

```
In [3]: # enable logging widget
%load_ext pymor.discretizers.builtin.gui.jupyter
```

The `pymor.discretizers.builtin.gui.jupyter` module is not an IPython extension.

Practical class: pyMOR -- Model Order Reduction with Python

This Jupyter Notebook was taken from the PyMOR school 2024 crash course (August 26-30 (User meeting: August 29-30)). Please, for more information refer to their [website](#).

What is Model Order Reduction?

What is Model Order Reduction?

What is pyMOR?

pyMOR is ...

- a software library for writing **M**odel **O**rders **R**eduction applications
- in the **python** programming language.
- BSD-licensed, fork us on [GitHub](#).
- Everyone can contribute, everyone can become main developer.
- Started 2012, 27k lines of code (without tests), 10k commits.

Design Goals

- **Goal 1:** One library for algorithm development *and* large-scale applications.
 - Small NumPy/SciPy-based discretization toolkit for easy prototyping.
 - `VectorArray`, `Operator`, `Model` interfaces for seamless integration with high-performance PDE solvers.
- **Goal 2:** Unified view on MOR.
 - Implement RB and system-theoretic methods in one common language.

Implemented Algorithms

- Gram-Schmidt, POD, HAPOD
- Greedy basis generation with different extension algorithms
- Automatic (Petrov-)Galerkin projection of arbitrarily nested affine combinations of operators
- Successive constraints method
- Interpolation of arbitrary (nonlinear) operators, EI-Greedy, DEIM
- A posteriori error estimation
- System theory methods: balanced truncation, IRKA, ...
- Structure-preserving methods: symplectic MOR, second-order systems, port-Hamiltonian systems
- Data-driven methods: DMD, Loewner, ANNs, ...
- Matrix equation solvers
- Iterative linear solvers, eigenvalue computation, Cholesky QR, randomized LA, time-stepping algorithms

PDE Solvers

Official Support:

- [deal.II](#)
- [FEniCS](#)
- [NGSolve](#)
- [scikit-fem](#)

Used with:

- [DUNE](#)
- [FEniCSx](#)
- [BEST](#)
- [GridLOD](#)
- file I/O, e.g. [COMSOL](#)
- ...

Installation

pyMOR can be installed via `pip`. To follow this notebook, pyMOR should be installed with [Jupyter](#) support:

```
pip install pymor[jupyter]
```

If you follow along in the terminal / in an IDE, you should install pyMOR using the `gui` extra:

```
pip install pymor[gui]
```

We also provide [conda-forge](#) packages:

```
conda install -c conda-forge pymor
```

See [README.md](#) for details.

Hello pyMOR!

```
In [4]: import pymor
        pymor.config
```

```
Out[4]: pyMOR Version 2024.1.2
```

```
Python 3.12.7 on Linux-6.5.0-45-generic-x86_64-with-glibc2.35
```

External Packages

```
-----
DEALII:      missing
DUNEGDT:     missing
FENICS:      missing
GL:          3.1.7
IPYPARALLEL: missing
IPYTHON:     8.29.0
IPYWIDGETS:  missing
K3D:         missing
MATPLOTLIB:  3.9.2
MESHIO:      missing
MPI:         4.0.1
NGSOLVE:     missing
NUMPY:       1.26.4
PYTEST:      missing
QT:          PySide6 (Qt 6.8.0)
QTOPENGL:    present
SCIKIT_FEM:   missing
SCIPY:       1.14.1
SCIPY_LSMR:   present
SLYCOT:      0.6.0
SPHINX:      missing
TORCH:       missing
TYPER:       0.13.1
VTKIO:       missing
```

Defaults

```
-----
```

```
See pymor.core.defaults.print_defaults.
```

```
In [5]: from pymor.basic import *
        print_defaults()
```

```
01:46 |WARNING|_import_all: Failed to import pymor.discretizers.fenics
```

pyMOR defaults

=====

path (shortened)	value
source	
-----	-----

basic.almost_equal.atol	1e-14
code	
basic.almost_equal.rtol	1e-14
code	
bfgs.error_aware_bfgs.maxiter	100
code	
bfgs.error_aware_bfgs.miniter	0
code	
bfgs.error_aware_bfgs.rtol_mu	1e-16
code	
bfgs.error_aware_bfgs.rtol_output	1e-16
code	
bfgs.error_aware_bfgs.stagnation_thr	inf
code	
eshold	
bfgs.error_aware_bfgs.stagnation_win	3
code	
dow	
bfgs.error_aware_bfgs.tol_sub	1e-08
code	
dmd.dmd.svd_method	'method_of_snapshots'
code	
genericsolvers.apply_inverse.check_f	True
code	
inite	
genericsolvers.apply_inverse.default	'generic_least_squares_lsmr'
code	
_least_squares_solver	
genericsolvers.apply_inverse.default	'generic_lgmres'
code	
_solver	
genericsolvers.solver_options.least_	1e-06
code	
squares_lsmr_atol	
genericsolvers.solver_options.least_	1e-06
code	
squares_lsmr_btol	
genericsolvers.solver_options.least_	100000000.0
code	
squares_lsmr_conlim	
genericsolvers.solver_options.least_	0.0
code	
squares_lsmr_damp	
genericsolvers.solver_options.least_	None
code	
squares_lsmr_maxiter	
genericsolvers.solver_options.least_	False
code	
squares_lsmr_show	

```

genericsolvers.solver_options.least_ 1e-06
code
squares_lsqr_atol
genericsolvers.solver_options.least_ 1e-06
code
squares_lsqr_btol
genericsolvers.solver_options.least_ 100000000.0
code
squares_lsqr_conlim
genericsolvers.solver_options.least_ None
code
squares_lsqr_iter_lim
genericsolvers.solver_options.least_ False
code
squares_lsqr_show
genericsolvers.solver_options.lgmres 39
code
_inner_m
genericsolvers.solver_options.lgmres 1000
code
_maxiter
genericsolvers.solver_options.lgmres 3
code
_outer_k
genericsolvers.solver_options.lgmres 1e-05
code
_tol
gram_schmidt.gram_schmidt.atol 1e-13
code
gram_schmidt.gram_schmidt.check True
code
gram_schmidt.gram_schmidt.check_tol 0.001
code
gram_schmidt.gram_schmidt.reiterate True
code
gram_schmidt.gram_schmidt.reiteratio 0.9
code
n_threshold
gram_schmidt.gram_schmidt.rtol 1e-13
code
line_search._armijo.alpha_init 1.0
code
line_search._armijo.beta 0.0001
code
line_search._armijo.maxiter 10
code
line_search._armijo.tau 0.5
code
line_search.armijo.alpha_init 1.0
code
line_search.armijo.beta 0.0001
code
line_search.armijo.maxiter 10
code
line_search.armijo.tau 0.5
code

```

```

line_search.constrained_armijo.alpha  1.0
code
_init
line_search.constrained_armijo.beta   0.0001
code
line_search.constrained_armijo.maxit  10
code
er
line_search.constrained_armijo.tau    0.5
code
lradi.lyap_lrcf_solver_options.lradi  500
code
_maxiter
lradi.lyap_lrcf_solver_options.lradi  'projection_shifts'
code
_shifts
lradi.lyap_lrcf_solver_options.lradi  1e-10
code
_tol
lradi.lyap_lrcf_solver_options.proje  20
code
ction_shifts_init_maxiter
lradi.lyap_lrcf_solver_options.proje  6
code
ction_shifts_subspace_columns
lradi.lyap_lrcf_solver_options.wachs  50
code
press_large_ritz_num
lradi.lyap_lrcf_solver_options.wachs  25
code
press_small_ritz_num
lradi.lyap_lrcf_solver_options.wachs  1e-10
code
press_tol
lrradi.ricc_lrcf_solver_options.hami  20
code
ltonian_shifts_init_maxiter
lrradi.ricc_lrcf_solver_options.hami  6
code
ltonian_shifts_subspace_columns
lrradi.ricc_lrcf_solver_options.lrra  500
code
di_maxiter
lrradi.ricc_lrcf_solver_options.lrra  'hamiltonian_shifts'
code
di_shifts
lrradi.ricc_lrcf_solver_options.lrra  1e-10
code
di_tol
lyapunov.mat_eqn_sparse_min_size.val  1000
code
ue
lyapunov.solve_cont_lyap_dense.defau  'slycot'
code
lt_solver_backend
lyapunov.solve_cont_lyap_lrcf.defaul  'slycot'

```

```

code
t_dense_solver_backend
lyapunov.solve_cont_lyap_lrcf.default 'lradi'
code
t_sparse_solver_backend
lyapunov.solve_disc_lyap_dense.defau 'slycot'
code
lt_solver_backend
lyapunov.solve_disc_lyap_lrcf.default 'slycot'
code
t_dense_solver_backend
newton.newton.atol 0.0
code
newton.newton.maxiter 100
code
newton.newton.miniter 0
code
newton.newton.relax 'armijo'
code
newton.newton.rtol 1e-07
code
newton.newton.stagnation_threshold inf
code
newton.newton.stagnation_window 3
code
pod.pod.atol 0.0
code
pod.pod.l2_err 0.0
code
pod.pod.method 'method_of_snapshots'
code
pod.pod.orth_tol 1e-10
code
pod.pod.rtol 1e-07
code
rand_la.RandomizedRangeFinder.__init 1e-15
code
__.failure_tolerance
rand_la.RandomizedRangeFinder.__init 20
code
__.num_testvecs
rand_la.adaptive_rrf.failure_toleran 1e-15
code
ce
rand_la.adaptive_rrf.num_testvecs 20
code
rand_la.adaptive_rrf.tol 0.0001
code
rand_la.random_generalized_svd.modes 6
code
rand_la.random_generalized_svd.p 20
code
rand_la.random_generalized_svd.q 2
code
rand_la.random_ghep.modes 6
code

```

rand_la.random_ghep.p	20
code	
rand_la.random_ghep.q	2
code	
rand_la.randomized_ghep.n	6
code	
rand_la.randomized_ghep.oversampling	20
code	
rand_la.randomized_ghep.power_iterat	0
code	
ions	
rand_la.randomized_svd.oversampling	20
code	
rand_la.randomized_svd.power_iterati	0
code	
ons	
rand_la.rrf.l	8
code	
rand_la.rrf.q	2
code	
riccati.solve_pos_ricc_dense.default	'slycot'
code	
_solver_backend	
riccati.solve_pos_ricc_lrcf.default_	'slycot'
code	
dense_solver_backend	
riccati.solve_ricc_dense.default_sol	'slycot'
code	
ver_backend	
riccati.solve_ricc_lrcf.default_dens	'slycot'
code	
e_solver_backend	
riccati.solve_ricc_lrcf.default_spar	'lrradi'
code	
se_solver_backend	
samdp.samdp.conjtol	1e-08
code	
samdp.samdp.dorqitol	0.0001
code	
samdp.samdp.imagtol	1e-06
code	
samdp.samdp.init_shifts	None
code	
samdp.samdp.krestart	20
code	
samdp.samdp.maxrestart	100
code	
samdp.samdp.rqi_maxiter	10
code	
samdp.samdp.rqitol	1e-10
code	
samdp.samdp.tol	1e-10
code	
samdp.samdp.which	'NR'
code	
scm.LBSuccessiveConstraintsFunctiona	'highs'


```

code
l.__init__.linprog_method
svd_va.method_of_snapshots.atol      0.0
code
svd_va.method_of_snapshots.l2_err    0.0
code
svd_va.method_of_snapshots.rtol      1e-07
code
svd_va.qr_svd.atol                   0.0
code
svd_va.qr_svd.l2_err                 0.0
code
svd_va.qr_svd.rtol                   4e-08
code
symplectic.symplectic_gram_schmidt.a 1e-13
code
tol
symplectic.symplectic_gram_schmidt.c True
code
heck
symplectic.symplectic_gram_schmidt.c 0.001
code
heck_tol
symplectic.symplectic_gram_schmidt.r True
code
eiterate
symplectic.symplectic_gram_schmidt.r 0.9
code
eiteration_threshold
symplectic.symplectic_gram_schmidt.r 1e-13
code
tol
tr.trust_region.beta                 0.95
code
tr.trust_region.maxiter               30
code
tr.trust_region.maxiter_subproblem    400
code
tr.trust_region.miniter               0
code
tr.trust_region.miniter_subproblem    0
code
tr.trust_region.radius                0.1
code
tr.trust_region.radius_tol            0.75
code
tr.trust_region.rtol_mu               1e-16
code
tr.trust_region.rtol_output           1e-16
code
tr.trust_region.shrink_factor         0.5
code
tr.trust_region.stagnation_threshold  inf
code
tr.trust_region.stagnation_window     3
code

```

tr.trust_region.tol_criticality	1e-06
code	
tr.trust_region.tol_sub	1e-08
code	
text.text_problem.font_name	None
code	
pymor.auto_load_jupyter_extension.enabled	True
code	
scipy.apply_inverse.check_finite	True
code	
scipy.apply_inverse.default_least_squares_solver	'scipy_least_squares_lsqr'
code	
scipy.apply_inverse.default_solver	'scipy_spsolve'
code	
scipy.solver_options.bicgstab_maxiter	None
code	
scipy.solver_options.bicgstab_tol	1e-15
code	
scipy.solver_options.least_squares_lsqr_atol	1e-06
code	
scipy.solver_options.least_squares_lsqr_btol	1e-06
code	
scipy.solver_options.least_squares_lsqr_conlim	100000000.0
code	
scipy.solver_options.least_squares_lsqr_damp	0.0
code	
scipy.solver_options.least_squares_lsqr_maxiter	None
code	
scipy.solver_options.least_squares_lsqr_show	False
code	
scipy.solver_options.least_squares_lsqr_atol	1e-06
code	
scipy.solver_options.least_squares_lsqr_btol	1e-06
code	
scipy.solver_options.least_squares_lsqr_conlim	100000000.0
code	
scipy.solver_options.least_squares_lsqr_iter_lim	None
code	
scipy.solver_options.least_squares_lsqr_iter_lim	False
code	
scipy.solver_options.lgmres_inner_maxiter	39
code	

```

scipy.solver_options.lgmres_maxiter 1000
code
scipy.solver_options.lgmres_outer_k 3
code
scipy.solver_options.lgmres_tol 1e-05
code
scipy.solver_options.spilu_drop_rule None
code
scipy.solver_options.spilu_drop_tol 0.0001
code
scipy.solver_options.spilu_fill_fact 10
code
or
scipy.solver_options.spilu_permc_spe 'COLAMD'
code
c
scipy.solver_options.spsolve_keep_fa True
code
ctorization
scipy.solver_options.spsolve_permc_s 'COLAMD'
code
pec
cache.default_regions.disk_max_size 1073741824
code
cache.default_regions.disk_path '/tmp/pymor.cache.dayron'
code
cache.default_regions.memory_max_key 1000
code
s
cache.default_regions.persistent_max 1073741824
code
_size
cache.default_regions.persistent_pat '/tmp/pymor.persistent.cache.dayron'
code
h
logger.default_handler.filename None
code
logger.getLogger.filename None
code
logger.set_log_format.block_timings False
code
logger.set_log_format.indent_blocks True
code
logger.set_log_format.max_hierarchy_ 1
code
level
logger.set_log_levels.levels None
code
builtin.fv.jacobian_options.delta 1e-07
code
builtin.gui.jupyter.get_visualizer.b 'prefer_k3d'
code
ackend
builtin.gui.qt.background_visualizat 'ipython_if_possible'
code
ion_method.method

```

```

builtin.gui.qt.gl.colormap_texture.n    'viridis'
code
ame
builtin.gui.visualizers.OnedVisualiz    'jupyter_or_matplotlib'
code
er.__init__.backend
builtin.gui.visualizers.PatchVisuali    'jupyter_or_gl'
code
zer.__init__.backend
iosys.LTIModel.hinf_norm.tol            1e-10
code
iosys.LTIModel.linf_norm.tol            1e-10
code
iosys.SecondOrderModel.hinf_norm.tol    1e-10
code
iosys.sparse_min_size.value             1000
code
constructions.induced_norm.raise_neg    True
code
ative
constructions.induced_norm.tol          1e-10
code
interface.as_array_max_length.value     100
code
numpy.NumpyMatrixOperator.apply_inve    True
code
rse.check_cond
numpy.NumpyMatrixOperator.apply_inve    True
code
rse.check_finite
numpy.NumpyMatrixOperator.apply_inve    'scipy'
code
rse.default_sparse_solver_backend
default.new_parallel_pool.allow_mpi      True
code
default.new_parallel_pool.ipython_nu    None
code
m_engines
default.new_parallel_pool.ipython_pr     None
code
ofile
basic.ProjectionBasedReductor.__init     True
code
__.check_orthonormality
basic.ProjectionBasedReductor.__init     0.001
code
__.check_tol
floatcmp.almost_less.atol               1e-14
code
floatcmp.almost_less.rtol               1e-14
code
floatcmp.float_cmp.atol                 1e-14
code
floatcmp.float_cmp.rtol                 1e-14
code
formatrepr.format_repr.max_width        120

```

```

code
formatrepr.format_repr.verbosity      1
code
mpi.event_loop_settings.auto_launch    True
code
plot.adaptive.angle_tol                 2
code
plot.adaptive.aspect_ratio              1.3333333333333333
code
plot.adaptive.initial_num               10
code
plot.adaptive.max_num                   2000
code
plot.adaptive.min_rel_dist               0.01
code
plot.adaptive.xscale                    'linear'
code
plot.adaptive.yscale                    'linear'
code
pprint.format_array.compact_print       False
code
random.new_rng.seed_seq                 42
code
interface.VectorArray.norm2.raise_co    True
code
mplex
interface.VectorArray.norm2.tol         1e-10
code

```

Setting defaults

We need to disable WebGL-based visualizations, as they render incorrectly in RISE:

```
In [6]: set_defaults({'pymor.discretizers.builtin.gui.jupyter.get_visualizer.backend': 'none'})
```

Subpackages of the pymor Package

pymor.algorithms	generic algorithms
pymor.analyticalproblems	problem descriptions for use with discretizers
pymor.bindings	bindings to external solvers
pymor.core	base classes/caching/defaults/logging
pymor.discretizers	create Models from analyticalproblems
pymor.models	Model interface/implementations
pymor.operators	Operator interface/constructions

<code>pymor.parallel</code>	<code>WorkerPools</code> for parallelization
<code>pymor.parameters</code>	parameter support/ <code>ParameterFunctionals</code>
<code>pymor.reductors</code>	most MOR algorithms (rest in <code>pymor.algorithms</code>)
<code>pymor.scripts</code>	executable scripts (<code>pymor-demo</code> , internal scripts)
<code>pymor.tools</code>	non MOR-specific support code (pprint/floatcmp, ...)
<code>pymor.vectorarrays</code>	<code>VectorArray</code> interface/implementations

Example: Reduced Basis Method for Elliptic Problem

- Snapshot- and projection-based MOR method for parameterized (PDE) models.
- Classic theory for elliptic parameter-separable problems.
- Extendable to non-linear/time-dependent/systems.
- Details: lecture tomorrow morning.

Thermal-block problem

Find $u(x, \mu)$ such that

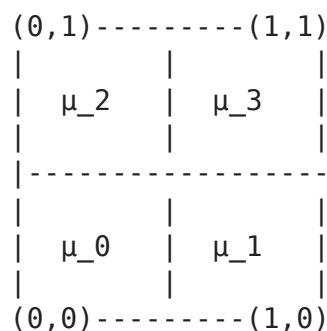
$$-\nabla \cdot [d(x, \mu) \nabla u(x, \mu)] = 1 \quad x \in \Omega, \mu \in \mathcal{P} \quad (1)$$

$$u(x, \mu) = 0 \quad x \in \partial\Omega, \mu \in \mathcal{P} \quad (2)$$

where $\Omega := [0, 1]^2 = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$

$$d(x, \mu) \equiv \mu_i \text{ const.} \quad x \in \Omega_i$$

and $\mu \in [\mu_{\min}, \mu_{\max}]^4$



After discretization, we get parameterized FOM:

$$(L + \mu_0 L_0 + \mu_1 L_1 + \mu_2 L_2 + \mu_3 L_3)u(\mu) = f$$

where $L, L_i \in \mathbb{R}^{n \times n}$, $u(\mu), f \in \mathbb{R}^n$.

Thermal-block problem in pyMOR

The thermal-block problem one of the (currently two) pre-defined FOMs in pyMOR:

```
In [16]: from pymor.models.examples import thermal_block_example  
fom_tb = thermal_block_example()
```

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...

32:56 **DiffusionOperatorP1**: Determine global dofs ...

32:56 **DiffusionOperatorP1**: Boundary treatment ...

32:56 **DiffusionOperatorP1**: Assemble system matrix ...

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...

32:56 **DiffusionOperatorP1**: Determine global dofs ...

32:56 **DiffusionOperatorP1**: Boundary treatment ...

32:56 **DiffusionOperatorP1**: Assemble system matrix ...

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...

32:56 **DiffusionOperatorP1**: Determine global dofs ...

32:56 **DiffusionOperatorP1**: Boundary treatment ...

32:56 **DiffusionOperatorP1**: Assemble system matrix ...

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...

32:56 **DiffusionOperatorP1**: Determine global dofs ...

32:56 **DiffusionOperatorP1**: Boundary treatment ...

32:56 **DiffusionOperatorP1**: Assemble system matrix ...

32:56 **L2ProductP1**: Integrate the products of the shape functions on each element

32:56 **L2ProductP1**: Determine global dofs ...

32:56 **L2ProductP1**: Boundary treatment ...

32:56 **L2ProductP1**: Assemble system matrix ...

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...

32:56 **DiffusionOperatorP1**: Determine global dofs ...

32:56 **DiffusionOperatorP1**: Boundary treatment ...

32:56 **DiffusionOperatorP1**: Assemble system matrix ...

32:56 **L2ProductP1**: Integrate the products of the shape functions on each element

32:56 **L2ProductP1**: Determine global dofs ...

32:56 **L2ProductP1**: Boundary treatment ...

32:56 **L2ProductP1**: Assemble system matrix ...

32:56 **DiffusionOperatorP1**: Calculate gradients of shape functions transformed by reference map ...

32:56 **DiffusionOperatorP1**: Calculate all local scalar products between gradients ...


```
32:56 DiffusionOperatorP1: Determine global dofs ...  
32:56 DiffusionOperatorP1: Boundary treatment ...  
32:56 DiffusionOperatorP1: Assemble system matrix ...
```

The FOM

`fom_tb` is an instance of `StationaryModel`, which encodes the mathematical structure of the model through its `Operators` :

```
In [17]: fom_tb
```

```

Out[17]: StationaryModel(
  LincombOperator(
    (NumpyMatrixOperator(<20201x20201 sparse, 400 nnz>, source_id='STATE', range_id='STATE', name='boundary_part'),
      NumpyMatrixOperator(<20201x20201 sparse, 24801 nnz>, source_id='STATE', range_id='STATE', name='diffusion_0'),
      NumpyMatrixOperator(<20201x20201 sparse, 24801 nnz>, source_id='STATE', range_id='STATE', name='diffusion_1'),
      NumpyMatrixOperator(<20201x20201 sparse, 24801 nnz>, source_id='STATE', range_id='STATE', name='diffusion_2'),
      NumpyMatrixOperator(<20201x20201 sparse, 24801 nnz>, source_id='STATE', range_id='STATE', name='diffusion_3')),
    (1.0,
      ProjectionParameterFunctional('diffusion', size=4, index=0, name='diffusion_0_0'),
      ProjectionParameterFunctional('diffusion', size=4, index=1, name='diffusion_1_0'),
      ProjectionParameterFunctional('diffusion', size=4, index=2, name='diffusion_0_1'),
      ProjectionParameterFunctional('diffusion', size=4, index=3, name='diffusion_1_1')),
    name='ellipticOperator'),
    NumpyMatrixOperator(<20201x1 dense>, range_id='STATE'),
    output_functional=ZeroOperator(NumpyVectorSpace(0), NumpyVectorSpace(20201, id='STATE')),
    products={h1: NumpyMatrixOperator(<20201x20201 sparse, 140601 nnz>, source_id='STATE', range_id='STATE'),
      h1_semi: NumpyMatrixOperator(<20201x20201 sparse, 100201 nnz>, source_id='STATE', range_id='STATE', name='h1_semi'),
      l2: NumpyMatrixOperator(<20201x20201 sparse, 140601 nnz>, source_id='STATE', range_id='STATE', name='l2'),
      h1_0: NumpyMatrixOperator(<20201x20201 sparse, 137417 nnz>, source_id='STATE', range_id='STATE'),
      h1_0_semi: NumpyMatrixOperator(<20201x20201 sparse, 98609 nnz>, source_id='STATE', range_id='STATE', name='h1_0_semi'),
      l2_0: NumpyMatrixOperator(<20201x20201 sparse, 137417 nnz>, source_id='STATE', range_id='STATE', name='l2_0')},
    visualizer=PatchVisualizer(
      TriaGrid(num_intervals=(100, 100), domain=array([[0, 0], [1, 1]])),
      bounding_box=array([[0, 0], [1, 1]]),
      backend='jupyter'),
    output_d_mu_use_adjoint=True,
    name='ThermalBlock((2, 2))_CG')

```

Solving the FOM

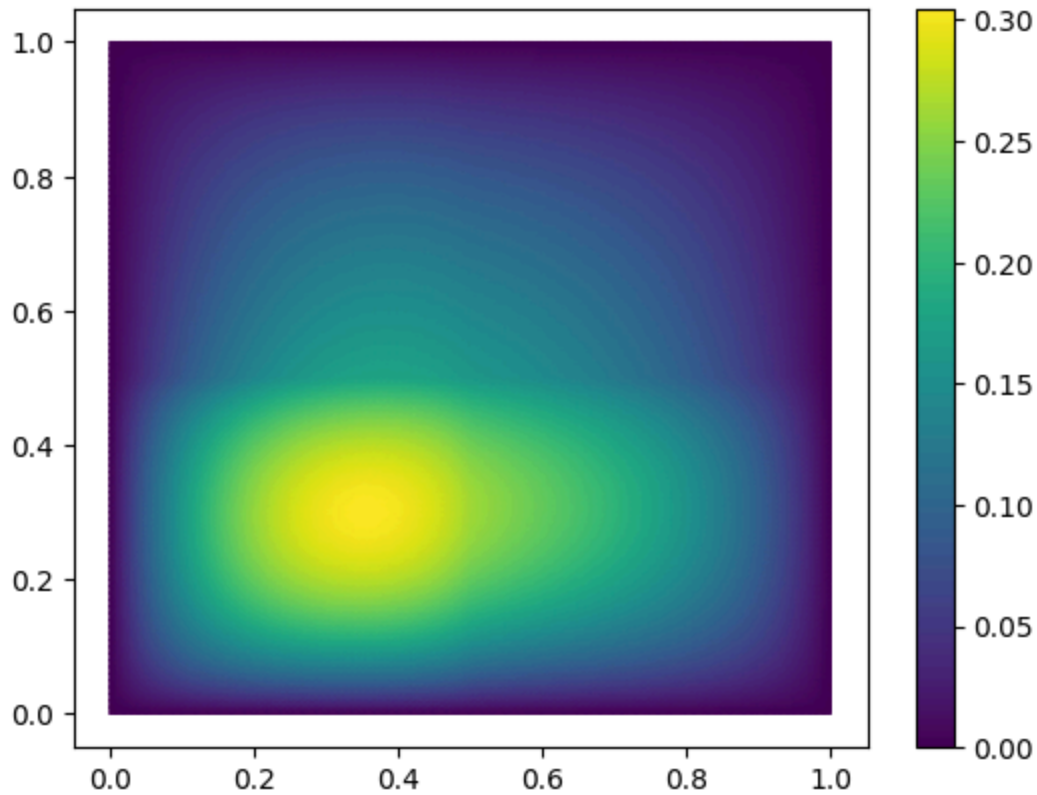
Let us compute and show the solution for particular parameter values:

```

In [18]: mu = [0.1, 0.2, 0.5, 1]
         U = fom_tb.solve(mu)
         fom_tb.visualize(U)

```

39:26 **StationaryModel**: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.1, 0.2, 0.5, 1.0], input: } ...



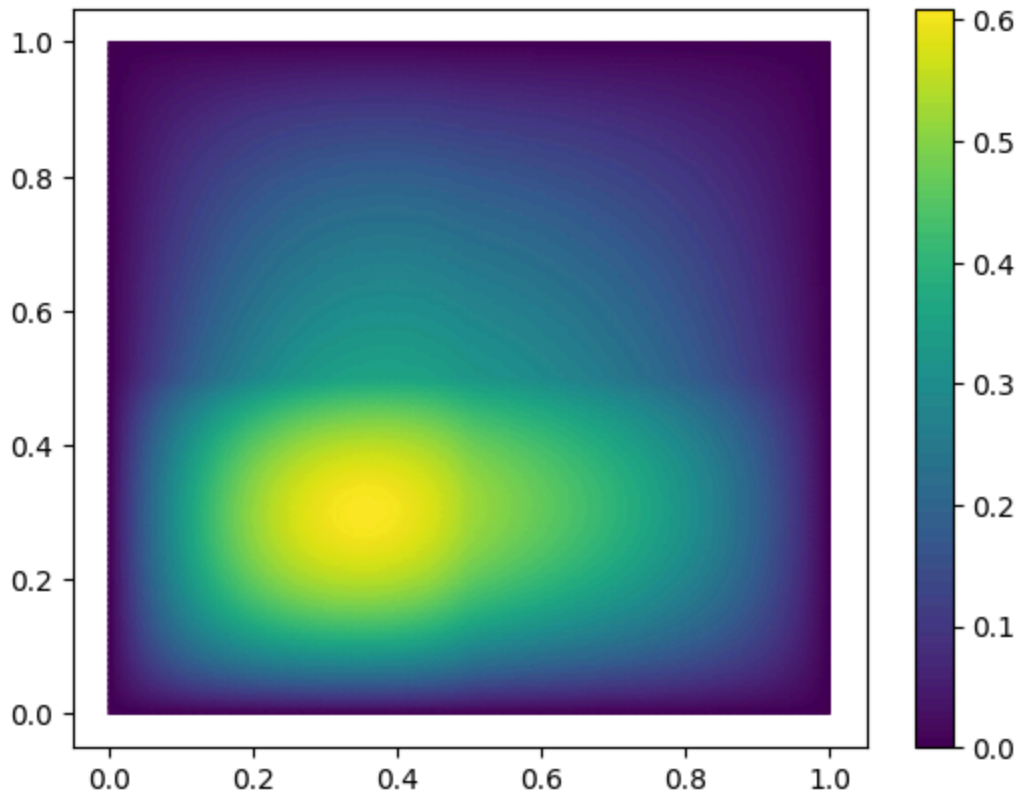
Your turn

- Solve the FOM for another set of parameter values.
- Visualize the new solution along the old one.
- Also show the difference between both solutions.

Hint: You can pass a tuple of solutions `visualize`.

```
In [31]: mu2 = [0.05, 0.1, 0.25, 0.5]
         U = fom_tb.solve(mu2)
         fom_tb.visualize(U)
```

```
02:53:48 StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for
{diffusion: [0.05, 0.1, 0.25, 0.5], input: } ...
```



Choosing a redutor

To build the reduced-order model (ROM), we need to create a `reductor`:

```
In [25]: from pymor.parameters.functionals import ExpressionParameterFunctional
from pymor.reducers.coercive import CoerciveRBReductor

reductor = CoerciveRBReductor(
    fom_tb,
    product=fom_tb.h1_0_semi_product,
    coercivity_estimator=ExpressionParameterFunctional('min(diffusion)',
                                                    fom_tb.parameters)
)
```

- The `reductor` takes care of projecting the FOM to a ROM.
- It also build an error estimator, for which `product` and `coercivity_estimator` is needed.

Basis generation

We need to compute a reduced space, onto which `reductor` projects. We use a greedy algorithm for that:

```
In [26]: from pymor.algorithms.greedy import rb_greedy

parameter_space = fom_tb.parameters.space(0.1, 1)
```

```
greedy_data = rb_greedy(fom_tb, reductor, parameter_space.sample_randomly(16,
                                                                           rtol=1e-2))
rom_tb = greedy_data['rom']
```

```

50:18 weak_greedy: Started greedy search on training set of size 1000.
50:18 weak_greedy: Estimating errors ...
50:18 |   RBSurrogate: Reducing ...
50:18 |   |   CoerciveRBReductor: Operator projection ...
50:18 |   |   CoerciveRBReductor: Assembling error estimator ...
50:18 |   |   |   ResidualReductor: Estimating residual range ...
50:18 |   |   |   |   estimate_image_hierarchical: Estimating image for basis vector -1 ...
50:18 |   |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:18 |   |   |   ResidualReductor: Projecting residual operator ...
50:18 |   |   CoerciveRBReductor: Building ROM ...
50:18 weak_greedy: Maximum error after 0 extensions: 1.865862036741425 (mu = {diffusion: [0.14752639126009934, 0.10046686974913498, 0.4926849937248369, 0.7971614931630068]})
50:18 weak_greedy: Extending surrogate for mu = {diffusion: [0.14752639126009934, 0.10046686974913498, 0.4926849937248369, 0.7971614931630068]} ...
50:18 |   RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.14752639126009934, 0.10046686974913498, 0.4926849937248369, 0.7971614931630068]} ...
50:18 |   |   StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.14752639126009934, 0.10046686974913498, 0.4926849937248369, 0.7971614931630068], input: } ...
50:18 |   RBSurrogate: Extending basis with solution snapshot ...
50:18 |   RBSurrogate: Reducing ...
50:18 |   |   CoerciveRBReductor: Operator projection ...
50:18 |   |   CoerciveRBReductor: Assembling error estimator ...
50:18 |   |   |   ResidualReductor: Estimating residual range ...
50:18 |   |   |   |   estimate_image_hierarchical: Estimating image for basis vector 0 ...
50:18 |   |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:18 |   |   |   |   |   gram_schmidt: Removing vector 1 of norm 0.0
50:18 |   |   |   |   |   gram_schmidt: Orthonormalizing vector 2 again
50:18 |   |   |   |   |   gram_schmidt: Orthonormalizing vector 3 again
50:18 |   |   |   |   |   gram_schmidt: Orthonormalizing vector 4 again
50:19 |   |   |   |   |   gram_schmidt: Removing linearly dependent vector 5
50:19 |   |   |   ResidualReductor: Projecting residual operator ...
50:19 |   |   CoerciveRBReductor: Building ROM ...

50:19 weak_greedy: Estimating errors ...
50:19 weak_greedy: Maximum error after 1 extensions: 1.6019936919981914 (mu = {diffusion: [0.10173673708600552, 0.9576359115634129, 0.11798193180683139, 0.47672751299908034]})
50:19 weak_greedy: Extending surrogate for mu = {diffusion: [0.10173673708600552, 0.9576359115634129, 0.11798193180683139, 0.47672751299908034]} ...
50:19 |   RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.10173673708600552, 0.9576359115634129, 0.11798193180683139, 0.47672751299908034]} ...
50:19 |   |   StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.10173673708600552, 0.9576359115634129, 0.11798193180683139, 0.47672751299908034], input: } ...
50:19 |   RBSurrogate: Extending basis with solution snapshot ...
50:19 |   |   gram_schmidt: Orthonormalizing vector 1 again
50:19 |   RBSurrogate: Reducing ...
50:19 |   |   CoerciveRBReductor: Operator projection ...
50:19 |   |   CoerciveRBReductor: Assembling error estimator ...
50:19 |   |   |   ResidualReductor: Estimating residual range ...

```

```

50:19 | | | | estimate_image_hierarchical: Estimating image for basis vector 1 ...
50:19 | | | | estimate_image_hierarchical: Orthonormalizing ...
50:19 | | | | gram_schmidt: Removing vector 4 of norm 7.624028394151657e-18
50:19 | | | | gram_schmidt: Orthonormalizing vector 5 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 6 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 7 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 8 again
50:19 | | | ResidualReductor: Projecting residual operator ...
50:19 | | CoerciveRBReductor: Building ROM ...

50:19 weak_greedy: Estimating errors ...
50:19 weak_greedy: Maximum error after 2 extensions: 1.4393499963676994 (mu = {diffusion: [0.6906421353947412, 0.1325464394452266, 0.10488685071583043, 0.14649212531295905]})
50:19 weak_greedy: Extending surrogate for mu = {diffusion: [0.6906421353947412, 0.1325464394452266, 0.10488685071583043, 0.14649212531295905]} ...
50:19 | RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.6906421353947412, 0.1325464394452266, 0.10488685071583043, 0.14649212531295905]} ...
50:19 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.6906421353947412, 0.1325464394452266, 0.10488685071583043, 0.14649212531295905], input: } ...
50:19 | RBSurrogate: Extending basis with solution snapshot ...
50:19 | | gram_schmidt: Orthonormalizing vector 2 again
50:19 | RBSurrogate: Reducing ...
50:19 | | CoerciveRBReductor: Operator projection ...
50:19 | | CoerciveRBReductor: Assembling error estimator ...
50:19 | | | ResidualReductor: Estimating residual range ...
50:19 | | | estimate_image_hierarchical: Estimating image for basis vector 2 ...
50:19 | | | | estimate_image_hierarchical: Orthonormalizing ...
50:19 | | | | gram_schmidt: Removing vector 8 of norm 3.2816200392025538e-18
50:19 | | | | gram_schmidt: Orthonormalizing vector 9 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 10 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 11 again
50:19 | | | | gram_schmidt: Orthonormalizing vector 12 again
50:19 | | | ResidualReductor: Projecting residual operator ...
50:20 | | CoerciveRBReductor: Building ROM ...

50:20 weak_greedy: Estimating errors ...
50:20 weak_greedy: Maximum error after 3 extensions: 1.0958879346945165 (mu = {diffusion: [0.10387519810091078, 0.6140506258501669, 0.9946699952071406, 0.13477987884419615]})
50:20 weak_greedy: Extending surrogate for mu = {diffusion: [0.10387519810091078, 0.6140506258501669, 0.9946699952071406, 0.13477987884419615]} ...
50:20 | RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.10387519810091078, 0.6140506258501669, 0.9946699952071406, 0.13477987884419615]} ...
50:20 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.10387519810091078, 0.6140506258501669, 0.9946699952071406, 0.13477987884419615], input: } ...
50:20 | RBSurrogate: Extending basis with solution snapshot ...
50:20 | | gram_schmidt: Orthonormalizing vector 3 again

```

```

50:20 | RBSurrogate: Reducing ...
50:20 | | CoerciveRBReductor: Operator projection ...
50:20 | | CoerciveRBReductor: Assembling error estimator ...
50:20 | | | ResidualReductor: Estimating residual range ...
50:20 | | | | estimate_image_hierarchical: Estimating image for basis vector 3 ...
50:20 | | | | estimate_image_hierarchical: Orthonormalizing ...
50:20 | | | | gram_schmidt: Removing vector 12 of norm 5.012430166665421e-18
50:20 | | | | gram_schmidt: Orthonormalizing vector 13 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 14 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 15 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 16 again
50:20 | | | ResidualReductor: Projecting residual operator ...
50:20 | | CoerciveRBReductor: Building ROM ...

```

```

50:20 weak_greedy: Estimating errors ...
50:20 weak_greedy: Maximum error after 4 extensions: 1.0152289685620326 (mu = {diffusion: [0.556563431423839, 0.9925970283358396, 0.10360684426000555, 0.11492182170780124]})
50:20 weak_greedy: Extending surrogate for mu = {diffusion: [0.556563431423839, 0.9925970283358396, 0.10360684426000555, 0.11492182170780124]} ...
50:20 | RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.556563431423839, 0.9925970283358396, 0.10360684426000555, 0.11492182170780124]} ...
50:20 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.556563431423839, 0.9925970283358396, 0.10360684426000555, 0.11492182170780124], input: } ...
50:20 | RBSurrogate: Extending basis with solution snapshot ...
50:20 | | gram_schmidt: Orthonormalizing vector 4 again
50:20 | RBSurrogate: Reducing ...
50:20 | | CoerciveRBReductor: Operator projection ...
50:20 | | CoerciveRBReductor: Assembling error estimator ...
50:20 | | | ResidualReductor: Estimating residual range ...
50:20 | | | | estimate_image_hierarchical: Estimating image for basis vector 4 ...
50:20 | | | | estimate_image_hierarchical: Orthonormalizing ...
50:20 | | | | gram_schmidt: Removing vector 16 of norm 2.620134530841409e-17
50:20 | | | | gram_schmidt: Orthonormalizing vector 17 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 18 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 19 again
50:20 | | | | gram_schmidt: Orthonormalizing vector 20 again
50:20 | | | ResidualReductor: Projecting residual operator ...
50:21 | | CoerciveRBReductor: Building ROM ...

```

```

50:21 weak_greedy: Estimating errors ...
50:21 weak_greedy: Maximum error after 5 extensions: 0.6855356256148512 (mu = {diffusion: [0.3471132191148264, 0.4611757018439767, 0.5761354541400012, 0.1008909945337475]})
50:21 weak_greedy: Extending surrogate for mu = {diffusion: [0.3471132191148264, 0.4611757018439767, 0.5761354541400012, 0.1008909945337475]} ...
50:21 | RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.3471132191148264, 0.4611757018439767, 0.5761354541400012, 0.1008909945337475]} ...
50:21 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG

```



```

for {diffusion: [0.3471132191148264, 0.4611757018439767, 0.5761354541400012,
0.1008909945337475], input: } ...
50:21 |   RBSurrogate: Extending basis with solution snapshot ...
50:21 |   |   gram_schmidt: Orthonormalizing vector 5 again
50:21 |   RBSurrogate: Reducing ...
50:21 |   |   CoerciveRBReductor: Operator projection ...
50:21 |   |   CoerciveRBReductor: Assembling error estimator ...
50:21 |   |   |   ResidualReductor: Estimating residual range ...
50:21 |   |   |   |   estimate_image_hierarchical: Estimating image for basis vector 5 ...
50:21 |   |   |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:21 |   |   |   |   |   |   gram_schmidt: Removing vector 20 of norm 1.3525392746001474e-17
50:21 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 21 again
50:21 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 22 again
50:21 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 23 again
50:21 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 24 again
50:21 |   |   |   |   |   |   |   ResidualReductor: Projecting residual operator ...
50:21 |   |   |   |   |   |   |   CoerciveRBReductor: Building ROM ...

50:21 weak_greedy: Estimating errors ...
50:21 weak_greedy: Maximum error after 6 extensions: 0.5490564169425676 (mu = {diffusion: [0.6311758444895136, 0.8644917997808012, 0.10426725362068653, 0.86803251340813]})
50:21 weak_greedy: Extending surrogate for mu = {diffusion: [0.6311758444895136, 0.8644917997808012, 0.10426725362068653, 0.86803251340813]} ...
50:21 |   RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.6311758444895136, 0.8644917997808012, 0.10426725362068653, 0.86803251340813]} ...
...
50:21 |   |   StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.6311758444895136, 0.8644917997808012, 0.10426725362068653, 0.86803251340813], input: } ...
50:21 |   RBSurrogate: Extending basis with solution snapshot ...
50:21 |   |   gram_schmidt: Orthonormalizing vector 6 again
50:21 |   RBSurrogate: Reducing ...
50:21 |   |   CoerciveRBReductor: Operator projection ...
50:21 |   |   CoerciveRBReductor: Assembling error estimator ...
50:21 |   |   |   ResidualReductor: Estimating residual range ...
50:21 |   |   |   |   estimate_image_hierarchical: Estimating image for basis vector 6 ...
50:21 |   |   |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:21 |   |   |   |   |   |   gram_schmidt: Removing vector 24 of norm 3.2159605655936935e-18
50:21 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 25 again
50:22 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 26 again
50:22 |   |   |   |   |   |   |   gram_schmidt: Orthonormalizing vector 27 again
50:22 |   |   |   |   |   |   |   gram_schmidt: Removing linearly dependent vector 28
50:22 |   |   |   |   |   |   |   ResidualReductor: Projecting residual operator ...
50:22 |   |   |   |   |   |   |   CoerciveRBReductor: Building ROM ...

50:22 weak_greedy: Estimating errors ...
50:22 weak_greedy: Maximum error after 7 extensions: 0.3644769343540863 (mu = {diffusion: [0.11823501538372526, 0.32459335834484837, 0.2066367271193214, 0.2951147311583003]})
50:22 weak_greedy: Extending surrogate for mu = {diffusion: [0.1182350153837

```

```

2526, 0.32459335834484837, 0.2066367271193214, 0.2951147311583003]] ...
50:22 |   RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.1
1823501538372526, 0.32459335834484837, 0.2066367271193214, 0.295114731158300
3]]} ...
50:22 |   |   StationaryModel: Computing solution of ThermalBlock((2, 2))_CG
for {diffusion: [0.11823501538372526, 0.32459335834484837, 0.206636727119321
4, 0.2951147311583003], input: } ...
50:22 |   RBSurrogate: Extending basis with solution snapshot ...
50:22 |   |   gram_schmidt: Orthonormalizing vector 7 again
50:22 |   RBSurrogate: Reducing ...
50:22 |   |   CoerciveRBReductor: Operator projection ...
50:22 |   |   CoerciveRBReductor: Assembling error estimator ...
50:22 |   |   |   ResidualReductor: Estimating residual range ...
50:22 |   |   |   estimate_image_hierarchical: Estimating image for basi
s vector 7 ...
50:22 |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:22 |   |   |   gram_schmidt: Removing vector 27 of norm 2.2565595
516559427e-17
50:22 |   |   |   |   gram_schmidt: Orthonormalizing vector 28 again
50:22 |   |   |   |   gram_schmidt: Orthonormalizing vector 29 again
50:22 |   |   |   |   gram_schmidt: Orthonormalizing vector 30 again
50:22 |   |   |   |   gram_schmidt: Orthonormalizing vector 31 again
50:22 |   |   |   ResidualReductor: Projecting residual operator ...
50:22 |   |   CoerciveRBReductor: Building ROM ...

50:22 weak_greedy: Estimating errors ...
50:23 weak_greedy: Maximum error after 8 extensions: 0.34868275615951083 (mu
= {diffusion: [0.9994510328630741, 0.10673469649608996, 0.24494111453794593,
0.706729870803821]})
50:23 weak_greedy: Extending surrogate for mu = {diffusion: [0.9994510328630
741, 0.10673469649608996, 0.24494111453794593, 0.706729870803821]} ...
50:23 |   RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.9
994510328630741, 0.10673469649608996, 0.24494111453794593, 0.70672987080382
1]]} ...
50:23 |   |   StationaryModel: Computing solution of ThermalBlock((2, 2))_CG
for {diffusion: [0.9994510328630741, 0.10673469649608996, 0.2449411145379459
3, 0.706729870803821], input: } ...
50:23 |   RBSurrogate: Extending basis with solution snapshot ...
50:23 |   |   gram_schmidt: Orthonormalizing vector 8 again
50:23 |   RBSurrogate: Reducing ...
50:23 |   |   CoerciveRBReductor: Operator projection ...
50:23 |   |   CoerciveRBReductor: Assembling error estimator ...
50:23 |   |   |   ResidualReductor: Estimating residual range ...
50:23 |   |   |   estimate_image_hierarchical: Estimating image for basi
s vector 8 ...
50:23 |   |   |   estimate_image_hierarchical: Orthonormalizing ...
50:23 |   |   |   gram_schmidt: Removing vector 31 of norm 4.0324855
74830063e-17
50:23 |   |   |   |   gram_schmidt: Orthonormalizing vector 32 again
50:23 |   |   |   |   gram_schmidt: Orthonormalizing vector 33 again
50:23 |   |   |   |   gram_schmidt: Orthonormalizing vector 34 again
50:23 |   |   |   |   gram_schmidt: Orthonormalizing vector 35 again
50:23 |   |   |   ResidualReductor: Projecting residual operator ...
50:23 |   |   CoerciveRBReductor: Building ROM ...

50:23 weak_greedy: Estimating errors ...

```

```

50:23 weak_greedy: Maximum error after 9 extensions: 0.1039373862771786 (mu
= {diffusion: [0.10149463690732048, 0.2008633338988278, 0.8764881073667903,
0.10110975626071733]})
50:23 weak_greedy: Extending surrogate for mu = {diffusion: [0.1014946369073
2048, 0.2008633338988278, 0.8764881073667903, 0.10110975626071733]} ...
50:23 | RBSSurrogate: Computing solution snapshot for mu = {diffusion: [0.1
0149463690732048, 0.2008633338988278, 0.8764881073667903, 0.1011097562607173
3]} ...
50:23 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG
for {diffusion: [0.10149463690732048, 0.2008633338988278, 0.876488107366790
3, 0.10110975626071733], input: } ...
50:23 | RBSSurrogate: Extending basis with solution snapshot ...
50:23 | | gram_schmidt: Orthonormalizing vector 9 again
50:23 | RBSSurrogate: Reducing ...
50:23 | | CoerciveRBReductor: Operator projection ...
50:23 | | CoerciveRBReductor: Assembling error estimator ...
50:23 | | | ResidualReductor: Estimating residual range ...
50:23 | | | estimate_image_hierarchical: Estimating image for basi
s vector 9 ...
50:23 | | | estimate_image_hierarchical: Orthonormalizing ...
50:23 | | | gram_schmidt: Removing vector 35 of norm 1.2822432
866770694e-16
50:23 | | | gram_schmidt: Orthonormalizing vector 36 again
50:23 | | | gram_schmidt: Orthonormalizing vector 37 again
50:23 | | | gram_schmidt: Orthonormalizing vector 38 again
50:23 | | | gram_schmidt: Orthonormalizing vector 39 again
50:23 | | | ResidualReductor: Projecting residual operator ...
50:23 | | CoerciveRBReductor: Building ROM ...

50:23 weak_greedy: Estimating errors ...
50:24 weak_greedy: Maximum error after 10 extensions: 0.09274463701023093 (m
u = {diffusion: [0.6751122843377482, 0.10207789532806781, 0.994031648570060
2, 0.35280674736773965]})
50:24 weak_greedy: Extending surrogate for mu = {diffusion: [0.6751122843377
482, 0.10207789532806781, 0.9940316485700602, 0.35280674736773965]} ...
50:24 | RBSSurrogate: Computing solution snapshot for mu = {diffusion: [0.6
751122843377482, 0.10207789532806781, 0.9940316485700602, 0.3528067473677396
5]} ...
50:24 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG
for {diffusion: [0.6751122843377482, 0.10207789532806781, 0.994031648570060
2, 0.35280674736773965], input: } ...
50:24 | RBSSurrogate: Extending basis with solution snapshot ...
50:24 | | gram_schmidt: Orthonormalizing vector 10 again
50:24 | RBSSurrogate: Reducing ...
50:24 | | CoerciveRBReductor: Operator projection ...
50:24 | | CoerciveRBReductor: Assembling error estimator ...
50:24 | | | ResidualReductor: Estimating residual range ...
50:24 | | | estimate_image_hierarchical: Estimating image for basi
s vector 10 ...
50:24 | | | estimate_image_hierarchical: Orthonormalizing ...
50:24 | | | gram_schmidt: Removing vector 39 of norm 2.1627522
429678418e-16
50:24 | | | gram_schmidt: Orthonormalizing vector 40 again
50:24 | | | gram_schmidt: Orthonormalizing vector 41 again
50:24 | | | gram_schmidt: Orthonormalizing vector 42 again
50:24 | | | gram_schmidt: Orthonormalizing vector 43 again

```

```

50:24 | | | ResidualReductor: Projecting residual operator ...
50:24 | | | CoerciveRBReductor: Building ROM ...

50:24 weak_greedy: Estimating errors ...
50:24 weak_greedy: Maximum error after 11 extensions: 0.028583454751637666
(mu = {diffusion: [0.1800358687452544, 0.12138085581146787, 0.15065993168379
208, 0.8682671558320606]})
50:24 weak_greedy: Extending surrogate for mu = {diffusion: [0.1800358687452
544, 0.12138085581146787, 0.15065993168379208, 0.8682671558320606]} ...
50:24 | RBSurrogate: Computing solution snapshot for mu = {diffusion: [0.1
800358687452544, 0.12138085581146787, 0.15065993168379208, 0.868267155832060
6]} ...
50:24 | | StationaryModel: Computing solution of ThermalBlock((2, 2))_CG
for {diffusion: [0.1800358687452544, 0.12138085581146787, 0.1506599316837920
8, 0.8682671558320606], input: } ...
50:24 | RBSurrogate: Extending basis with solution snapshot ...
50:24 | | gram_schmidt: Orthonormalizing vector 11 again
50:24 | RBSurrogate: Reducing ...
50:24 | | CoerciveRBReductor: Operator projection ...
50:24 | | CoerciveRBReductor: Assembling error estimator ...
50:24 | | | ResidualReductor: Estimating residual range ...
50:24 | | | estimate_image_hierarchical: Estimating image for basi
s vector 11 ...
50:24 | | | | estimate_image_hierarchical: Orthonormalizing ...
50:24 | | | | gram_schmidt: Removing vector 43 of norm 2.0369248
129428669e-16
50:24 | | | | gram_schmidt: Orthonormalizing vector 44 again
50:24 | | | | gram_schmidt: Orthonormalizing vector 45 again
50:24 | | | | gram_schmidt: Orthonormalizing vector 46 again
50:24 | | | | gram_schmidt: Orthonormalizing vector 47 again
50:24 | | | ResidualReductor: Projecting residual operator ...
50:24 | | CoerciveRBReductor: Building ROM ...

50:24 weak_greedy: Estimating errors ...
50:25 weak_greedy: Maximum error after 12 extensions: 0.00809749610183574 (m
u = {diffusion: [0.10051131633305001, 0.4887510949004338, 0.792318323168399
2, 0.7601197985198522]})
50:25 weak_greedy: Relative error tolerance (0.01) reached! Stopping extensi
on loop.
50:25 weak_greedy: Greedy search took 6.805584290999832 seconds

```

- Greedy search over training set of 1000 random parameters
- Fast thanks to efficient error estimator.
- FOM only solved for selected snapshot parameters.

The ROM

ROM is also a `StationaryModel`, but of lower order:

```
In [27]: rom_tb
```

```

Out[27]: StationaryModel(
  LincombOperator(
    (NumpyMatrixOperator(<12x12 dense>),
      NumpyMatrixOperator(<12x12 dense>),
      NumpyMatrixOperator(<12x12 dense>),
      NumpyMatrixOperator(<12x12 dense>),
      NumpyMatrixOperator(<12x12 dense>)),
    (1.0,
      ProjectionParameterFunctional('diffusion', size=4, index=0, name
='diffusion_0_0'),
      ProjectionParameterFunctional('diffusion', size=4, index=1, name
='diffusion_1_0'),
      ProjectionParameterFunctional('diffusion', size=4, index=2, name
='diffusion_0_1'),
      ProjectionParameterFunctional('diffusion', size=4, index=3, name
='diffusion_1_1'))),
    NumpyMatrixOperator(<12x1 dense>),
    output_functional=ZeroOperator(NumpyVectorSpace(0), NumpyVectorSpace(1
2)),
    products={h1: NumpyMatrixOperator(<12x12 dense>),
      h1_semi: NumpyMatrixOperator(<12x12 dense>),
      l2: NumpyMatrixOperator(<12x12 dense>),
      h1_0: NumpyMatrixOperator(<12x12 dense>),
      h1_0_semi: NumpyMatrixOperator(<12x12 dense>),
      l2_0: NumpyMatrixOperator(<12x12 dense>)},
    error_estimator=CoerciveRBEstimator(
      ResidualOperator(
        LincombOperator(
          (NumpyMatrixOperator(<47x12 dense>),
            NumpyMatrixOperator(<47x12 dense>),
            NumpyMatrixOperator(<47x12 dense>),
            NumpyMatrixOperator(<47x12 dense>),
            NumpyMatrixOperator(<47x12 dense>)),
          (1.0,
            ProjectionParameterFunctional('diffusion',
size=4, index=0, name='diffusion_0_0'),
            ProjectionParameterFunctional('diffusion',
size=4, index=1, name='diffusion_1_0'),
            ProjectionParameterFunctional('diffusion',
size=4, index=2, name='diffusion_0_1'),
            ProjectionParameterFunctional('diffusion',
size=4, index=3, name='diffusion_1_1'))),
          NumpyMatrixOperator(<47x1 dense>)),
          (1, 4, 8, 12, 16, 20, 24, 27, 31, 35, 39, 43, 47),
          ExpressionParameterFunctional('min(diffusion)', {di
ffusion: 4})),
        projected_output_adjoint=ZeroOperator(NumpyVectorSp
ace(0), NumpyVectorSpace(0))),
    output_d_mu_use_adjoint=True,
    name='ThermalBlock((2, 2))_CG_reduced')

```

Comparing FOM and ROM solutions

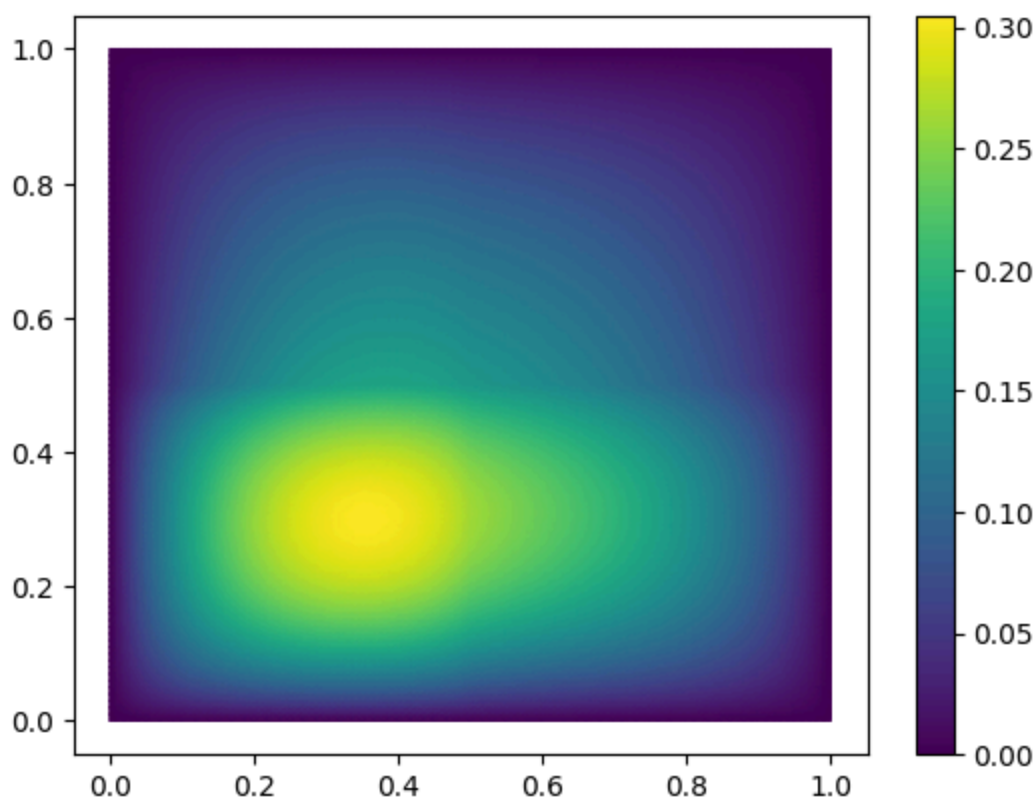
ROM solution:

```
In [28]: U_rom = rom_tb.solve(mu)
         U_rom
```

```
Out[28]: NumpyVectorArray(
  NumpyVectorSpace(12),
  [[ 0.67677343  0.16392157 -0.11003419  0.1264038  -0.0090707  -0.081917
    52          0.00760874  0.06166952 -0.03416274 -0.00680792 -0.00094802 -0.002067
    79]],
  _len=1)
```

Reconstruct and compare to FOM solution:

```
In [29]: U_rec = reductor.reconstruct(U_rom)
         fom_tb.visualize(U - U_rec)
```



Is it faster?

Finally, we check that the reduced-order model is indeed faster:

```
In [30]: from time import perf_counter
         tic = perf_counter()
         fom_tb.solve(mu)
         toc = perf_counter()
         rom_tb.solve(mu)
         tac = perf_counter()
         print(f't_fom: {toc-tic}  t_rom: {tac-toc}  speedup: {(toc-tic)/(tac-toc)}')
```

```
51:55 StationaryModel: Computing solution of ThermalBlock((2, 2))_CG for {diffusion: [0.1, 0.2, 0.5, 1.0], input: } ...
t_fom: 0.08190578299945628 t_rom: 0.0005040910000388976 speedup: 162.48213714019118
```

Your turn

- Validate the FOM by computing the maximum error between ROM and FOM solution on a validation set of 10 random parameter values.

```
In [ ]: validation_set = parameter_space.sample_randomly(...)
errors = []
for mu in ...:
    u_rom = ...
    U_rom = ...
    U_fom = ...
    U_err = ...
    err = U_err.norm()[0]
    errors.append(err)

print(max(errors))
```

Example: Balanced Truncation for LTI System

Details in a lecture tomorrow.

Here we consider a synthetic linear time-invariant (LTI) system of the form

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t),\end{aligned}$$

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input, and $y(t) \in \mathbb{R}^p$ is the output.

```
In [9]: from pymor.models.examples import penzl_example

fom_lti = penzl_example()
```

The result is an `LTIModel` :

```
In [10]: fom_lti
```

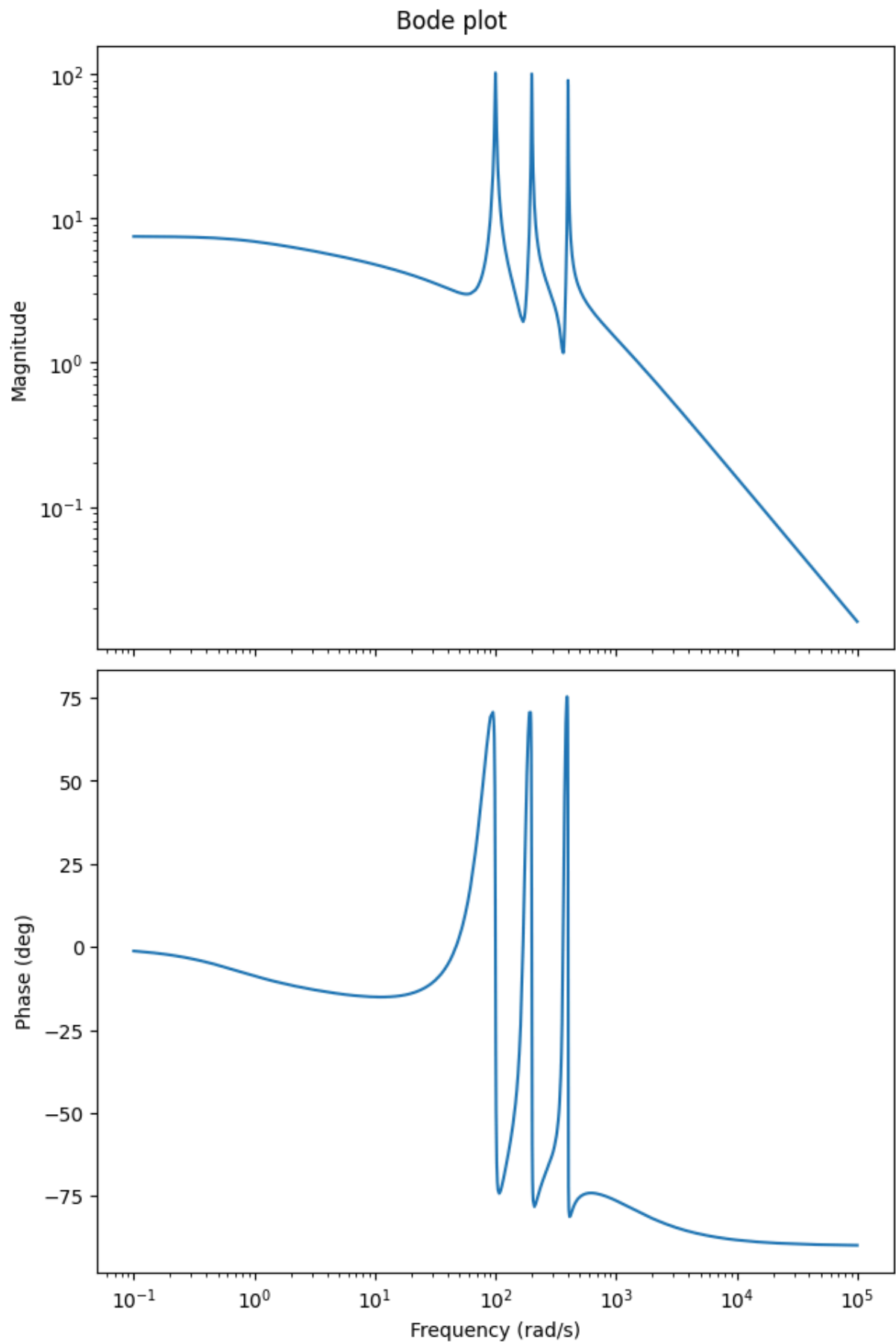
```
Out[10]: LTIModel(
  NumpyMatrixOperator(<1006x1006 sparse, 1012 nnz>, source_id='STATE', range_id='STATE'),
  NumpyMatrixOperator(<1006x1 dense>, range_id='STATE'),
  NumpyMatrixOperator(<1x1006 dense>, source_id='STATE'),
  D=ZeroOperator(NumpyVectorSpace(1), NumpyVectorSpace(1)),
  E=IdentityOperator(NumpyVectorSpace(1006, id='STATE')),
  presets={})
```

```
In [11]: print(fom_lti)
```

```
LTIModel
  class: LTIModel
  number of equations: 1006
  number of inputs:    1
  number of outputs:   1
  continuous-time
  linear time-invariant
  solution_space: NumpyVectorSpace(1006, id='STATE')
```

We can use the Bode plot to show the frequency response of the LTI system, i.e., to see which input frequencies are amplified and phase-shifted in the output.

```
In [12]: w = (1e-1, 1e5)
_ = fom_lti.transfer_function.bode_plot(w)
```

We can run balanced truncation to obtain a reduced-order model.

```
In [13]: from pymor.reductors.bt import BTReductor
```

```
bt = BTReductor(fom_lti)  
rom_lti = bt.reduce(10)
```

```
13:13 solve_lyap_lrcf: Relative residual at step 1: 4.70519e-01
13:13 solve_lyap_lrcf: Relative residual at step 2: 4.11276e-01
13:13 solve_lyap_lrcf: Relative residual at step 4: 3.80485e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 solve_lyap_lrcf: Relative residual at step 6: 3.60769e-01
13:13 solve_lyap_lrcf: Relative residual at step 8: 3.47048e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 10: 3.38956e-01
13:13 solve_lyap_lrcf: Relative residual at step 12: 2.64325e-01
13:13 solve_lyap_lrcf: Relative residual at step 14: 2.55645e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 16: 2.53360e-01
13:13 solve_lyap_lrcf: Relative residual at step 18: 2.35558e-01
13:13 solve_lyap_lrcf: Relative residual at step 20: 1.36598e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 22: 1.36522e-01
13:13 solve_lyap_lrcf: Relative residual at step 24: 1.36044e-01
13:13 solve_lyap_lrcf: Relative residual at step 26: 1.30162e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 28: 1.30069e-01
13:13 solve_lyap_lrcf: Relative residual at step 30: 1.29750e-01
13:13 solve_lyap_lrcf: Relative residual at step 32: 1.29687e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 33: 1.27134e-01
13:13 solve_lyap_lrcf: Relative residual at step 35: 1.27075e-01
13:13 solve_lyap_lrcf: Relative residual at step 36: 1.23738e-01
13:13 solve_lyap_lrcf: Relative residual at step 38: 2.87264e-02
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 39: 2.83539e-02
13:13 solve_lyap_lrcf: Relative residual at step 41: 2.81845e-02
13:13 solve_lyap_lrcf: Relative residual at step 43: 1.26187e-03
13:13 solve_lyap_lrcf: Relative residual at step 44: 1.08075e-04
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 45: 3.78805e-05
13:13 solve_lyap_lrcf: Relative residual at step 46: 1.93073e-05
```

13:13 solve_lyap_lrcf: Relative residual at step 47: 1.39965e-05
13:13 solve_lyap_lrcf: Relative residual at step 49: 1.39009e-05
13:13 solve_lyap_lrcf: Relative residual at step 50: 8.18039e-07
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 51: 1.97127e-07
13:13 solve_lyap_lrcf: Relative residual at step 52: 7.04409e-08
13:13 solve_lyap_lrcf: Relative residual at step 53: 4.08047e-08
13:13 solve_lyap_lrcf: Relative residual at step 55: 3.21046e-08
13:13 solve_lyap_lrcf: Relative residual at step 56: 4.49953e-09
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 57: 1.91517e-09
13:13 solve_lyap_lrcf: Relative residual at step 58: 1.44064e-09
13:13 solve_lyap_lrcf: Relative residual at step 59: 1.12671e-09
13:13 solve_lyap_lrcf: Relative residual at step 61: 8.17619e-10
13:13 solve_lyap_lrcf: Relative residual at step 62: 6.89158e-10
13:13 gram_schmidt: Orthonormalizing vector 1 again
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 63: 6.71517e-10
13:13 solve_lyap_lrcf: Relative residual at step 64: 6.69988e-10
13:13 solve_lyap_lrcf: Relative residual at step 65: 6.61104e-10
13:13 solve_lyap_lrcf: Relative residual at step 67: 6.44165e-10
13:13 solve_lyap_lrcf: Relative residual at step 68: 6.32330e-10
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 69: 6.32028e-10
13:13 solve_lyap_lrcf: Relative residual at step 70: 6.29770e-10
13:13 solve_lyap_lrcf: Relative residual at step 72: 3.88769e-10
13:13 solve_lyap_lrcf: Relative residual at step 74: 4.48577e-11
13:13 gram_schmidt: Orthonormalizing vector 1 again
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 1: 4.70519e-01
13:13 solve_lyap_lrcf: Relative residual at step 2: 4.11276e-01
13:13 solve_lyap_lrcf: Relative residual at step 4: 3.80485e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 solve_lyap_lrcf: Relative residual at step 6: 3.60769e-01
13:13 solve_lyap_lrcf: Relative residual at step 8: 3.47048e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 10: 3.38956e-01
13:13 solve_lyap_lrcf: Relative residual at step 12: 2.64325e-01
13:13 solve_lyap_lrcf: Relative residual at step 14: 2.55645e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again

13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 16: 2.53360e-01
13:13 solve_lyap_lrcf: Relative residual at step 18: 2.35558e-01
13:13 solve_lyap_lrcf: Relative residual at step 20: 1.36598e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 22: 1.36522e-01
13:13 solve_lyap_lrcf: Relative residual at step 24: 1.36044e-01
13:13 solve_lyap_lrcf: Relative residual at step 26: 1.30162e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 28: 1.30069e-01
13:13 solve_lyap_lrcf: Relative residual at step 30: 1.29750e-01
13:13 solve_lyap_lrcf: Relative residual at step 32: 1.29687e-01
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 33: 1.27134e-01
13:13 solve_lyap_lrcf: Relative residual at step 35: 1.27075e-01
13:13 solve_lyap_lrcf: Relative residual at step 36: 1.23738e-01
13:13 solve_lyap_lrcf: Relative residual at step 38: 2.87264e-02
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 39: 2.83539e-02
13:13 solve_lyap_lrcf: Relative residual at step 41: 2.81845e-02
13:13 solve_lyap_lrcf: Relative residual at step 43: 1.26187e-03
13:13 solve_lyap_lrcf: Relative residual at step 44: 1.08075e-04
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 45: 3.78805e-05
13:13 solve_lyap_lrcf: Relative residual at step 46: 1.93073e-05
13:13 solve_lyap_lrcf: Relative residual at step 47: 1.39965e-05
13:13 solve_lyap_lrcf: Relative residual at step 49: 1.39009e-05
13:13 solve_lyap_lrcf: Relative residual at step 50: 8.18039e-07
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 51: 1.97127e-07
13:13 solve_lyap_lrcf: Relative residual at step 52: 7.04409e-08
13:13 solve_lyap_lrcf: Relative residual at step 53: 4.08047e-08
13:13 solve_lyap_lrcf: Relative residual at step 55: 3.21046e-08
13:13 solve_lyap_lrcf: Relative residual at step 56: 4.49953e-09
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 57: 1.91517e-09
13:13 solve_lyap_lrcf: Relative residual at step 58: 1.44064e-09
13:13 solve_lyap_lrcf: Relative residual at step 59: 1.12671e-09
13:13 solve_lyap_lrcf: Relative residual at step 61: 8.17619e-10

```

13:13 solve_lyap_lrcf: Relative residual at step 62: 6.89158e-10
13:13 gram_schmidt: Orthonormalizing vector 1 again
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 63: 6.71517e-10
13:13 solve_lyap_lrcf: Relative residual at step 64: 6.69988e-10
13:13 solve_lyap_lrcf: Relative residual at step 65: 6.61104e-10
13:13 solve_lyap_lrcf: Relative residual at step 67: 6.44165e-10
13:13 solve_lyap_lrcf: Relative residual at step 68: 6.32330e-10
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 solve_lyap_lrcf: Relative residual at step 69: 6.32028e-10
13:13 solve_lyap_lrcf: Relative residual at step 70: 6.29770e-10
13:13 solve_lyap_lrcf: Relative residual at step 72: 3.88769e-10
13:13 solve_lyap_lrcf: Relative residual at step 74: 4.48577e-11
13:13 gram_schmidt: Orthonormalizing vector 1 again
13:13 gram_schmidt: Orthonormalizing vector 2 again
13:13 gram_schmidt: Orthonormalizing vector 3 again
13:13 gram_schmidt: Orthonormalizing vector 4 again
13:13 gram_schmidt: Orthonormalizing vector 5 again
13:13 LTIPGReductor: Operator projection ...
13:13 LTIPGReductor: Building ROM ...

```

The reduced-order model is again an `LTIModel`, but of lower order.

```
In [14]: rom_lti
```

```

Out[14]: LTIModel(
  NumpyMatrixOperator(<10x10 dense>),
  NumpyMatrixOperator(<10x1 dense>),
  NumpyMatrixOperator(<1x10 dense>),
  D=ZeroOperator(NumpyVectorSpace(1), NumpyVectorSpace(1)),
  E=NumpyMatrixOperator(<10x10 dense>),
  presets={},
  name='LTIModel_reduced')

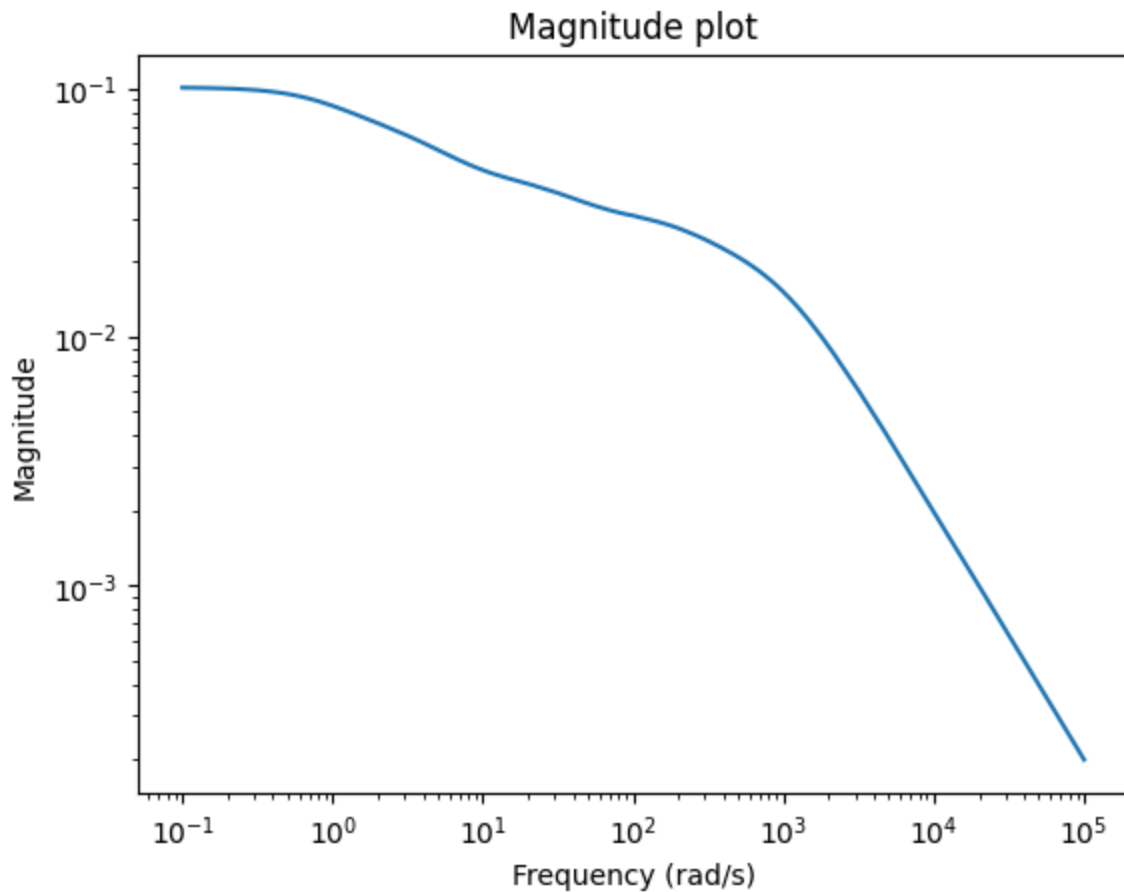
```

Looking at the error system, we can see which frequencies are well approximated.

```

In [15]: err_lti = fom_lti - rom_lti
_ = err_lti.transfer_function.mag_plot(w)

```



Your turn

1. Change the reduced order above from 10 to 5, then regenerate the error plot.
2. Change the reduced order to 20 and regenerate the plot.

Running Demo Scripts

pyMOR ships several example scripts that showcase various features of the library. While many features are also covered in our tutorials, the demos are more extensive and often have various command-line flags which allow to run the script for different parameters or problems. All demos can be found in the [src/pymordemos](#) directory of the source repository.

The demo scripts can be launched directly from the source tree:

```
./thermalblock.py --plot-err --plot-solutions 3 2 3 32
```

or by using the `pymor-demo` script that is installed with pyMOR:

```
pymor-demo thermalblock --plot-err --plot-solutions 3 2 3 32
```

Getting help

- pyMOR's documentation can be found at

<https://docs.pymor.org/latest>

- Be sure to read the [introduction](#), the [technical overview](#) and the [tutorials](#).
- Ask questions on

<https://github.com/pymor/pymor/discussions>