

---

# **mud-examples Documentation**

***Release 0.2.1.post1.dev1+g89f84a2***

**Mathematical Michael**

**Jan 22, 2024**



# CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	readme . . . . .	3
1.2	License . . . . .	4
1.3	Contributors . . . . .	5
1.4	mud_examples . . . . .	5
1.5	Changelog . . . . .	13
<b>2</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



This is the documentation of **mud\_examples**.



## CONTENTS

### 1.1 readme

#### 1.1.1 MUD-Examples

*Examples for Existence, Uniqueness, and Convergence of Parameter Estimates with Maximal Updated Densities*

Authors: Troy Butler & Michael Pilosov

#### 1.1.2 Installation

For Python 3.7-3.12:

```
pip install mud-examples
```

To reproduce the results in Michael's thesis, use `mud-examples==0.1`. However, this comes with `mud==0.0.28`. Newer versions should still produce the same figures.

TeX is recommended (but not required):

```
apt-get install -yqq \  
  texlive-base \  
  texlive-latex-base \  
  texlive-latex-extra \  

```

(continues on next page)

(continued from previous page)

```
texlive-fonts-recommended \  
texlive-fonts-extra \  
texlive-science \  
latexmk \  
dvipng \  
cm-super
```

### 1.1.3 Quickstart

Generate all of the figures the way they are referenced in the paper:

```
mud_run_all
```

The above is equivalent to running all of the examples sequentially:

```
mud_run_inv  
mud_run_lin  
mud_run_ode  
mud_run_pde
```

### 1.1.4 Usage

The `mud_run_X` scripts all call the same primary entrypoint, which you can call with the console script `mud_examples`.

Here are two examples:

```
mud_examples --example ode
```

```
mud_examples --example lin
```

and so on. (More on this later, once argparsing is better handled, they might just be entrypoints to the modules themselves rather than a central `runner.py`, which really only exists to compare several experiments, so perhaps it warrants renaming to reflect that).

## 1.2 License

The MIT License (MIT)

Copyright (c) 2021 Mathematical Michael

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT



HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 1.3 Contributors

- Mathematical Michael <[consistentbayes@gmail.com](mailto:consistentbayes@gmail.com)>

## 1.4 mud\_examples

### 1.4.1 mud\_examples package

#### Subpackages

#### mud\_examples.linear package

#### Submodules

#### mud\_examples.linear.lin module

`mud_examples.linear.lin.compare_linear_sols(transform, lam_ref, A, b, alpha=1, mean=None, cov=None)`

Input dimension fixed, varying according to the output of the anonymous function *transform*'s return.

`mud_examples.linear.lin.compare_linear_sols_dim(lam_ref, A, b, alpha=1, mean=None, cov=None)`

Input dimension fixed, varying output dimension.

`mud_examples.linear.lin.compare_linear_sols_rank_list(lam_ref, A, b, alpha=1, mean=None, cov=None)`

Input and output dimensions fixed, varying rank 1..dim\_output.

`mud_examples.linear.lin.compare_mud_map_pin(A, b, y, mean, cov)`

`mud_examples.linear.lin.contour_example(A=array([[1, 1]]), b=array([[0.]]), cov_11=0.5, cov_01=-0.25, initial_mean=array([0.25, 0.25]), alpha=1, omega=1, obs_std=1, show_full=True, show_data=True, show_est=False, param_ref=None, compare=False, fsize=42, filename='latest_figure.png', save=False)`

alpha: float in [0, 1], weight of Tikhonov regularization omega: float in [0, 1], weight of Modified regularization

`mud_examples.linear.lin.main(args)`

Main entrypoint for example-generation

`mud_examples.linear.lin.main_contours(args)`

Main entrypoint for 2D Linear Rank-Deficient Example (Contour Plots)

`mud_examples.linear.lin.main_dim(args)`

Main entrypoint for High-Dim Linear Dimension Example

`mud_examples.linear.lin.main_meas(args)`

Main entrypoint for High-Dim Linear Measurement Example

`mud_examples.linear.lin.main_meas_var(args)`

Main entrypoint for High-Dim Linear Measurement Example

`mud_examples.linear.lin.main_rank(args)`

Main entrypoint for High-Dim Linear Rank Example

`mud_examples.linear.lin.run()`

Entry point for console\_scripts

`mud_examples.linear.lin.run_meas()`

Entry point for console\_scripts

`mud_examples.linear.lin.run_meas_var()`

Entry point for console\_scripts

`mud_examples.linear.lin.setup_logging(loglevel)`

Setup basic logging

#### Parameters

**loglevel** (*int*) – minimum loglevel for emitting messages

`mud_examples.linear.lin.transform_dim_out(lam_ref, A, b, dim)`

`mud_examples.linear.lin.transform_measurements(operator_list, data_list, measurements, std_list, noise)`

`mud_examples.linear.lin.transform_rank_list(lam_ref, A, b, rank)`

A is a list here. We sum the first *rank* elements of it to return a matrix with the desired rank.

## mud\_examples.linear.models module

`mud_examples.linear.models.createNoisyReferenceData(M, reference_point, std, num_data=None)`

`mud_examples.linear.models.createRandomLinearMap(dim_input, dim_output, dist='normal',  
repeated=False)`

Create random linear map from P dimensions to S dimensions.

`mud_examples.linear.models.createRandomLinearPair(reference_point, num_data, std, dist='normal',  
repeated=False)`

data will come from a normal distribution centered at zero with standard deviation given by *std* QoI map will come from standard uniform, or normal if *dist='normal'* if *repeated* is True, the map will be rank 1.

`mud_examples.linear.models.createRandomLinearProblem(reference_point, num_qoi, num_observations,  
std_list, dist='normal', repeated=False)`

Wrapper around *createRandomLinearQoI* to generalize to multiple QoI maps.

`mud_examples.linear.models.randA_gauss(dim_output, dim_input=None, seed=None)`

Generate random Gaussian matrix, perform QR, and returns the resulting (orthogonal) Q

`mud_examples.linear.models.randA_list_svd(dim_output, dim_input=None, seed=None) → List`

Generate random square Gaussian matrix, perform SVD, and construct rank-1 matrices from components. Return list of them. Sum *R* entries of this returned list to generate a rank-R matrix.

```
mud_examples.linear.models.randA_outer(dim_output, dim_input=None, seed=None)
```

Generate *dimension* rank-1 matrices using Gaussian entries to generate a vector  $x$  and then take outer-product with self.

```
mud_examples.linear.models.randA_qr(dim_output, dim_input=None, seed=None)
```

Generate random Gaussian matrix, perform QR, and returns the resulting (orthogonal) Q

```
mud_examples.linear.models.randP(dim_output, dim_input=None, randA=<function randA_gauss>,
                                   seed=None)
```

Constructs problem set

## Module contents

### Submodules

#### mud\_examples.experiments module

```
mud_examples.experiments.experiment_equipment(fun, num_measure, sd_vals, num_trials, seed=21)
```

Fixed number of sensors, varying the quality of equipment.

```
mud_examples.experiments.experiment_measurements(fun, num_measurements, sd, num_trials, seed=21)
```

Fixed sensors, varying how much data is incorporated into the solution.

```
mud_examples.experiments.plot_experiment_equipment(tolerances, res, prefix, fsize=32, linewidth=5,
                                                    title='Variance of MUD Error', save=True)
```

```
mud_examples.experiments.plot_experiment_measurements(res, prefix, fsize=32, linewidth=5,
                                                       xlabel='Number of Measurements',
                                                       save=True, legend=True)
```

#### mud\_examples.models module

```
mud_examples.models.generate_decay_model(t, lam_true)
```

```
mud_examples.models.generate_rotation_map(qnum=10, orth=True)
```

```
mud_examples.models.generate_spatial_measurements(num_measure, xmin=0.05, xmax=0.95, ymin=0.05,
                                                    ymax=0.95)
```

```
mud_examples.models.generate_temporal_measurements(measurement_hertz=100, start_time=1,
                                                    end_time=3)
```

#### mud\_examples.monomial module

```
mud_examples.monomial.QoI(lam, p)
```

Defines a QoI mapping function as monomials to some power  $p$

```
mud_examples.monomial.data_likelihood(qvals, data, num_data, sigma)
```

```
mud_examples.monomial.main(args)
```

Main entrypoint for example-generation

```
mud_examples.monomial.run()
```

```
mud_examples.monomial.setup_logging(loglevel)
```

Setup basic logging

**Parameters**

**loglevel** (*int*) – minimum loglevel for emitting messages

## mud\_examples.ode module

```
mud_examples.ode.main_ode(num_trials=20, fsize=32, seed=21, lam_true=0.5, domain=[[0, 1]],
                           tolerances=[0.1], time_ratios=[0.01, 1], alt=False, bayes=True)
```

```
>>> from mud_examples.ode import main_ode
>>> res = main_ode(num_trials=5, time_ratios=[0.01, 0.1, 1])
Will run simulations for %T=[0.01, 0.1, 1]
Running example: mud
Measurements: [2, 20, 200]
Plotting decay solution.
Running example: map
Measurements: [2, 20, 200]
Plotting decay solution.
```

## mud\_examples.parsers module

```
mud_examples.parsers.parse_args(args)
```

Parse command line parameters

**Parameters**

**args** (*[str]*) – command line parameters as list of strings

**Returns**

command line parameters namespace

**Return type**

*argparse.Namespace*

## mud\_examples.pde module

```
mud_examples.pde.main_pde(num_trials=20, tolerances=[0.1], measurements=[20, 100, 500], fsize=32,
                           seed=21, lam_true=-3.0, input_dim=2, dist='u', sample_dist='u',
                           num_samples=None, sample_tol=0.95, alt=True, bayes=True, **kwargs)
```

**\*\*kwargs** are used for the setting of the initial distribution. >>> res = main\_pde(num\_trials=3) Attempt run for measurements = [20, 100, 500] Running example: mud Running example: mud-alt Running example: map

```
>>> res = main_pde(num_trials=3, dist='n')
Attempt run for measurements = [20, 100, 500]
Running example: mud
Running example: mud-alt
Running example: map
```

```
>>> res = main_pde(num_trials=3, dist='n', sample_dist='n', sample_tol=0.99)
Attempt run for measurements = [20, 100, 500]
Running example: mud
Running example: mud-alt
Running example: map
```

## mud\_examples.plotting module

```
mud_examples.plotting.plotChain(mud_chain, ref_param, color='k', s=100)

mud_examples.plotting.plotContours(A, ref_param, subset=None, color='k', ls=':', lw=1, fs=20, w=1,
                                   s=100, **kws)

mud_examples.plotting.plot_decay_solution(solutions, model_generator, sigma, prefix, time_vector,
                                           lam_true, qoi_true, end_time=3, fsize=32, save=True)

mud_examples.plotting.plot_scalar_poisson_summary(res, measurements, prefix, lam_true, fsize=32,
                                                  save=False)
```

## mud\_examples.poisson module

```
mud_examples.poisson.band_qoi(sensors, num_qoi=1, axis=1)

mud_examples.poisson.copy_expression(expression)

mud_examples.poisson.dist_from_fname(fname)
    Function that infers distribution used to generate samples from the filename It looks for a letter before .pkl, i.e.
    ..n.pkl -> normal distribution.

mud_examples.poisson.eval_boundary(u, n)

mud_examples.poisson.eval_boundary_piecewise(u, n, d=1)
    Takes an Expression u (on unit domain) and returns the string for another expression based on evaluating a
    piecewise-linear approximation. The mesh is equispaced into n intervals.

mud_examples.poisson.evaluate_and_save_poisson(sample, save_prefix)
    sample is a tuple (index, gamma)

mud_examples.poisson.expressionNorm(u, v, n=100)

mud_examples.poisson.gamma_boundary_condition(gamma=-3)
    Defines boundary condition parameterized by either a scalar or list/iterable. In the latter case, piecewise-
    interpolation on an equispaced grid over the interior of (0, 1). In the former, the scalar defines the minimum
    displacement value of the boundary condition.

mud_examples.poisson.get_boundary_markers_for_rect(mesh, width=1)

mud_examples.poisson.load_poisson_from_disk(fname)

mud_examples.poisson.load_poisson_from_fenics_run(sensors, file_list, nx=36, ny=36)
```

`mud_examples.poisson.main(args)`

Main entry point allowing external calls. Generates PDE data (requires fenics to be installed)

**Parameters**

**args** (*[str]*) – command line parameter list

`mud_examples.poisson.make_map_wrapper(domain, lam, qoi, qoi_true, log=False, dist=<scipy.stats._continuous_distns.norm_gen object>, **kwargs)`

Anonymous function

`mud_examples.poisson.make_mud_wrapper(domain, lam, qoi, qoi_true, indices=None, sample_dist='u', dist=<scipy.stats._continuous_distns.norm_gen object>, **kwargs)`

Anonymous function

`mud_examples.poisson.make_reproducible_without_fenics(example='mud', lam_true=-3, input_dim=2, sample_dist='u', sample_tol=0.95, num_samples=None, num_measure=100)`

(Currently) requires XML data to be on disk, simulates sensors and saves everything required to one pickle file.

`mud_examples.poisson.parse_args(args)`

Parse command line parameters

**Parameters**

**args** (*[str]*) – command line parameters as list of strings

**Returns**

command line parameters namespace

**Return type**

*argparse.Namespace*

`class mud_examples.poisson.pdeProblem(fname=None)`

Bases: *object*

**property dist**

**property domain**

**property g**

**property lam**

**property lam\_ref**

**load(fname=None)**

Loads from filename, e.g., “data/pde\_2D/ref\_1000\_2u.pkl”

**map\_scalar(log=True, \*\*kwargs)**

**mud\_scalar(\*\*kwargs)**

**mud\_vector\_horizontal(num\_qoi=None, \*\*kwargs)**

**mud\_vector\_vertical(num\_qoi=None, \*\*kwargs)**

**plot(sols=None, num\_measurements=20, example='mud', fsize=36, ftype='png', save=False)**

**plot\_initial**(*save=True, \*\*kwargs*)

**plot\_solutions**(*sols, num, save=True, \*\*kwargs*)

**property qoi**

**property qoi\_ref**

**property sample\_dist**

**property sensors**

**property u**

**mud\_examples.poisson.piecewise\_eval**(*xvals, yvals, d=1*)

**mud\_examples.poisson.piecewise\_eval\_from\_vector**(*u, d=1*)

Takes an iterable *u* with y-values (on interior of equispaced unit domain) and returns the string for an expression based on evaluating a piecewise-linear approximation through these points.

**mud\_examples.poisson.plot\_without\_fenics**(*fname, num\_sensors=None, num\_qoi=2, mode='sca', fsize=36, example=None*)

**mud\_examples.poisson.poissonModel**(*gamma=-3, mesh=None, width=1, nx=36, ny=36*)

*gamma* is scaling parameter for left boundary condition *n\_x* and *n\_y* are the number of elements for the horizontal/vertical axes of the mesh

**mud\_examples.poisson.poisson\_sensor\_model**(*sensors, gamma, nx, ny, mesh=None*)

Convenience function wrapper to just return a qoi given a parameter.

**mud\_examples.poisson.run()**

Entry point for console\_scripts

**mud\_examples.poisson.setup\_logging**(*loglevel*)

Setup basic logging

#### Parameters

**loglevel** (*int*) – minimum loglevel for emitting messages

### **mud\_examples.runner module**

**mud\_examples.runner.main**(*in\_args*)

Main entrypoint for example-generation

**mud\_examples.runner.run()**

Entry point for console\_scripts

**mud\_examples.runner.run\_all()**

Recreates all figures in MUD paper.

**mud\_examples.runner.run\_linear()**

Recreates Contour figures in MUD paper. >>> run\_linear() Running Linear Examples. >>> import os; os.system('rm -rf figures/') 0

**mud\_examples.runner.run\_monomial()**

Recreates Contour figures in MUD paper. >>> run\_monomial() Running BIP vs SIP Comparison (1D). >>> import os; os.system('rm -rf figures/') 0

`mud_examples.runner.run_ode()`

Recreates Poisson figures in MUD paper.

```
>>> run_ode()
Will run simulations for %T=[0.125, 0.25, 0.5, 1.0]
Running example: mud
Measurements: [25, 50, 100, 200]
Plotting decay solution.
Running example: map
Measurements: [25, 50, 100, 200]
Plotting decay solution.
Plotting experiments involving increasing # of measurements.
>>> import os; os.system('rm -rf figures/')
0
```

`mud_examples.runner.run_pde()`

Recreates Poisson figures in MUD paper.

```
>>> run_pde()
Attempt run for measurements = [25, 50, 100, 200, 400]
Running example: mud
Running example: map
Plotting experiments involving increasing # of measurements.
>>> import os; os.system('rm -rf figures/')
0
```

`mud_examples.runner.setup_logging(loglevel)`

Setup basic logging

**Parameters**

**loglevel** (*int*) – minimum loglevel for emitting messages

## mud\_examples.summary module

`mud_examples.summary.extract_statistics(solutions, reference_value)`

Extracts experiment statistics from solutions set Assumes keys of dictionary are sample sizes, and each value is a list containing solutions for each trial.

```
>>> S = {2: [1, 1, 1], 4: [1, 1, 1]}
>>> means, vars = extract_statistics(S, 0)
>>> print(means)
[1.0, 1.0]
>>> print(vars)
[0.0, 0.0]
```

`mud_examples.summary.maybe_fit_log_linear_regression(input_values, output_values)`

Fits a log-linear regression

```
>>> import numpy as np
>>> x = np.arange(1,11)
>>> np.round(maybe_fit_log_linear_regression(x,x)[1], 4)
1.0
```



## mud\_examples.utils module

**class** mud\_examples.utils.**LazyLoader**(*module\_name='utensor\_cgen', submod\_name=None*)

Bases: module

mud\_examples.utils.**check\_dir**(*directory*)

mud\_examples.utils.**make\_2d\_normal\_mesh**(*N=50, window=1*)

Constructs mesh based on normal distribution to discretize each axis. >>> from mud\_examples.utils import make\_2d\_normal\_mesh >>> x, y, XX = make\_2d\_normal\_mesh(3) >>> print(XX) [[-1. -1.]

[ 0. -1.] [ 1. -1.] [-1. 0.] [ 0. 0.] [ 1. 0.] [-1. 1.] [ 0. 1.] [ 1. 1.]]

mud\_examples.utils.**make\_2d\_unit\_mesh**(*N=50, window=1*)

Constructs mesh based on uniform distribution to discretize each axis. >>> from mud\_examples.utils import make\_2d\_unit\_mesh >>> x, y, XX = make\_2d\_unit\_mesh(3) >>> print(XX) [[0. 0. ]

[0.5 0. ] [1. 0. ] [0. 0.5] [0.5 0.5] [1. 0.5] [0. 1. ] [0.5 1. ] [1. 1. ]]

## Module contents

## 1.5 Changelog



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### m

- `mud_examples`, 13
- `mud_examples.experiments`, 7
- `mud_examples.linear`, 7
- `mud_examples.linear.lin`, 5
- `mud_examples.linear.models`, 6
- `mud_examples.models`, 7
- `mud_examples.monomial`, 7
- `mud_examples.ode`, 8
- `mud_examples.parsers`, 8
- `mud_examples.pde`, 8
- `mud_examples.plotting`, 9
- `mud_examples.poisson`, 9
- `mud_examples.runner`, 11
- `mud_examples.summary`, 12
- `mud_examples.utils`, 13



## INDEX

### B

`band_qoi()` (in module *mud\_examples.poisson*), 9

### C

`check_dir()` (in module *mud\_examples.utils*), 13

`compare_linear_sols()` (in module *mud\_examples.linear.lin*), 5

`compare_linear_sols_dim()` (in module *mud\_examples.linear.lin*), 5

`compare_linear_sols_rank_list()` (in module *mud\_examples.linear.lin*), 5

`compare_mud_map_pin()` (in module *mud\_examples.linear.lin*), 5

`contour_example()` (in module *mud\_examples.linear.lin*), 5

`copy_expression()` (in module *mud\_examples.poisson*), 9

`createNoisyReferenceData()` (in module *mud\_examples.linear.models*), 6

`createRandomLinearMap()` (in module *mud\_examples.linear.models*), 6

`createRandomLinearPair()` (in module *mud\_examples.linear.models*), 6

`createRandomLinearProblem()` (in module *mud\_examples.linear.models*), 6

### D

`data_likelihood()` (in module *mud\_examples.monomial*), 7

`dist` (*mud\_examples.poisson.pdeProblem* property), 10

`dist_from_fname()` (in module *mud\_examples.poisson*), 9

`domain` (*mud\_examples.poisson.pdeProblem* property), 10

### E

`eval_boundary()` (in module *mud\_examples.poisson*), 9

`eval_boundary_piecewise()` (in module *mud\_examples.poisson*), 9

`evaluate_and_save_poisson()` (in module *mud\_examples.poisson*), 9

`experiment_equipment()` (in module *mud\_examples.experiments*), 7

`experiment_measurements()` (in module *mud\_examples.experiments*), 7

`expressionNorm()` (in module *mud\_examples.poisson*), 9

`extract_statistics()` (in module *mud\_examples.summary*), 12

### G

`g` (*mud\_examples.poisson.pdeProblem* property), 10

`gamma_boundary_condition()` (in module *mud\_examples.poisson*), 9

`generate_decay_model()` (in module *mud\_examples.models*), 7

`generate_rotation_map()` (in module *mud\_examples.models*), 7

`generate_spatial_measurements()` (in module *mud\_examples.models*), 7

`generate_temporal_measurements()` (in module *mud\_examples.models*), 7

`get_boundary_markers_for_rect()` (in module *mud\_examples.poisson*), 9

### L

`lam` (*mud\_examples.poisson.pdeProblem* property), 10

`lam_ref` (*mud\_examples.poisson.pdeProblem* property), 10

`LazyLoader` (class in *mud\_examples.utils*), 13

`load()` (*mud\_examples.poisson.pdeProblem* method), 10

`load_poisson_from_disk()` (in module *mud\_examples.poisson*), 9

`load_poisson_from_fenics_run()` (in module *mud\_examples.poisson*), 9

### M

`main()` (in module *mud\_examples.linear.lin*), 5

`main()` (in module *mud\_examples.monomial*), 7

`main()` (in module *mud\_examples.poisson*), 9

`main()` (in module *mud\_examples.runner*), 11

`main_contours()` (in module *mud\_examples.linear.lin*), 5

`main_dim()` (in module `mud_examples.linear.lin`), 5  
`main_meas()` (in module `mud_examples.linear.lin`), 5  
`main_meas_var()` (in module `mud_examples.linear.lin`), 6  
`main_ode()` (in module `mud_examples.ode`), 8  
`main_pde()` (in module `mud_examples.pde`), 8  
`main_rank()` (in module `mud_examples.linear.lin`), 6  
`make_2d_normal_mesh()` (in module `mud_examples.utils`), 13  
`make_2d_unit_mesh()` (in module `mud_examples.utils`), 13  
`make_map_wrapper()` (in module `mud_examples.poisson`), 10  
`make_mud_wrapper()` (in module `mud_examples.poisson`), 10  
`make_reproducible_without_fenics()` (in module `mud_examples.poisson`), 10  
`map_scalar()` (`mud_examples.poisson.pdeProblem` method), 10  
`maybe_fit_log_linear_regression()` (in module `mud_examples.summary`), 12  
`module`  
    `mud_examples`, 13  
    `mud_examples.experiments`, 7  
    `mud_examples.linear`, 7  
    `mud_examples.linear.lin`, 5  
    `mud_examples.linear.models`, 6  
    `mud_examples.models`, 7  
    `mud_examples.monomial`, 7  
    `mud_examples.ode`, 8  
    `mud_examples.parsers`, 8  
    `mud_examples.pde`, 8  
    `mud_examples.plotting`, 9  
    `mud_examples.poisson`, 9  
    `mud_examples.runner`, 11  
    `mud_examples.summary`, 12  
    `mud_examples.utils`, 13  
`mud_examples`  
    module, 13  
`mud_examples.experiments`  
    module, 7  
`mud_examples.linear`  
    module, 7  
`mud_examples.linear.lin`  
    module, 5  
`mud_examples.linear.models`  
    module, 6  
`mud_examples.models`  
    module, 7  
`mud_examples.monomial`  
    module, 7  
`mud_examples.ode`  
    module, 8  
`mud_examples.parsers`  
    module, 8  
`mud_examples.pde`  
    module, 8  
`mud_examples.plotting`  
    module, 9  
`mud_examples.poisson`  
    module, 9  
`mud_examples.runner`  
    module, 11  
`mud_examples.summary`  
    module, 12  
`mud_examples.utils`  
    module, 13  
`mud_scalar()` (`mud_examples.poisson.pdeProblem` method), 10  
`mud_vector_horizontal()`  
    (`mud_examples.poisson.pdeProblem` method), 10  
`mud_vector_vertical()`  
    (`mud_examples.poisson.pdeProblem` method), 10

## P

`parse_args()` (in module `mud_examples.parsers`), 8  
`parse_args()` (in module `mud_examples.poisson`), 10  
`pdeProblem` (class in `mud_examples.poisson`), 10  
`piecewise_eval()` (in module `mud_examples.poisson`), 11  
`piecewise_eval_from_vector()` (in module `mud_examples.poisson`), 11  
`plot()` (`mud_examples.poisson.pdeProblem` method), 10  
`plot_contours()` (in module `mud_examples.plotting`), 9  
`plot_decay_solution()` (in module `mud_examples.plotting`), 9  
`plot_experiment_equipment()` (in module `mud_examples.experiments`), 7  
`plot_experiment_measurements()` (in module `mud_examples.experiments`), 7  
`plot_initial()` (`mud_examples.poisson.pdeProblem` method), 10  
`plot_scalar_poisson_summary()` (in module `mud_examples.plotting`), 9  
`plot_solutions()` (`mud_examples.poisson.pdeProblem` method), 11  
`plot_without_fenics()` (in module `mud_examples.poisson`), 11  
`plotChain()` (in module `mud_examples.plotting`), 9  
`poisson_sensor_model()` (in module `mud_examples.poisson`), 11  
`poissonModel()` (in module `mud_examples.poisson`), 11

## Q

`qoi` (`mud_examples.poisson.pdeProblem` property), 11  
`QoI()` (in module `mud_examples.monomial`), 7



`qoi_ref` (*mud\_examples.poisson.pdeProblem* property),  
11

## R

`randA_gauss()` (in module *mud\_examples.linear.models*), 6  
`randA_list_svd()` (in module *mud\_examples.linear.models*), 6  
`randA_outer()` (in module *mud\_examples.linear.models*), 6  
`randA_qr()` (in module *mud\_examples.linear.models*), 7  
`randP()` (in module *mud\_examples.linear.models*), 7  
`run()` (in module *mud\_examples.linear.lin*), 6  
`run()` (in module *mud\_examples.monomial*), 7  
`run()` (in module *mud\_examples.poisson*), 11  
`run()` (in module *mud\_examples.runner*), 11  
`run_all()` (in module *mud\_examples.runner*), 11  
`run_linear()` (in module *mud\_examples.runner*), 11  
`run_meas()` (in module *mud\_examples.linear.lin*), 6  
`run_meas_var()` (in module *mud\_examples.linear.lin*),  
6  
`run_monomial()` (in module *mud\_examples.runner*), 11  
`run_ode()` (in module *mud\_examples.runner*), 11  
`run_pde()` (in module *mud\_examples.runner*), 12

## S

`sample_dist` (*mud\_examples.poisson.pdeProblem*  
property), 11  
`sensors` (*mud\_examples.poisson.pdeProblem* property),  
11  
`setup_logging()` (in module *mud\_examples.linear.lin*),  
6  
`setup_logging()` (in module *mud\_examples.monomial*), 8  
`setup_logging()` (in module *mud\_examples.poisson*),  
11  
`setup_logging()` (in module *mud\_examples.runner*),  
12

## T

`transform_dim_out()` (in module *mud\_examples.linear.lin*), 6  
`transform_measurements()` (in module *mud\_examples.linear.lin*), 6  
`transform_rank_list()` (in module *mud\_examples.linear.lin*), 6

## U

`u` (*mud\_examples.poisson.pdeProblem* property), 11