exactly $P(i\omega_k) = \hat{x}(0)$, where $\hat{x}(\cdot)$ is the (unique) solution of the boundary value problem

$$\begin{aligned}i\omega_k \hat{x}(\xi) - \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial \hat{x}}{\partial \xi})(\xi) &= 0\\ -\frac{\partial \hat{x}}{\partial \xi}(0) = 1, \qquad \hat{x}(1) &= 0.\end{aligned}$$

Chebfun excels in solving such equations with high accuracy, and therefore the use of this package yields excellent estimates of $P(i\omega_k)$ for the original system.

Case 3: Distributed input and output, boundary disturbance at $\xi = 0$, Dirichlet boundary condition at $\xi = 1$.

Main file name: `Heat1DCase3_Main.m`

The model on $\Omega = [0,1]$ is

$$\frac{\partial x}{\partial t}(\xi,t) = \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial x}{\partial \xi})(\xi,t) + b_1(\xi)u_1(t) + b_2(\xi)u_2(t)$$
$$-\frac{\partial x}{\partial \xi}(0,t) = w_{dist}(t), \qquad x(1,t) = 0, \quad x(\xi,0) = x_0(\xi),$$
$$y(t) = \int_0^1 \begin{bmatrix} c_1(\xi)x(\xi,t) \\ c_2(\xi)x(\xi,t) \end{bmatrix} d\xi \in \mathbb{R}^2$$

where $c(\cdot) \geq c_0 > 0$ is the spatially varying thermal diffusivity of the material and

$$b_1(\xi) = 10\chi_{[.3,.6]}(\xi), \quad b_2(\xi) = 10\chi_{[.6,.7]}(\xi),$$
$$c_1(\xi) = 10\chi_{[.1,.2]}(\xi), \quad c_2(\xi) = 10\chi_{[.8,.9]}(\xi).$$

Here $\chi_{[a,b]}(\cdot)$ is the characteristic function on the interval $[a,b] \subset [0,1]$, and thus the control inputs act on the intervals $[0.3, 0.4]$ and $[0.6, 0.7]$, and the outputs measure the average temperatures on the intervals $[0.1, 0.2]$ and $[0.8, 0.9]$ (alternative input and output locations can be set with parameters for the routine `ConstrHeat1DCase3`). The uncontrolled system is stable due to the homogeneous Dirichlet boundary condition at $\xi = 1$. The default controller used in this example is the Reduced Order Observer Based Robust Controller in `ObserverBasedROMRC`.

Figure 2 shows example results of the simulations including plots of the outputs and reference signals, norm of the regulation error, computed control signals, and the state of the controlled system as a function of ξ and t.

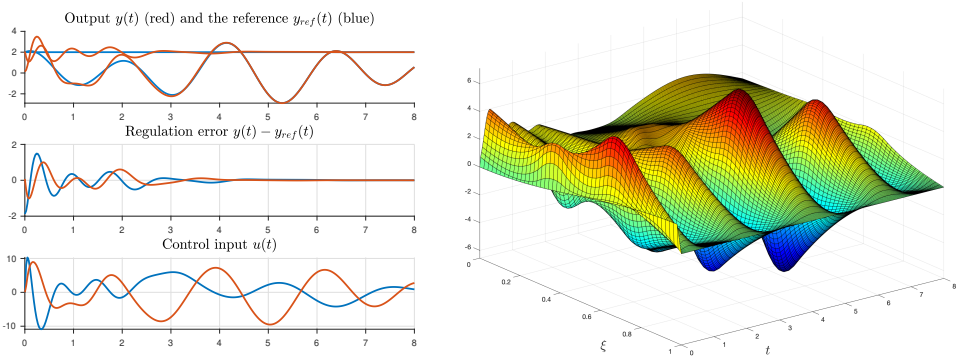

FIGURE 2. Example output of the 1D Heat equation ("Case 3").

Case 4: Dirichlet boundary input at $\xi = 1$, boundary disturbance and measurement at $\xi = 0$.

Main file name: `Heat1DCase4_Main.m`

The model on $\Omega = [0,1]$ is

$$\frac{\partial x}{\partial t}(\xi,t) = \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial x}{\partial \xi})(\xi,t)$$
$$-\frac{\partial x}{\partial \xi}(0,t) = w_{dist}(t), \qquad x(1,t) = u(t), \qquad x(\xi,0) = x_0(\xi),$$
$$y(t) = x(0,t)$$

where $c(\cdot) \geq c_0 > 0$ is the spatially varying thermal diffusivity of the material. The uncontrolled system is stable because of the Dirichlet-type boundary condition at $\xi = 1$. The system, however, does not give rise to a wellposed or regular linear system on the natural state space $X = L^2(0,1)$, but only on $X = H^{-1}(0,1)$. It does, however, give rise to a well-defined Boundary Control System [49, 10, 32] on $X = L^2(0,1)$, and the use of the Low-Gain Robust Controller (`LowGainRC`) is justified by the theory presented in [24, 23].

Other controller designs can be used in this simulation example, but as the authors of the software are not familiar with the details on approximation of regular linear systems defined on $H^{-1}(0,1)$, this part of the simulation example should be considered to be for experimentation purposes only. That is, *use at your own risk!* ☺

Case 5: Neumann boundary inputs and input disturbances at $\xi = 0$ and $\xi = 1$, and with collocated outputs.

Main file name: `Heat1DCase5_Main.m`

The model defined on $\Omega = [0,1]$ is similar to "Case 1", but now has two Neumann boundary control inputs and collocated outputs (at $\xi = 0$ and $\xi = 1$), i.e.,

$$\begin{aligned} \frac{\partial x}{\partial t}(\xi,t) &= \frac{\partial}{\partial \xi}(c(\xi)\frac{\partial x}{\partial \xi})(\xi,t) + b_d(\xi)w_{dist,3}(t), \qquad x(\xi,0) = x_0(\xi) \\ -\frac{\partial x}{\partial \xi}(0,t) &= u_1(t) + w_{dist,1}(t), \qquad \frac{\partial x}{\partial \xi}(1,t) = u_2(t) + w_{dist,2}(t), \\ y(t) &= (x(0,t), x(1,t))^T \end{aligned}$$

where $c(\cdot) \geq c_0 > 0$ is the spatially varying *thermal diffusivity* of the material. The third disturbance input acts through the input profile $b_d(\cdot)$ (defined by `Bd_profile` in the code). The uncontrolled system is unstable due to the eigenvalue $\lambda = 0$. The system is impedance passive (due to the collocated inputs and outputs) and it can be stabilized with negative output feedback. For controller design we can use the Passive Robust Controller (with a negative definite feedthrough term), or alternatively either the Observer Based Robust Controller or Dual Observer Based Robust Controller.

5.2. **The 2D Heat Equations on Rectangular Domains.** These examples consider two-dimensional heat equations on $\Omega = [0,1] \times [0,1]$ with boundary input and output and boundary disturbances. The inputs and outputs act in an averaged sense on parts of the boundaries. The reference and disturbance signals are again of the form (2.3) and can be seen from the main files of the simulations.

Case 1: Collocated input and output, a prestabilized system.

Main file name: `Heat2DCase1_Main.m`

This example is taken from the article [37, Sec. VII]. The system on $\Omega = [0,1] \times [0,1]$ with two inputs $u(t) = (u_1(t), u_2(t))^T$ and two outputs

$y(t) = (y_1(t), y_2(t))^T$ is determined by

$$\begin{aligned} x_t(\xi,t) &= \Delta x(\xi,t), \qquad x(\xi,0) = x_0(\xi) \\ \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_1} &= u_1(t), \qquad \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_2} = u_2(t), \qquad \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_0} = 0 \\ y_1(t) &= \int_{\Gamma_1} x(\xi,t)d\xi, \qquad y_2(t) = \int_{\Gamma_2} x(\xi,t)d\xi. \end{aligned}$$

Here the parts Γ_0, Γ_1, and Γ_2 of the boundary $\partial\Omega$ are defined so that $\Gamma_1 = \{\, \xi = (\xi_1, 0) \mid 0 \le \xi_1 \le 1/2 \,\}$, $\Gamma_2 = \{\, \xi = (\xi_1, 1) \mid 1/2 \le \xi_1 \le 1 \,\}$, $\Gamma_0 = \partial\Omega \setminus (\Gamma_1 \cup \Gamma_2)$. By [6, Cor. 2] the heat equation defines a regular linear system with feedthrough $D = 0$. The system is also impedance passive due to the fact that the inputs $u(t)$ and the outputs $y(t)$ are collocated. The uncontrolled system is unstable due to the eigenvalue $\lambda = 0$, but it can be stabilized exponentially with negative output feedback $u(t) = -\kappa y(t)$ for any $\kappa > 0$. In this example the constructed system is already pre-stabilized with $\kappa = 1$ (see "Case 3" for a version without prestabilization).

In the simulations, the system is approximated using the eigenmodes of the Laplacian. The system can be controlled either with the minimal Low-Gain Robust Controller (`LowGainRC`) or the Passive Controller (`PassiveRC`) (with pre-stabilizing negative output feedback), or with one of the Observer Based Robust Controllers (`ObserverBasedRC` or `DualObserverBasedRC`).

Case 2: Non-collocated input and output.

Main file name: `Heat2DCase2_Main.m`

The model is similar to "Case 1", but the inputs and outputs of the system are not collocated. Instead, the control input, the disturbance input, and the measured output act on distinct parts of the boundary of the rectangle $\Omega = [0,1] \times [0,1]$. The PDE system is defined as

$$\begin{aligned} x_t(\xi,t) &= \Delta x(\xi,t), \qquad x(\xi,0) = x_0(\xi) \\ \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_1} &= u(t), \qquad \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_3} = w_{dist}(t), \qquad \frac{\partial x}{\partial n}(\xi,t)|_{\Gamma_0} = 0 \\ y(t) &= \int_{\Gamma_2} x(\xi,t)d\xi. \end{aligned}$$

Here the parts Γ_0, Γ_1, Γ_2, and Γ_3 of the boundary $\partial\Omega$ are defined so that $\Gamma_1 = \{\, \xi = (0, \xi_2) \mid 0 \le \xi_2 \le 1 \,\}$, $\Gamma_2 = \{\, \xi = (\xi_1, 1) \mid 0 \le \xi_1 \le 1 \,\}$, $\Gamma_3 = \{\, \xi = (\xi_1, 0) \mid 0 \le \xi_1 \le 1/2 \,\}$, $\Gamma_0 = \partial\Omega \setminus (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3)$. By [6, Cor. 2] the heat equation defines a regular linear system with feedthrough $D = 0$. The uncontrolled system is unstable due to the eigenvalue $\lambda = 0$.

Since the system is unstable, the primary choices of the controllers are the Observer Based Robust Controller (`ObserverBasedRC`) and Dual Observer Based Robust Controller (`DualObserverBasedRC`). However, since the single unstable eigenvalue can be stabilized also by negative output feedback of the form $u(t) = -\kappa y(t)$, it is also possible to use the Low-Gain Robust Controller (`LowGainRC`) with a negative definite feedthrough term $D_c = -\kappa I$ (given as the optional parameter `Dc`). The Low-Gain Robust Controller is by default commented out in the code in `Heat2DCase2_Main.m`.

In the simulations, the system is approximated using Finite Differences with a uniform grid.

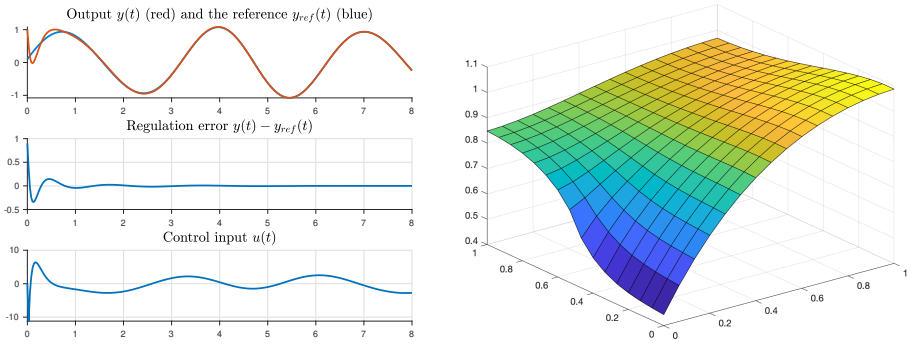

FIGURE 3. Example output of the 2D Heat equation ("Case 2").

Case 3: Collocated input and output.

Main file name: `Heat2DCase3_Main.m`

The example is the same as "Case 1", but does not include prestabilization of the unstable eigenvalue at $\lambda = 0$. Since the system is impedance passive, the controller design can be completed using the Passive Robust Controller (`PassiveRC`), but the controller requires the (otherwise optional) feedthrough term, which should be a negative definite 2×2 matrix. Alternatively, the controller design can be completed using either the Observer Based Robust Controller (`ObserverBasedRC`) or the Dual Observer Based Robust Controller (`DualObserverBasedRC`). In these cases, the required state feedback operator K_{21} and and output injection operator L for the exponential stabilization of the pairs (A, B) and (C_Λ, A) can be designed using collocated design, i.e., by choosing these as $K_{21} = -\kappa_K B^*$ and $L = -\kappa_L C^*$. Note that in the article [37] the stabilizing operators were assumed to be bounded, but the controller design can also be constructed with *unbounded* operators due to the more general results in [38].

5.3. **The 1D Wave Equations.** In these examples we consider one-dimensional undamped and damped wave equations with control and observation at the boundaries and inside the domain.

Case 1: Non-collocated boundary input and output, boundary disturbance near the output.

Main file name: `Wave1DCase1_Main.m`

The model on $\Omega = [0, 1]$ is

$$x_{tt}(\xi, t) = x_{\xi\xi}(\xi, t), \qquad x(\xi, 0) = x_0(\xi), \quad x_t(\xi, 0) = x_1(\xi)$$
$$-\frac{\partial x}{\partial \xi}(0, t) = w_{dist}(t), \qquad \frac{\partial x}{\partial \xi}(1, t) = u(t),$$
$$y(t) = x_t(0, t), \qquad y_m(t) = \int_0^1 x(\xi, t) dt$$

The uncontrolled system is unstable due to lack of damping. It also cannot be stabilized with output feedback due to the non-collocated configuration of the inputs and outputs. The goal is to achieve tracking of the output $y(t)$. The additional measured output $y_m(t)$ is required to achieve closed-loop stability due to the fact that the system is not exponentially detectable with output $y(t)$ (because of the unobservability of the eigenvalue $\lambda = 0$).

In the example case, the second output $y_m(t)$ is used to prestabilize eigenvalue $\lambda = 0$ of the system with preliminary output feedback of the form $u(t) = -\kappa_m y_m(t) + \tilde{u}(t)$ (where $\tilde{u}(t)$ is the new input of the system). There are also other (and better) ways to handle the situation, and these will be implemented in later versions of this example. The pairs (A, B) and (C, A) are stabilized using collocated designs, i.e., we choose $K = -\kappa_K B^*$ and $L = -\kappa_L C^*$ to stabilize the pairs $A + BK$ and $A + LC$. The main file contains the pre-stabilization gain $K_m > 0$ and the gains $\kappa_K, \kappa_L > 0$ as design parameters.

The example considers output tracking of the velocity $x_t(0, t)$ to arbitrary 2-periodic reference signals. This is achieved by including frequencies of the form $k\pi$ for $k \in \{1, \ldots, q\}$ in the internal model. The reference signal is defined by defining its profile over one period in the variable '`yref1per`'. Note that it is not necessary to compute the amplitudes of the frequency components of $y_{ref}(t)$ (which would be equivalent to finding the Fourier series expansion of the reference signal).

In the simulations the system is approximated using the orthonormal eigenfunctions of the undamped wave operator.

Case 2: Collocated distributed input and output.

Main file name: `Wave1DCase2_Main.m`

The model on $\Omega = [0, 1]$ is

$$\begin{aligned} x_{tt}(\xi, t) &= x_{\xi\xi}(\xi, t) + b(\xi)u(t) + b_d(\xi)w_{dist}(t) \\ x(0, t) &= 0, \qquad x(1, t) = 0 \\ x(\xi, 0) &= x_0(\xi), \quad x_t(\xi, 0) = x_1(\xi) \\ y(t) &= \int_0^1 b(\xi)x(\xi, t)dt \end{aligned}$$

for some $b(\cdot), b_d(\cdot) \in L^2(0, 1; \mathbb{R})$. The uncontrolled system is unstable due to the lack of damping. Since the input and output are distributed, it is also not exponentially stabilizable or detectable, but instead it is only strongly (or polynomially [4]) stabilizable provided that $\langle b(\cdot), \sin(k\pi\cdot)\rangle_{L^2} \neq 0$ for all $k \in \mathbb{N}$. Because of this, it is not possible to use the Low-Gain Robust Controller, and the two observer based controller designs are not guaranteed to achieve closed-loop stability. However, since the system is impedance passive, the Passive Robust Controller (`PassiveRC`) can be used in achieving robust output regulation even in the absence of exponential stability as shown in [39, Sec. 5.1]. Due to the sub-exponential closed-loop stability, the regulation error does not converge with any uniform convergence rate, but instead the rate depends on the initial state of the system.

In the simulations the system is approximated using the orthonormal eigenfunctions of the undamped wave operator. Figure 4 shows example

results of the simulation for a smooth periodic reference signal with two frequencies π and 2π.

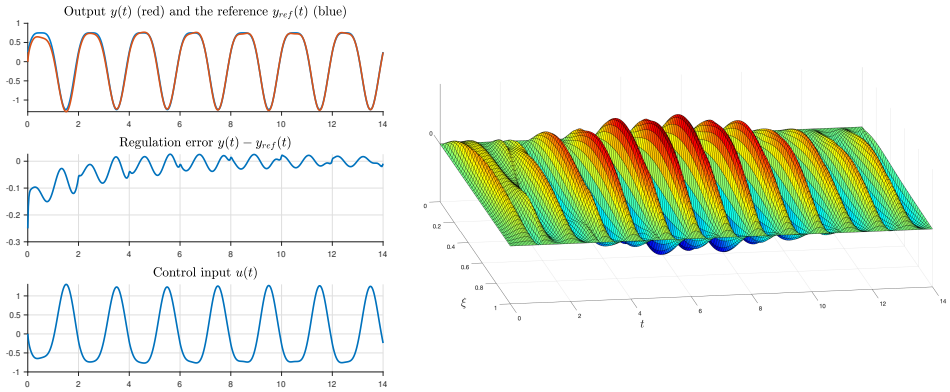

FIGURE 4. Example outputs of the periodic tracking for the 1D wave equation ("Case 2").

5.4. **The 2D Wave Equation on an Annulus.**

Main file name: `Wave2DCase1_Main.m`

This simulation example of a boundary controlled 2D wave equation is taken from the article [23]. In particular, $\Omega \subset \mathbb{R}^n$ is a bounded domain (an open connected set) with a Lipschitz-continuous boundary $\partial\Omega$ split into two parts Γ_0, Γ_1 such that $\overline{\Gamma_0} \cup \overline{\Gamma_1} = \partial\Omega$, $\Gamma_0 \cap \Gamma_1 = \emptyset$, and $\partial\Gamma_0, \partial\Gamma_1$ both have surface measure zero. The wave equation is of the form

$$\begin{aligned}
\rho(\zeta)\frac{\partial^2 w}{\partial t^2}(\zeta,t) &= \nabla\cdot\big(T(\zeta)\,\nabla w\,(\zeta,t)\big), \quad \zeta\in\Omega,\\
u(\zeta,t) &= \nu\cdot T(\zeta)\,\nabla w(\zeta,t), \quad \zeta\in\Gamma_1,\\
y(\zeta,t) &= \frac{\partial w}{\partial t}(\zeta,t), \qquad \zeta\in\Gamma_1,\\
0 &= \frac{\partial w}{\partial t}(\zeta,t), \qquad \zeta\in\Gamma_0,\ t>0\\
z(\cdot,0) &= w_0, \qquad \frac{\partial z}{\partial t}(\cdot,0) = w_1,
\end{aligned}$$

where $w(\zeta,t)$ is the displacement from the equilibrium at the point $\zeta \in \Omega$ and time $t \geq 0$, $\rho(\cdot)$ is the mass density, $T^*(\cdot) = T(\cdot) \in L^2(\Omega;\mathbb{R}^n)$ is Young's modulus and $\nu \in L^\infty(\partial\Omega;\mathbb{R}^n)$ is the unit outward normal at $\partial\Omega$. The functions $\rho(\cdot)$ and $T(\cdot)$ are essentially bounded from both above and below away from zero. In the equation, the boundary input and boundary measurement are collocated.

In particular, we consider the wave equation on an annulus $\Omega := \{\zeta \in \mathbb{R}^2 \mid 1 < \|\zeta\| < 2\}$. We choose $\partial\Omega = \Gamma_0 \cup \Gamma_1$ where $\Gamma_0 = \{\zeta \in \partial\Omega \mid \|\zeta\| = 1\}$ and $\Gamma_1 = \{\zeta \in \partial\Omega \mid \|\zeta\| = 2\}$ so that the control and the measurement are on the outer boundary of the annulus.

The output space is infinite-dimensional, $Y := \{\, h \in H^{1/2}(\partial\Omega) \mid h|_{\Gamma_0} = 0 \,\}$ which is continuously and densely embedded into $L^2(\Gamma_1)$ [52, Thm. 13.6.10].

The initial reference and disturbance signals in the example are given by

$$y_{ref}(\theta,t) = \frac{1}{2\pi^2}(\pi-\theta)^2\sin(\pi t) + \frac{1}{2}\sin\left(\frac{\theta}{2}\right)\cos(2\pi t)$$
$$w(\theta,t) = \cos(\theta)\sin(2\pi t) + \sin(\theta)\sin(\pi t).$$

The real frequencies are then $\omega_1 = \pi$, $\omega_2 = 2\pi$. Because of of the technical challenges arising from the infinite-dimensional input and outputs spaces of the model, the example is not (at the moment) written completely in "**RORPack** style". In particular, the closed-loop construction is based on expressing the signals $y_{ref}(t)$ and $w_{dist}(t)$ as outputs of an *exosystem*, and also the controller is constructed by hand in the main file of the simulation example. The controller is the "Approximate internal model based controller" introduced in the reference [23], and it is not (yet) implemented in **RORPack** as a general controller construction routine.

The numerical approximation in the example is done using the spectral method based on the eigenvalues and eigenfunctions of the undamped wave equation on the annulus. The approximate controller includes a negative definite feedthrough term D_c which corresponds to pre-stabilizing the system with negative output feedback.

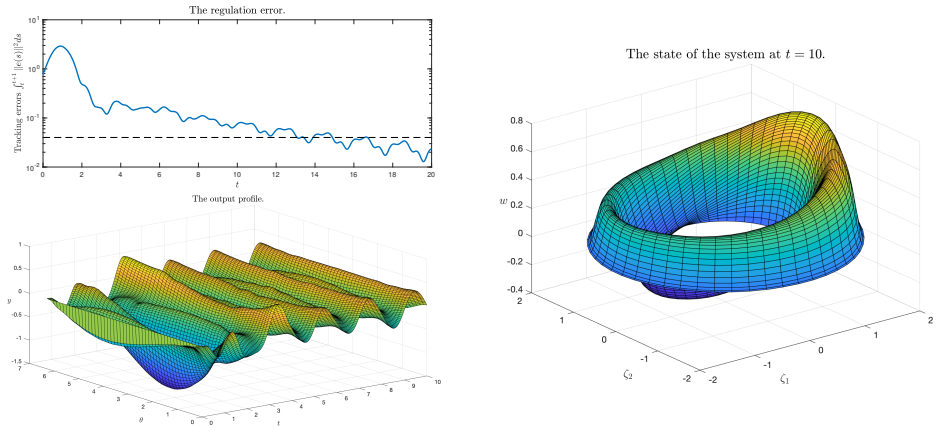

FIGURE 5. Example outputs of the approximate tracking for the 2D wave equation ("Case 1").

5.5. **The Controlled Beam Equations.** The following example study controller design for controlled beam equation models.

Example: An Euler–Bernoulli Beam with Kelvin–Voigt Damping.

Main file name: `BeamKelvinVoigt_Main.m`

The simulation example is taken from the conference article [41]. The goal is to construct a *finite-dimensional* and low-order error feedback controller for output tracking and disturbance rejection of an Euler–Bernoulli beam

equation with Kelvin-Voigt damping on $\Omega = (-1, 1)$ [28, Sec. 3],

$$\begin{aligned} v_{tt}(\xi,t) + \left(EIv_{\xi\xi} + d_{KV}Iv_{\xi\xi t}\right)_{\xi\xi}(\xi,t) + d_v v_t(\xi,t) \\ = b_1(\xi)u_1(t) + b_2(\xi)u_2(t) + B_{d0}(\xi)w_{dist}(t) \\ v(-1,t) = v_\xi(-1,t) = 0, \\ v(1,t) = v_\xi(1,t) = 0 \\ v(\xi,0) = v_0(\xi), \qquad v_t(\xi,0) = v_1(\xi), \\ y(t) = (v(\xi_1,t), v(\xi_2,t))^T. \end{aligned}$$

The parameters $E > 0$ and $I > 0$ are the (constant) elastic modulus and the second moment of area, respectively, and $d_{KV} > 0$ and $d_v \geq 0$ are the damping coefficients associated to the Kelvin–Voigt damping and viscous damping, respectively. The beam is assumed to have constant density $\rho = 1$.

The system has two control inputs $u(t) = (u_1(t), u_2(t))^T$ and two measured outputs $y(t) = (y_1(t), y_2(t))^T$. The outputs are point observations of the deflection of the beam at distinct points $\xi_1, \xi_2 \in (-1, 1)$. The input profiles $b_1(\cdot), b_2(\cdot) \in C^1(-1, 1; \mathbb{R})$ are fixed input profiles satisfying $b_j(\pm 1) = b_j'(\pm 1) = 0$, $j = 1, 2$, and the disturbance input term has the form $B_{d0}(\xi)w_{dist}(t) = \sum_{k=1}^{n_d} b_{dk}(\xi)w_{dist}^k(t)$ for $w_{dist}(t) = (w_{dist}^k(t))_{k=1}^{n_d}$ and for some fixed but unknown profile functions $b_{dk}(\cdot) \in C^1(-1, 1; \mathbb{R})$ with $b_{dk}(\pm 1) = b_{dk}'(\pm 1) = 0$ for $k \in \{1, \ldots, n_d\}$.

In the simulation example the physical parameters of the model are

$$\rho = 1, \qquad E = 10, \quad I = 1, \quad d_{KV} = 0.01, \quad d_v = 0.4.$$

The point observations are located at $\xi_1 = -0.6$ and $\xi_2 = 0.3$, and the input profiles are

$$b_1(\xi) = \frac{1}{3}(\xi + 1)^2(1 - \xi)^6, \quad \text{and} \quad b_2(\xi) = \frac{1}{3}(\xi + 1)^6(1 - \xi)^2.$$

The system has a single disturbance with a disturbance input profile $b_{d1}(\xi) = (\xi + 1)^2(1 - \xi)^2$.

Because of the Kelvin–Voigt damping, the controlled PDE model satisfies the assumptions required for Reduced Order Internal Model Based Controller design in [35]. In this example, the PDE model is approximated using a spectral Galerkin method described in [51]. This approximation scheme is used for two purposes, first of all to design the Reduced Order Observer Based Robust Controller with the routine `ObserverBasedROMRC`, and in addition to approximate the controlled PDE system in the closed-loop simulation (with a higher order approximation).

With the chosen parameters of the beam model the controlled PDE system is in fact exponentially stable (but with a fairly small stability margin). Because of this, it is possible to alternatively use the Low-Gain Robust Controller (`LowGainRC`) to achieve output tracking and disturbance rejection. Construction of this controller is included in the main file (code commented out). However, the small stability margin of the controlled PDE system also severely limits the size of the achievable closed-loop stability margin. On the other hand, in the simulation example the Reduced Order Observer Based Controller achieves a much higher closed-loop stability margin and convergence rate of the output while only having 4 additional state variables than

the Low-Gain Robust Controller (whose order is *minimal*), and with only very reasonable increase in the control input gain. For a more detailed comparison of the performances of the two controllers, you can see [41, Sec. 4], and complete similar simulations by uncommenting the construction of the Low-Gain Robust Controller in the main simulation file.

Figure 6 shows the outputs of the simulation example.

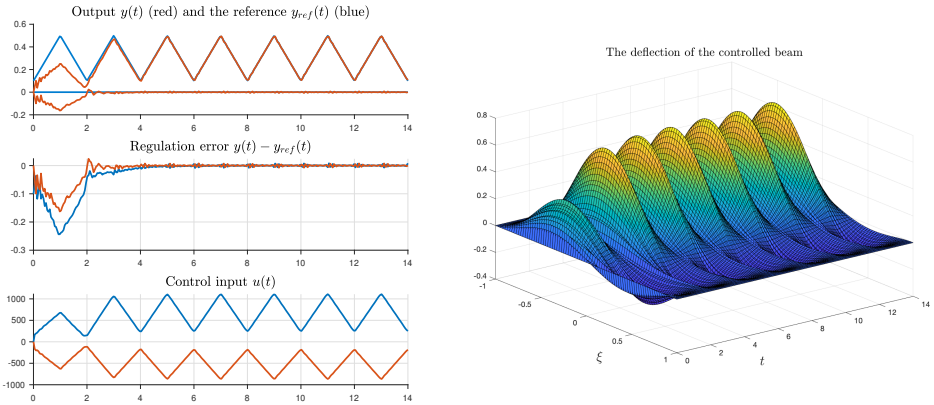

FIGURE 6. Example outputs of the controlled Kelvin–Voigt beam.

Example: A Timoshenko beam with collocated in-domain control and observation.

Main file name: `TimoshenkoLHMNC18_Main.m`

The example is related to the article [**?**]. The controlled Timoshenko beam with constant parameters ρ (mass density), I_ρ rotary moment of inertia, E Young's modulus, I moment of inertia, K shear modulus, is given by

$$\rho \frac{\partial^2 w}{\partial t^2}(\xi,t) = K\frac{\partial}{\partial \xi}\left(\frac{\partial w}{\partial \xi}(\xi,t) - \phi(\xi,t)\right) - d_w \frac{\partial w}{\partial t}(\xi,t)$$

$$I_\rho \frac{\partial^2 \phi}{\partial t^2}(\xi,t) = EI\frac{\partial^2 \phi}{\partial \xi^2} + K\left(\frac{\partial w}{\partial \xi}(\xi,t) - \phi(\xi,t)\right) - d_\phi \frac{\partial \phi}{\partial t}(\xi,t) + \chi_{[1-\eta,1]}(\xi)u(t).$$

Here $w(\xi,t)$ is the displacement of the beam at $\xi \in (0,1)$ and $\phi(\xi,t)$ is the rotation angle of the beam. The d_w and d_ϕ are viscous damping parameters. The control acts through the input profile $\chi_{[1-\eta,1]}(\cdot)$ (the characteristic function of the interval $[1-\eta,1]$ for some $\eta < 1$). The measured output (which is collocated with the control input) is given by

$$y(t) = \int_{1-\eta}^{1} \frac{\partial \phi}{\partial t}(\xi,t)dt.$$

In the simulation example the physical parameters of the model are

$$\rho = 1, \quad I_\rho = 1, \quad E = 1, \quad I = 1, \quad K = 1, \quad d_w = 2, \quad d_\phi = 2, \quad \eta = 0.2.$$

In this example, the PDE model is approximated with Finite Differences scheme. This scheme is very simple and easy to implement, but it should be noted that literature offers several more reliable and advanced numerical

approximation methods for this class of hyperbolic PDEs, and some of these are able to, for example, preserve the *port–Hamiltonian* structure of the control system.

By default, the control design used in the example is the Passive Robust Controller (`PassiveRC`), which was also the controller of choice in [40]. This controller can be used because the system is both exponentially stable and impedance passive. Because the control system has bounded input and output operators, it is also possible to use other control design methods, for example the Observer Based Robust Controller (`ObserverBasedRC`) or the Dual Observer Based Robust Controller (`DualObserverBasedRC`). The former is implemented and commented out in the code. For these alternative controller designs, the parameters K_{21} and L (the stabilizing state feedback and output injection gain) can be designed using "collocated design" as $K_{21} = -\kappa_K B^*$ and $L = -\kappa_L C^*$ for some gain parameters $\kappa_K, \kappa_L > 0$.

Figure 7 shows the outputs of the simulation example.

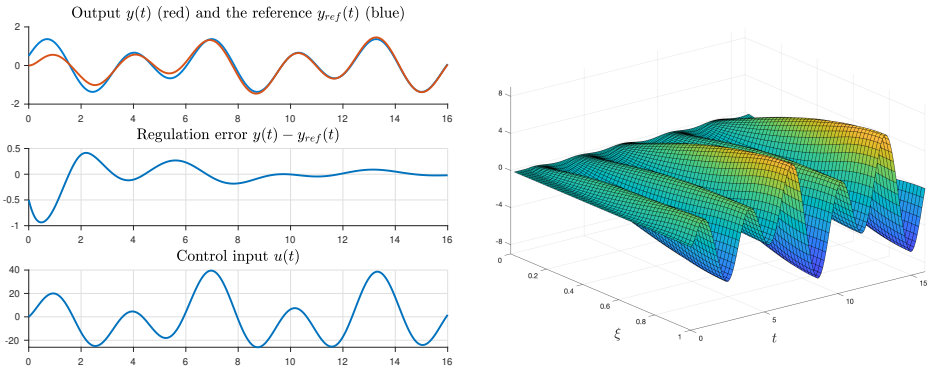

FIGURE 7. Example outputs of the controlled Timoshenko beam.

6. CONTRIBUTORS

Lassi Paunonen: Project leader, developer (2017–)
Jonne Karru: Developer (2021)
Mikko Aarnos: Developer (2018)
Jukka-Pekka Humaloja: Developer (2018–)

APPENDIX A. EXTERNAL LINKS

- https://sysgrouptampere.wordpress.com — Systems Theory Research Group at Tampere University.
- https://github.com/lassipau/rorpack/ — RORPack for Python
- https://github.com/lassipau/rorpack-matlab/ — RORPack for Matlab
- https://lassipau.github.io/rorpack/ — RORPack homepage (hosted by GitHub Pages)
- http://mathesaurus.sourceforge.net/matlab-numpy.html — Useful information on differences in the Matlab and Python/Numpy syntax for vectors and matrices.
- https://chebfun.org — Homepage of the Chebfun Project.

References

[1] Eugenio Aulisa and David Gilliam. *A Practical Guide to Geometric Regulation for Distributed Parameter Systems*. Chapman and Hall, 2015.

[2] Eugenio Aulisa and David S. Gilliam. Approximation Methods for Geometric Regulation. *arXiv e-prints*, page arXiv:2102.06196, February 2021.

[3] Eugenio Aulisa, David S. Gilliam, and Thanuka W. Pathiranage. Analysis of the error in an iterative algorithm for asymptotic regulation of linear distributed parameter control systems. *ESAIM Math. Model. Numer. Anal.*, 53(5):1577–1606, 2019.

[4] Alexander Borichev and Yuri Tomilov. Optimal polynomial decay of functions and operator semigroups. *Math. Ann.*, 347(2):455–478, 2010.

[5] S. Boulite, A. Idrissi, and A. Ould Maaloum. Robust multivariable PI-controllers for linear systems in Banach state spaces. *J. Math. Anal. Appl.*, 349(1):90–99, 2009.

[6] Christopher I. Byrnes, David S. Gilliam, Victor I. Shubov, and George Weiss. Regular linear systems governed by a boundary controlled heat equation. *J. Dyn. Control Syst.*, 8(3):341–370, 2002.

[7] Christopher I. Byrnes and Alberto Isidori. Limit sets, zero dynamics, and internal models in the problem of nonlinear output regulation. *IEEE Trans. Automat. Control*, 48(10):1712–1723, 2003.

[8] Christopher I. Byrnes and Alberto Isidori. Nonlinear internal models for output regulation. *IEEE Trans. Automat. Control*, 49(12):2244–2247, 2004.

[9] Christopher I. Byrnes, István G. Laukó, David S. Gilliam, and Victor I. Shubov. Output regulation problem for linear distributed parameter systems. *IEEE Trans. Automat. Control*, 45(12):2236–2252, 2000.

[10] Ada Cheng and Kirsten Morris. Well-posedness of boundary control systems. *SIAM J. Control Optim.*, 42(4):1244–1265, 2003.

[11] Ruth F. Curtain and Hans J. Zwart. *An Introduction to Infinite-Dimensional Linear Systems Theory*. Springer-Verlag, New York, 1995.

[12] Edward J. Davison. The robust control of a servomechanism problem for linear time-invariant multivariable systems. *IEEE Trans. Automat. Control*, 21(1):25–34, 1976.

[13] Joachim Deutscher. Output regulation for linear distributed-parameter systems using finite-dimensional dual observers. *Automatica J. IFAC*, 47(11):2468–2473, 2011.

[14] V. Dos Santos, G. Bastin, J.-M. Coron, and B. d'Andréa Novel. Boundary control with integral action for hyperbolic systems of conservation laws: stability and experiments. *Automatica J. IFAC*, 44(5):1310–1318, 2008.

[15] Tobin A. Driscoll, Nicholas Hale, and Lloyd N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014.

[16] Bruce A. Francis and W. Murray Wonham. The internal model principle for linear multivariable regulators. *Appl. Math. Optim.*, 2(2):170–194, 1975.

[17] Timo Hämäläinen and Seppo Pohjolainen. Robust control and tuning problem for distributed parameter systems. *Internat. J. Robust Nonlinear Control*, 6(5):479–500, 1996.

[18] Timo Hämäläinen and Seppo Pohjolainen. A finite-dimensional robust controller for systems in the CD-algebra. *IEEE Trans. Automat. Control*, 45(3):421–431, 2000.

[19] Timo Hämäläinen and Seppo Pohjolainen. Robust regulation of distributed parameter systems with infinite-dimensional exosystems. *SIAM J. Control Optim.*, 48(8):4846–4873, 2010.

[20] Timo Hämäläinen and Seppo Pohjolainen. A self-tuning robust regulator for infinite-dimensional systems. *IEEE Trans. Automat. Control*, 56(9):2116–2127, 2011.

[21] Jie Huang. *Nonlinear Output Regulation, Theory and Applications*. SIAM, Philadelphia, 2004.

[22] Konsta Huhtala, Lassi Paunonen, and Weiwei Hu. Robust output tracking for a room temperature model with distributed control and observation. In *Proceedings of the 24th International Symposium on Mathematical Theory of Networks and Systems*, Cambridge, UK, August 24–28 2020.

[23] J. Humaloja, M. Kurula, and L. Paunonen. Approximate robust output regulation of boundary control systems. *IEEE Trans. Automat. Control*, 64(6):2210–2223, June 2019.

[24] Jukka-Pekka Humaloja and Lassi Paunonen. Robust regulation of infinite-dimensional port-Hamiltonian systems. *IEEE Trans. Automat. Control*, 63(5):1480–1486, 2018.
[25] Eero Immonen. *State Space Output Regulation Theory for Infinite-Dimensional Linear Systems and Bounded Uniformly Continuous Exogenous Signals*. PhD thesis, Tampere University of Technology, 2006.
[26] Eero Immonen. On the internal model structure for infinite-dimensional systems: Two common controller types and repetitive control. *SIAM J. Control Optim.*, 45(6):2065–2093, 2007.
[27] Eero Immonen and Seppo Pohjolainen. Feedback and feedforward output regulation of bounded uniformly continuous signals for infinite-dimensional systems. *SIAM J. Control Optim.*, 45(5):1714–1735, 2006.
[28] Kazufumi Ito and K. A. Morris. An approximation theory of solutions to operator Riccati equations for H^∞ control. *SIAM J. Control Optim.*, 36(1):82–99, 1998.
[29] Birgit Jacob and Hans Zwart. *Linear Port-Hamiltonian Systems on Infinite-Dimensional Spaces*, volume 223 of *Operator Theory: Advances and Applications*. Birkhäuser, Basel, 2012.
[30] Hartmut Logemann and Stuart Townley. Low-gain control of uncertain regular linear systems. *SIAM J. Control Optim.*, 35(1):78–116, 1997.
[31] Hartmut Logemann and Stuart Townley. Adaptive low-gain integral control of multivariable well-posed linear systems. *SIAM J. Control Optim.*, 41(6):1722–1732, 2003.
[32] Jarmo Malinen and Olof J. Staffans. Conservative boundary control systems. *J. Differential Equations*, 231(1):290–312, 2006.
[33] Kirsten Morris. Design of finite-dimensional controllers for infinite-dimensional systems by approximation. *J. Math. Systems Estim. Control*, 4(2):1–30, 1994.
[34] V. Natarajan, D.S. Gilliam, and G. Weiss. The state feedback regulator problem for regular linear systems. *IEEE Trans. Automat. Control*, 59(10):2708–2723, 2014.
[35] L. Paunonen and D. Phan. Reduced order controller design for robust output regulation of parabolic systems. *IEEE Trans. Automat. Control*, 65(6):2480–2493, 2020.
[36] Lassi Paunonen. *Output Regulation Theory for Linear Systems with Infinite-Dimensional and Periodic Exosystems*. PhD thesis, Tampere University of Technology, 2011.
[37] Lassi Paunonen. Controller design for robust output regulation of regular linear systems. *IEEE Trans. Automat. Control*, 61(10):2974–2986, 2016.
[38] Lassi Paunonen. Robust controllers for regular linear systems with infinite-dimensional exosystems. *SIAM J. Control Optim.*, 55(3):1567–1597, 2017.
[39] Lassi Paunonen. Stability and robust regulation of passive linear systems. *SIAM J. Control Optim.*, 57(6):3827–3856, 2019.
[40] Lassi Paunonen, Yann Le Gorrec, and Héctor Ramírez. A simple robust controller for port–Hamiltonian systems. In *Proceedings of the 6th IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control*, pages 92–96, Valparaiso, Chile, May 1–4 2018.
[41] Lassi Paunonen and Duy Phan. A reduced order controller for output tracking of a Kelvin–Voigt beam. In *Proceedings of the 24th International Symposium on Mathematical Theory of Networks and Systems*, Cambridge, UK, August 24–28 2020, submitted.
[42] Lassi Paunonen and Seppo Pohjolainen. Internal model theory for distributed parameter systems. *SIAM J. Control Optim.*, 48(7):4753–4775, 2010.
[43] Lassi Paunonen and Seppo Pohjolainen. Reduced order internal models in robust output regulation. *IEEE Trans. Automat. Control*, 58(9):2307–2318, 2013.
[44] Lassi Paunonen and Seppo Pohjolainen. The internal model principle for systems with unbounded control and observation. *SIAM J. Control Optim.*, 52(6):3967–4000, 2014.
[45] Duy Phan and Lassi Paunonen. Finite-dimensional controllers for robust regulation of boundary control systems. *Math. Control Relat. Fields*, 11(1):95–117, 2021.
[46] Seppo Pohjolainen. A feedforward controller for distributed parameter systems. *Internat. J. Control*, 34(1):173–184, 1981.
[47] Seppo A. Pohjolainen. Robust multivariable PI-controller for infinite-dimensional systems. *IEEE Trans. Automat. Control*, 27(1):17–31, 1982.

[48] Richard Rebarber and George Weiss. Internal model based tracking and disturbance rejection for stable well-posed systems. *Automatica J. IFAC*, 39(9):1555–1569, 2003.
[49] Dietmar Salamon. Infinite-dimensional linear systems with unbounded control and observation: A functional analytic approach. *Trans. Amer. Math. Soc.*, 300(2):383–431, 1987.
[50] J. M. Schumacher. Finite-dimensional regulators for a class of infinite-dimensional systems. *Systems Control Lett.*, 3:7–12, 1983.
[51] Jie Shen. Efficient spectral-Galerkin method II. direct solvers of second- and fourth-order equations using Chebyshev polynomials. *SIAM J. Sci. Comput.*, 16(1):74–87, 1995.
[52] M. Tucsnak and G. Weiss. *Observation and Control for Operator Semigroups*. Birkhäuser Basel, 2009.
[53] Cheng-Zhong Xu and Hamadi Jerbi. A robust PI-controller for infinite-dimensional systems. *Internat. J. Control*, 61(1):33–45, 1995.
[54] Cheng-Zhong Xu and Gauthier Sallet. Multivariable boundary PI control and regulation of a fluid flow system. *Math. Control Relat. Fields*, 4(4):501–520, 2014.
[55] Xiaodong Xu and Stevan Dubljevic. Output and error feedback regulator designs for linear infinite-dimensional systems. *Automatica J. IFAC*, 83:170–178, 2017.

Department of Mathematics, Tampere University, PO. Box 692, 33101 Tampere, Finland

Email address: lassi.paunonen@tuni.fi