

---

# **Model Analysis ToolKit (MATK) Documentation**

*Release 0*

**Dylan R. Harp**

May 16, 2015



## CONTENTS

<b>1</b>	<b>Obtaining MATK</b>	<b>1</b>
1.1	Downloading MATK . . . . .	1
1.2	Installing MATK . . . . .	1
1.3	Testing installation . . . . .	1
<b>2</b>	<b>Examples</b>	<b>3</b>
2.1	Latin Hypercube Sampling . . . . .	3
2.2	Calibration . . . . .	4
<b>3</b>	<b>Class Documentation</b>	<b>7</b>
3.1	MATK . . . . .	7
3.2	Parameter . . . . .	12
3.3	Observation . . . . .	12
3.4	SampleSet . . . . .	13
<b>4</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>
	<b>Index</b>	<b>21</b>



## OBTAINING MATK

### 1.1 Downloading MATK

MATK can be obtained by:

1. Using git:

```
git clone https://github.com/dharp/matk
```

2. Clicking [here](#)
3. Going to <https://github.com/dharp/matk> and clicking on the **Download ZIP** button on the right

### 1.2 Installing MATK

To install MATK, enter the main directory in a terminal and

```
python setup.py install
```

Depending on your system setup and privileges, you may want to do this as root (\*nix and mac systems):

```
sudo python setup.py install
```

or as user:

```
python setup.py install --user
```

If all these fail, you can set your PYTHONPATH to point to the MATK *src* directory

1. bash:

```
export PYTHONPATH=/path/to/matk/src
```

2. tcsh:

```
setenv PYTHONPATH /path/to/matk/src
```

3. Windows, I have no clue!

### 1.3 Testing installation

To test that the MATK module is accessible, open a python/ipython terminal and

```
import matk
```

If the MATK module is accessible, this will load without an error.

For more in depth analysis of MATK functionality on your system, the test suite can be run by entering the MATK *tests* directory in a terminal and:

```
python -W ignore matk_unittests.py -v
```

Test results will be printed to the terminal.

**EXAMPLES**

## 2.1 Latin Hypercube Sampling

This example demonstrates a Latin Hypercube Sampling of a 4 parameter 4 response model using the `lhs` function. The generation of diagnostic plots is demonstrated using `hist`, `panels`, and `corr`.

The script `sampling.py` is available in the `examples` folder of the repository or can be downloaded [here](#).

```
1 import sys,os
2 try:
3     import matk
4 except:
5     try:
6         sys.path.append(os.path.join('.', 'src', 'matk'))
7         import matk
8     except ImportError as err:
9         print 'Unable to load MATK module: '+str(err)
10 import numpy
11 from scipy import arange, randn, exp
12 try:
13     from collections import OrderedDict as dict
14 except:
15     print "Warning: collections module is not installed"
16     print "Ordering of observations will not be maintained in output"
17 from multiprocessing import freeze_support
18
19 # Model function
20 def dbexpl(p):
21     t=arange(0,100,20.)
22     y = (p['par1']*exp(-p['par2']*t) + p['par3']*exp(-p['par4']*t))
23     #nm = ['o1', 'o2', 'o3', 'o4', 'o5']
24     #return dict(zip(nm,y))
25     return y
26
27 def run():
28     # Setup MATK model with parameters
29     p = matk.matk(model=dbexpl)
30     p.add_par('par1',min=0,max=1)
31     p.add_par('par2',min=0,max=0.2)
32     p.add_par('par3',min=0,max=1)
33     p.add_par('par4',min=0,max=0.2)
34
35     # Create LHS sample
36     s = p.lhs('lhs', siz=500, seed=1000)
37
```

```

38     # Look at sample parameter histograms, correlations, and panels
39     s.samples.hist(ncols=2,title='Parameter Histograms')
40     s.samples.hist(ncols=2,title='Parameter Histograms',frequency=True)
41     parcor = s.samples.corr(plot=True, title='Parameter Correlations')
42     s.samples.panels(title='Parameter Panels')
43
44     # Run model with parameter samples
45     s.run( cpus=2, outfile='results.dat', logfile='log.dat',verbose=False)
46
47     # Look at response histograms, correlations, and panels
48     s.responses.hist(ncols=2,title='Model Response Histograms')
49     s.responses.hist(ncols=2,title='Model Response Histograms',frequency=True)
50     rescor = s.responses.corr(plot=True, title='Model Response Correlations')
51     s.responses.panels(title='Response Panels')
52
53     # Print and plot parameter/response correlations
54     print "\nPearson Correlation Coefficients:"
55     pcorr = s.corr(plot=True,title='Pearson Correlation Coefficients')
56     print "\nSpearman Correlation Coefficients:"
57     scorr = s.corr(plot=True,type='spearman',title='Spearman Rank Correlation Coefficients')
58     s.panels(figsize=(10,8))
59
60 # Freeze support is necessary for multiprocessing on windows
61 if __name__ == "__main__":
62     freeze_support()
63     run()

```

## 2.2 Calibration

```

1  import numpy as np
2  try:
3      import matk
4  except:
5      try:
6          sys.path.append(os.path.join('.', 'src', 'matk'))
7          import matk
8      except ImportError as err:
9          print 'Unable to load MATK module: '+str(err)
10 from multiprocessing import freeze_support
11
12 def fv(a):
13     a0 = a['a0']
14     a1 = a['a1']
15     a2 = a['a2']
16     X = np.array([1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.,12.])
17
18     out = a0 / (1. + a1 * np.exp( X * a2))
19     return out
20     #obsnames = ['obs'+str(i) for i in range(1,len(out)+1)]
21     #return dict(zip(obsnames,out))
22
23 def run():
24     p = matk.matk(model=fv)
25     p.add_par('a0', value=0.7, min=-2000., max=2000.)
26     #p.add_par('a0', value=0.7)
27     #p.add_par('a1', value=10., min=-2000., max=2000.)

```



```
28 p.add_par('a1', value=10.)
29 #p.add_par('a2', value=-0.4, min=-20000., max=20000.)
30 p.add_par('a2', value=-0.4)
31 #p.forward()
32 p.obsvalues = [5.308,7.24,9.638,12.866,17.069,23.192,31.443,38.558,50.156,62.948,75.995,91.972]
33
34 p.calibrate(cpus=6,verbose=True)
35
36 # Freeze support is necessary for multiprocessing on windows
37 if __name__ == "__main__":
38     freeze_support()
39     run()
```



## CLASS DOCUMENTATION

### 3.1 MATK

**class** `matk.matk` (*model=''*, *model\_args=None*, *model\_kwargs=None*, *cpus=1*, *workdir\_base=None*,  
*workdir=None*, *results\_file=None*, *seed=None*, *sample\_size=10*, *hosts={}*)  
Class for Model Analysis ToolKit (MATK) module

**Jac** (*h=0.001*, *cpus=1*, *workdir\_base=None*, *save=True*, *reuse\_dirs=False*)  
Numerical Jacobian calculation

**Parameters** *h* (*fl64* or *ndarray(fl64)*) – Parameter increment, single value or array with *npar* values

**Returns** *ndarray(fl64)* – Jacobian matrix

**MCMC** (*nruns=10000*, *burn=1000*, *init\_error\_std=1.0*, *max\_error\_std=100.0*, *verbose=1*)  
Perform Markov Chain Monte Carlo sampling using pymc package

**Parameters**

- **nruns** (*int*) – Number of MCMC iterations (samples)
- **burn** (*int*) – Number of initial samples to burn (discard)
- **verbose** (*int*) – verbosity of output
- **init\_error\_std** (*fl64*) – Initial standard deviation of residuals
- **max\_error\_std** (*fl64*) – Maximum standard deviation of residuals that will be considered

**Returns** *pymc* MCMC object

**add\_obs** (*name*, *sim=None*, *weight=1.0*, *value=None*)  
Add observation to problem

**Parameters**

- **name** (*str*) – Observation name
- **sim** (*fl64*) – Simulated value
- **weight** (*fl64*) – Observation weight
- **value** (*fl64*) – Value of observation

**Returns** Observation object

**add\_par** (*name*, *\*\*kwargs*)  
Add parameter to problem

### Parameters

- **name** (*str*) – Name of parameter
- **kwargs** – keyword arguments passed to parameter class

**calibrate** (*cpus=1, maxiter=100, lambdax=0.001, minchange=1e-16, minlambdax=1e-06, verbose=False, workdir=None, reuse\_dirs=False, h=1e-06*)

Calibrate MATK model using Levenberg-Marquardt algorithm based on original code written by Ernesto P. Adorio PhD. (UPDEPP at Clarkfield, Pampanga)

### Parameters

- **cpus** (*int*) – Number of cpus to use
- **maxiter** (*int*) – Maximum number of iterations
- **lambdax** (*fl64*) – Initial Marquardt lambda
- **minchange** (*fl64*) – Minimum change between successive ChiSquares
- **minlambdax** (*fl4*) – Minimum lambda value
- **verbose** (*bool*) – If True, additional information written to screen during calibration

**Returns** best fit parameters found by routine

**Returns** best Sum of squares.

**Returns** covariance matrix

**copy\_sampleset** (*oldname, newname=None*)

Copy sampleset

### Parameters

- **oldname** (*str*) – Name of sampleset to copy
- **newname** (*str*) – Name of new sampleset

**cpus**

Set number of cpus to use for concurrent model evaluations

**create\_sampleset** (*samples, name=None, responses=None, indices=None, index\_start=1*)

Add sample set to problem

### Parameters

- **name** (*str*) – Name of sample set
- **samples** (*list(fl64), ndarray(fl64)*) – Matrix of parameter samples with npar columns in order of `matk.pars.keys()`
- **responses** (*list(fl64), ndarray(fl64)*) – Matrix of associated responses with nobs columns in order `matk.obs.keys()` if observation exists (existence of observations is not required)
- **indices** (*list(int), ndarray(int)*) – Sample indices to use when creating working directories and output files

**emcee** (*lnprob=None, nwalkers=100, nsamples=500, burnin=50, pos0=None*)

Perform Markov Chain Monte Carlo sampling using emcee package

### Parameters

- **lnprob** (*function*) – Function specifying the natural logarithm of the likelihood function
- **nwalkers** (*int*) – Number of random walkers

- **nsamples** (*int*) – Number of samples per walker
- **burnin** (*int*) – Number of “burn-in” samples per walker to be discarded
- **pos0** (*list*) – list of initial positions for the walkers

**Returns** numpy array containing samples

**forward** (*pardict=None, workdir=None, reuse\_dirs=False, job\_number=None, hostname=None, processor=None*)

Run MATK model using current values

#### Parameters

- **pardict** (*dict*) – Dictionary of parameter values keyed by parameter names
- **workdir** (*str*) – Name of directory where model will be run. It will be created if it does not exist
- **reuse\_dirs** (*bool*) – If True and workdir exists, the model will reuse the directory
- **job\_number** (*int*) – Sample id
- **hostname** (*str*) – Name of host to run job on, will be passed to MATK model as kwarg ‘hostname’
- **processor** (*str or int*) – Processor id to run job on, will be passed to MATK model as kwarg ‘processor’

**Returns** int – 0: Successful run, 1: workdir exists

**levmar** (*workdir=None, reuse\_dirs=False, max\_iter=1000, full\_output=True*)

Calibrate MATK model using levmar package

#### Parameters

- **workdir** (*str*) – Name of directory where model will be run. It will be created if it does not exist
- **reuse\_dirs** (*bool*) – If True and workdir exists, the model will reuse the directory
- **max\_iter** (*int*) – Maximum number of iterations
- **full\_output** – If True, additional output displayed during calibration

**Returns** levmar output

**lhs** (*name=None, siz=None, noCorrRestr=False, corrmatrix=None, seed=None, index\_start=1*)

Draw lhs samples of parameter values from scipy.stats module distribution

#### Parameters

- **name** (*str*) – Name of sample set to be created
- **siz** (*int*) – Number of samples to generate, ignored if samples are provided
- **noCorrRestr** (*bool*) – If True, correlation structure is not enforced on sample, use if siz is less than number of parameters
- **corrmatrix** (*matrix*) – Correlation matrix
- **seed** (*int*) – Random seed to allow replication of samples
- **index\_start** – Starting value for sample indices

**Type** int

**Returns** matrix – Parameter samples

**lmfit** (*maxfev=0, report\_fit=True, cpus=1, epsfcn=None, xtol=1e-07, ftol=1e-07, workdir=None, verbose=False, \*\*kwargs*)  
Calibrate MATK model using lmfit package

#### Parameters

- **maxfev** (*int*) – Max number of function evaluations, if 0, 100\*(npars+1) will be used
- **report\_fit** (*bool*) – If True, parameter statistics and correlations are printed to the screen
- **cpus** (*int*) – Number of cpus to use for concurrent simulations during jacobian approximation
- **epsfcn** (*float*) – jacobian finite difference approximation increment
- **xtol** (*float*) – Relative error in approximate solution
- **ftol** (*float*) – Relative error in the desired sum of squares
- **workdir** (*str*) – Name of directory to use for model runs, calibrated parameters will be run there after calibration
- **verbose** (*bool*) – If true, print diagnostic information to the screen

**Returns** lmfit minimizer object

Additional keyword arguments will be passed to scipy leastsq function: <http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.optimize.leastsq.html>

**make\_workdir** (*workdir=None, reuse\_dirs=False*)  
Create a working directory

#### Parameters

- **workdir** (*str*) – Name of directory where model will be run. It will be created if it does not exist
- **reuse\_dirs** (*bool*) – If True and workdir exists, the model will reuse the directory

**Returns** int – 0: Successful run, 1: workdir exists

**model**  
Python function that runs model

**model\_args**  
Tuple of extra arguments to MATK model expected to come after parameter dictionary

**model\_kwargs**  
Dictionary of extra keyword arguments to MATK model expected to come after parameter dictionary and model\_args

**obsnames**  
Get observation names

**obsvalues**  
Observation values

**obsweights**  
Get observation names

**pardist\_pars**  
Get parameters needed by parameter distributions

**pardists**  
Get parameter probabilistic distributions

**parmaxs**

Get parameter lower bounds

**parmins**

Get parameter lower bounds

**parnames**

Get parameter names

**parstudy** (*name=None, nvals=2*)

Generate parameter study samples

**Parameters**

- **name** (*str*) – Name of sample set to be created
- **outfile** (*str*) – Name of file where samples will be written. If outfile=None, no file is written.
- **nvals** (*int or list(int)*) – number of values for each parameter

**Returns** ndarray(float64) – Array of samples

**parvalues**

Parameter values

**read\_sampleset** (*file, name=None*)

Read MATK output file and assemble corresponding sampleset with responses.

**Parameters**

- **name** (*str*) – Name of sample set
- **file** (*str*) – Path to MATK output file

**residuals**

Get least squares values

**results\_file**

Set the name of the results\_file for parallel runs

**seed**

Set the seed for random sampling

**simvalues**

Simulated values :returns: list(float64) – simulated values in order of matk.obs.keys()

**ssr**

Sum of squared residuals

**workdir**

Set the base name for parallel working directories

**workdir\_base**

Set the base name for parallel working directories

**workdir\_index**

Set the working directory index for parallel runs

## 3.2 Parameter

```
class matk.parameter.Parameter (name, value=None, vary=True, min=None, max=None, expr=None,  
                                discrete_vals=[], discrete_counts=[], **kwargs)
```

MATK parameter class

**dist**

Probabilistic distribution of parameter belonging to scipy.stats module

**dist\_pars**

Distribution parameters required by self.dist e.g. if dist == uniform, dist\_pars = (min,max-min)

if dist == norm, dist\_pars = (mean,stdev))

**expr**

Mathematical expression to use to evaluate value

**setup\_bounds** ()

set up Minuit-style internal/external parameter transformation of min/max bounds.

returns internal value for parameter from self.value (which holds the external, user-expected value). This internal values should actually be used in a fit...

As a side-effect, this also defines the self.from\_internal method used to re-calculate self.value from the internal value, applying the inverse Minuit-style transformation. This method should be called prior to passing a Parameter to the user-defined objective function.

This code borrows heavily from JJ Helmus' leastsqbound.py

**value**

Parameter value

**vary**

Boolean indicating whether or not to vary parameter

## 3.3 Observation

```
class matk.observation.Observation (name, sim=None, weight=1.0, value=None)
```

MATK observation class

**name**

Observation name

**residual**

Observation value minus simulated value

**sim**

Simulated value generated by MATK model

**value**

Observation value

**weight**

Weight to apply to simulated values



## 3.4 SampleSet

**class** `matk.sampleset.SampleSet` (*name, samples, parent, index\_start=1, \*\*kwargs*)

MATK SampleSet class - Stores information related to a sample including parameter samples, associated responses, and sample indices

**calc\_sse** ()

Calculate sum of squared errors (sse) for all samples

**Returns** `lst(fl64)`

**corr** (*type='pearson', plot=False, printout=True, plotvals=True, figsize=None, title=None*)

Calculate correlation coefficients of parameters and responses

**Parameters**

- **type** (*str*) – Type of correlation coefficient (pearson by default, spearman also available)
- **plot** (*bool*) – If True, plot correlation matrix
- **printout** (*bool*) – If True, print correlation matrix with row and column headings
- **plotvals** (*bool*) – If True, print correlation coefficients on plot matrix
- **figsize** (*tuple(fl64,fl64)*) – Width and height of figure in inches
- **title** (*str*) – Title of plot

**Returns** `ndarray(fl64)` – Correlation coefficients

**index\_start**

Starting integer value for sample indices

**indices**

Array of sample indices

**main\_effects** ()

For each parameter, compile array of main effects.

**name**

Sample set name

**obsnames**

Array of observation names

**panels** (*type='pearson', alpha=0.2, figsize=None, title=None, tight=False, symbol='.', fontsize=None, corrfontsize=None, ms=5, mins=None, maxs=None, frequency=False, bins=10, ylim=None, labels=[], filename=None, xticks=2, yticks=2*)

Plot histograms, scatterplots, and correlation coefficients in paired matrix

**Parameters**

- **type** (*str*) – Type of correlation coefficient (pearson by default, spearman also available)
- **alpha** (*float*) – Histogram color shading
- **figsize** (*tuple(fl64,fl64)*) – Width and height of figure in inches
- **title** (*str*) – Title of plot
- **tight** (*bool*) – Use matplotlib tight layout
- **symbol** (*str*) – matplotlib symbol for scatterplots
- **fontsize** (*fl64*) – Size of font for axis labels

- **corrfontsize** (*fl64*) – Size of font for correlation coefficients
- **ms** (*fl64*) – Scatterplot marker size
- **frequency** (*bool*) – If True, the first element of the return tuple will be the counts normalized by the length of data, i.e.,  $n/\text{len}(x)$
- **bins** (*int*) – Number of bins in histograms
- **ylim** (*tuples - 2 element tuples with y limits for histograms*) – y-axis limits for histograms.
- **labels** (*lst(str)*) – Names to use instead of parameter names in plot
- **filename** (*int*) – Name of file to save plot. File ending determines plot type (pdf, png, ps, eps, etc.). Plot types available depends on the matplotlib backend in use on the system. Plot will not be displayed.
- **xticks** – Number of ticks along x axes
- **yticks** – Number of ticks along y axes

**pardict** (*index*)

Get parameter dictionary for sample with specified index

**Parameters** **index** (*int*) – Sample index

**Returns** dict(fl64)

**parnames**

Array of observation names

**rank\_parameter\_frequencies** ()

Yields a printout of parameter value frequencies in the sample set

**returns** An array of tuples, each containing the parameter name tagged as min or max and a second tuple containing the parameter value and the frequency of its appearance in the sample set.

**recarray**

Structured (record) array of samples

**run** (*cpus=1, workdir\_base=None, save=True, reuse\_dirs=False, outfile=None, logfile=None, verbose=True, hosts={}*)

Run model using values in samples for parameter values If samples are not specified, LHS samples are produced

**Parameters**

- **cpus** (*int,dict(lst)*) – number of cpus; alternatively, dictionary of lists of processor ids keyed by hostnames to run models on (i.e. on a cluster); hostname provided as kwarg to model (hostname=<hostname>); processor id provided as kwarg to model (processor=<processor id>)
- **workdir\_base** (*str*) – Base name for model run folders, run index is appended to workdir\_base
- **save** (*bool*) – If True, model files and folders will not be deleted during parallel model execution
- **reuse\_dirs** (*bool*) – Will use existing directories if True, will return an error if False and directory exists
- **outfile** (*str*) – File to write results to
- **logfile** (*str*) – File to write details of run to during execution
- **hosts** (*lst(str)*) – Option deprecated, use cpus instead

**Returns** tuple(ndarray(fl64),ndarray(fl64)) - (Matrix of responses from sampled model runs size rows by npar columns, Parameter samples, same as input samples if provided)

**saveetxt** (*outfile*)

Save sampleset to file

**Parameters** **outfile** (*str*) – Name of file where sampleset will be written

**subset** (*boolfcn, obs, \*args, \*\*kwargs*)

Collect samples based on response values, remove all others

**Parameters**

- **boolfcn** – Function that returns true for samples to keep and false for samples to remove
- **obs** (*str*) – Name of response to apply boolfcn to
- **args** – Additional arguments to add to boolfcn
- **kwargs** – Keyword arguments to add to boolfcn

`matk.sampleset.hist` (*rc, ncols=4, figsize=None, alpha=0.2, title=None, tight=False, mins=None, maxs=None, frequency=False, bins=10, ylim=None, printout=True, labels=[], filename=None, fontsize=None, xticks=3*)

Plot histograms of dataset

**Parameters**

- **ncols** (*int*) – Number of columns in plot matrix
- **figsize** (*tuple(fl64,fl64)*) – Width and height of figure in inches
- **alpha** (*float*) – Histogram color shading
- **title** (*str*) – Title of plot
- **tight** (*bool*) – Use matplotlib tight layout
- **mins** (*lst(fl64)*) – Minimum values of recarray fields
- **maxs** (*lst(fl64)*) – Maximum values of recarray fields
- **frequency** (*bool*) – If True, the first element of the return tuple will be the counts normalized by the length of data, i.e.,  $n/\text{len}(x)$
- **bins** (*int or lst(lst(int))*) – If an integer is given, bins + 1 bin edges are returned. Unequally spaced bins are supported if bins is a list of sequences for each histogram.
- **ylim** (*tuple - 2 element tuple with y limits for histograms*) – y-axis limits for histograms.
- **labels** (*lst(str)*) – Names to use instead of parameter names in plot
- **filename** (*str*) – Name of file to save plot. File ending determines plot type (pdf, png, ps, eps, etc.). Plot types available depends on the matplotlib backend in use on the system. Plot will not be displayed.
- **fontsize** (*fl64*) – Size of font
- **xticks** (*int*) – Number of ticks on xaxes

**Returns** dict(lst(int),lst(fl64)) - dictionary of histogram data (counts,bins) keyed by name

`matk.sampleset.corr` (*rc1, rc2, type='pearson', plot=False, printout=True, plotvals=True, figsize=None, title=None*)

Calculate correlation coefficients of parameters and responses

**Parameters**

- **rc1** – Data
- **rc2** – Data
- **type** (*str*) – Type of correlation coefficient (pearson by default, spearman also available)
- **plot** (*bool*) – If True, plot correlation matrix
- **printout** (*bool*) – If True, print correlation matrix with row and column headings
- **plotvals** (*bool*) – If True, print correlation coefficients on plot matrix
- **figsize** (*tuple(fl64,fl64)*) – Width and height of figure in inches
- **title** (*str*) – Title of plot

**Returns** ndarray(fl64) – Correlation coefficients

## INDICES AND TABLES

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



## m

`matk.observation`, [12](#)  
`matk.parameter`, [12](#)  
`matk.sampleset`, [2](#)





**A**

add\_obs() (matk.matk method), 7  
 add\_par() (matk.matk method), 7

**C**

calc\_sse() (matk.sampleset.SampleSet method), 13  
 calibrate() (matk.matk method), 8  
 copy\_sampleset() (matk.matk method), 8  
 corr() (in module matk.sampleset), 15  
 corr() (matk.sampleset.SampleSet method), 13  
 cpus (matk.matk attribute), 8  
 create\_sampleset() (matk.matk method), 8

**D**

dist (matk.parameter.Parameter attribute), 12  
 dist\_pars (matk.parameter.Parameter attribute), 12

**E**

emcee() (matk.matk method), 8  
 expr (matk.parameter.Parameter attribute), 12

**F**

forward() (matk.matk method), 9

**H**

hist() (in module matk.sampleset), 15

**I**

index\_start (matk.sampleset.SampleSet attribute), 13  
 indices (matk.sampleset.SampleSet attribute), 13

**J**

Jac() (matk.matk method), 7

**L**

levmar() (matk.matk method), 9  
 lhs() (matk.matk method), 9  
 lmfit() (matk.matk method), 9

**M**

main\_effects() (matk.sampleset.SampleSet method), 13

make\_workdir() (matk.matk method), 10  
 matk (class in matk), 7  
 matk (module), 7  
 matk.observation (module), 12  
 matk.parameter (module), 12  
 matk.sampleset (module), 2, 5, 13  
 MCMC() (matk.matk method), 7  
 model (matk.matk attribute), 10  
 model\_args (matk.matk attribute), 10  
 model\_kwargs (matk.matk attribute), 10

**N**

name (matk.observation.Observation attribute), 12  
 name (matk.sampleset.SampleSet attribute), 13

**O**

Observation (class in matk.observation), 12  
 obsnames (matk.matk attribute), 10  
 obsnames (matk.sampleset.SampleSet attribute), 13  
 obsvalues (matk.matk attribute), 10  
 obsweights (matk.matk attribute), 10

**P**

panels() (matk.sampleset.SampleSet method), 13  
 Parameter (class in matk.parameter), 12  
 pardict() (matk.sampleset.SampleSet method), 14  
 pardist\_pars (matk.matk attribute), 10  
 pardists (matk.matk attribute), 10  
 parmaxs (matk.matk attribute), 10  
 parmins (matk.matk attribute), 11  
 parnames (matk.matk attribute), 11  
 parnames (matk.sampleset.SampleSet attribute), 14  
 parstudy() (matk.matk method), 11  
 parvalues (matk.matk attribute), 11

**R**

rank\_parameter\_frequencies()  
     (matk.sampleset.SampleSet method), 14  
 read\_sampleset() (matk.matk method), 11  
 recarray (matk.sampleset.SampleSet attribute), 14  
 residual (matk.observation.Observation attribute), 12  
 residuals (matk.matk attribute), 11

results\_file (matk.matk attribute), [11](#)

run() (matk.sampleset.SampleSet method), [14](#)

## S

SampleSet (class in matk.sampleset), [13](#)

savetxt() (matk.sampleset.SampleSet method), [15](#)

seed (matk.matk attribute), [11](#)

setup\_bounds() (matk.parameter.Parameter method), [12](#)

sim (matk.observation.Observation attribute), [12](#)

simvalues (matk.matk attribute), [11](#)

ssr (matk.matk attribute), [11](#)

subset() (matk.sampleset.SampleSet method), [15](#)

## V

value (matk.observation.Observation attribute), [12](#)

value (matk.parameter.Parameter attribute), [12](#)

vary (matk.parameter.Parameter attribute), [12](#)

## W

weight (matk.observation.Observation attribute), [12](#)

workdir (matk.matk attribute), [11](#)

workdir\_base (matk.matk attribute), [11](#)

workdir\_index (matk.matk attribute), [11](#)