

EMPIRE DA

0.1

Generated by Doxygen 1.8.6

Tue Sep 23 2014 12:07:47



# Contents

<b>1</b>	<b>EMPIRE Data Assimilation Documentation</b>	<b>1</b>
1.1	Downloading . . . . .	1
1.2	Compiling . . . . .	1
1.2.1	Compilation of the source code . . . . .	1
1.2.2	Compilation of the documentation . . . . .	2
1.3	Customising for specific models . . . . .	2
1.4	Testing . . . . .	3
1.5	Linking to your model using EMPIRE . . . . .	3
1.6	Running . . . . .	3
1.7	Bug Reports and Functionality Requests . . . . .	3
<b>2</b>	<b>Todo List</b>	<b>5</b>
<b>3</b>	<b>Data Type Index</b>	<b>7</b>
3.1	Data Types List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Data Type Documentation</b>	<b>11</b>
5.1	comms Module Reference . . . . .	11
5.1.1	Detailed Description . . . . .	11
5.1.2	Member Function/Subroutine Documentation . . . . .	11
5.1.2.1	allocate_data . . . . .	11
5.1.2.2	deallocate_data . . . . .	12
5.1.2.3	initialise_mpi . . . . .	12
5.1.3	Member Data Documentation . . . . .	12
5.1.3.1	cpl_mpi_comm . . . . .	12
5.1.3.2	gblcount . . . . .	12
5.1.3.3	gbldisp . . . . .	12
5.1.3.4	mype_id . . . . .	12
5.1.3.5	myrank . . . . .	12
5.1.3.6	npfs . . . . .	12

5.1.3.7	<a href="#">nproc</a>	12
5.1.3.8	<a href="#">pf_mpi_comm</a>	12
5.1.3.9	<a href="#">pfrank</a>	13
5.2	<a href="#">histogram_data Module Reference</a>	13
5.2.1	<a href="#">Detailed Description</a>	13
5.2.2	<a href="#">Member Function/Subroutine Documentation</a>	13
5.2.2.1	<a href="#">kill_histogram_data</a>	13
5.2.2.2	<a href="#">load_histogram_data</a>	13
5.2.3	<a href="#">Member Data Documentation</a>	13
5.2.3.1	<a href="#">rank_hist_list</a>	13
5.2.3.2	<a href="#">rank_hist_nums</a>	13
5.2.3.3	<a href="#">rhl_n</a>	13
5.2.3.4	<a href="#">rhn_n</a>	13
5.3	<a href="#">hqht_plus_r Module Reference</a>	14
5.3.1	<a href="#">Member Function/Subroutine Documentation</a>	14
5.3.1.1	<a href="#">hqhtr_factor</a>	14
5.3.1.2	<a href="#">kill_hqhtr</a>	14
5.3.1.3	<a href="#">load_hqhtr</a>	14
5.4	<a href="#">pf_control Module Reference</a>	14
5.4.1	<a href="#">Detailed Description</a>	15
5.4.2	<a href="#">Member Function/Subroutine Documentation</a>	15
5.4.2.1	<a href="#">allocate_pf</a>	16
5.4.2.2	<a href="#">deallocate_pf</a>	16
5.4.2.3	<a href="#">set_pf_controls</a>	16
5.4.3	<a href="#">Member Data Documentation</a>	16
5.4.3.1	<a href="#">pf</a>	17
5.5	<a href="#">pf_control::pf_control_type Type Reference</a>	17
5.5.1	<a href="#">Member Data Documentation</a>	18
5.5.1.1	<a href="#">count</a>	18
5.5.1.2	<a href="#">couple_root</a>	18
5.5.1.3	<a href="#">efac</a>	18
5.5.1.4	<a href="#">gen_data</a>	18
5.5.1.5	<a href="#">gen_q</a>	18
5.5.1.6	<a href="#">human_readable</a>	18
5.5.1.7	<a href="#">init</a>	18
5.5.1.8	<a href="#">keep</a>	18
5.5.1.9	<a href="#">mean</a>	19
5.5.1.10	<a href="#">nens</a>	19
5.5.1.11	<a href="#">nfac</a>	19
5.5.1.12	<a href="#">nudgefacs</a>	19

5.5.1.13	particles	19
5.5.1.14	psi	19
5.5.1.15	qscale	19
5.5.1.16	talagrand	19
5.5.1.17	time	19
5.5.1.18	time_bwn_obs	19
5.5.1.19	time_obs	19
5.5.1.20	timestep	19
5.5.1.21	type	20
5.5.1.22	ufac	20
5.5.1.23	use_mean	20
5.5.1.24	use_rmse	20
5.5.1.25	use_talagrand	20
5.5.1.26	use_traj	20
5.5.1.27	use_var	20
5.5.1.28	use_weak	20
5.5.1.29	weight	20
5.6	qdata Module Reference	20
5.6.1	Detailed Description	21
5.6.2	Member Function/Subroutine Documentation	21
5.6.2.1	killq	21
5.6.2.2	loadq	21
5.6.3	Member Data Documentation	22
5.6.3.1	qcol	22
5.6.3.2	qdiag	22
5.6.3.3	qn	22
5.6.3.4	qne	22
5.6.3.5	qrow	22
5.6.3.6	qscale	22
5.6.3.7	qval	22
5.7	random Module Reference	22
5.7.1	Detailed Description	23
5.7.2	Member Function/Subroutine Documentation	23
5.7.2.1	bin_prob	23
5.7.2.2	lngamma	23
5.7.2.3	random_beta	24
5.7.2.4	random_binomial1	24
5.7.2.5	random_binomial2	24
5.7.2.6	random_cauchy	24
5.7.2.7	random_chisq	25

5.7.2.8	random_exponential	25
5.7.2.9	random_gamma	25
5.7.2.10	random_gamma1	26
5.7.2.11	random_gamma2	26
5.7.2.12	random_inv_gauss	27
5.7.2.13	random_mvnorm	27
5.7.2.14	random_neg_binomial	28
5.7.2.15	random_normal	28
5.7.2.16	random_order	29
5.7.2.17	random_poisson	30
5.7.2.18	random_t	30
5.7.2.19	random_von_mises	30
5.7.2.20	random_weibull	31
5.7.2.21	seed_random_number	31
5.7.3	Member Data Documentation	31
5.7.3.1	dp	31
5.8	rdata Module Reference	31
5.8.1	Detailed Description	32
5.8.2	Member Function/Subroutine Documentation	32
5.8.2.1	killr	32
5.8.2.2	loadr	32
5.8.3	Member Data Documentation	32
5.8.3.1	rcol	32
5.8.3.2	rdiag	32
5.8.3.3	rn	33
5.8.3.4	rne	33
5.8.3.5	rrow	33
5.8.3.6	rval	33
5.9	sizes Module Reference	33
5.9.1	Detailed Description	33
5.9.2	Member Data Documentation	33
5.9.2.1	obs_dim	33
5.9.2.2	state_dim	33
<b>6</b>	<b>File Documentation</b>	<b>35</b>
6.1	model_specific.f90 File Reference	35
6.1.1	Function/Subroutine Documentation	35
6.1.1.1	configure_model	35
6.1.1.2	h	37
6.1.1.3	ht	38

6.1.1.4	q	39
6.1.1.5	qhalf	41
6.1.1.6	r	42
6.1.1.7	rhalf	43
6.1.1.8	solve_hqht_plus_r	44
6.1.1.9	solve_r	45
6.2	src/controllers/old_pf_couple.f90 File Reference	45
6.2.1	Function/Subroutine Documentation	45
6.2.1.1	couple_pf	46
6.3	src/controllers/pf_control.f90 File Reference	46
6.4	src/controllers/pf_couple.f90 File Reference	46
6.4.1	Function/Subroutine Documentation	46
6.4.1.1	empire	47
6.5	src/controllers/sizes.f90 File Reference	47
6.6	src/data/Qdata.f90 File Reference	47
6.7	src/data/Rdata.f90 File Reference	48
6.8	src/DOC_README.txt File Reference	48
6.9	src/filters/eakf_analysis.f90 File Reference	48
6.9.1	Function/Subroutine Documentation	48
6.9.1.1	eakf_analysis	48
6.10	src/filters/enkf_specific.f90 File Reference	49
6.10.1	Function/Subroutine Documentation	49
6.10.1.1	get_local_observation_data	49
6.10.1.2	h_local	50
6.10.1.3	localise_enkf	50
6.10.1.4	solve_rhalf_local	51
6.11	src/filters/equivalent_weights_step.f90 File Reference	51
6.11.1	Function/Subroutine Documentation	51
6.11.1.1	equal_weight_filter	51
6.12	src/filters/etkf_analysis.f90 File Reference	52
6.12.1	Function/Subroutine Documentation	52
6.12.1.1	etkf_analysis	52
6.13	src/filters/proposal_filter.f90 File Reference	53
6.13.1	Function/Subroutine Documentation	53
6.13.1.1	proposal_filter	53
6.14	src/filters/sir_filter.f90 File Reference	53
6.14.1	Function/Subroutine Documentation	53
6.14.1.1	sir_filter	54
6.15	src/filters/stochastic_model.f90 File Reference	54
6.15.1	Function/Subroutine Documentation	54

6.15.1.1	check_scaling	54
6.15.1.2	stochastic_model	54
6.16	src/operations/gen_rand.f90 File Reference	55
6.16.1	Function/Subroutine Documentation	55
6.16.1.1	mixturerandomnumbers1d	55
6.16.1.2	mixturerandomnumbers2d	56
6.16.1.3	normalrandomnumbers1d	57
6.16.1.4	normalrandomnumbers2d	58
6.16.1.5	random_seed_mpi	58
6.16.1.6	uniformrandomnumbers1d	59
6.17	src/operations/operator_wrappers.f90 File Reference	59
6.17.1	Function/Subroutine Documentation	60
6.17.1.1	bprime	60
6.17.1.2	innerhqht_plus_r_1	60
6.17.1.3	innerr_1	61
6.17.1.4	k	62
6.18	src/operations/perturb_particle.f90 File Reference	63
6.18.1	Function/Subroutine Documentation	64
6.18.1.1	perturb_particle	64
6.18.1.2	update_state	64
6.19	src/operations/resample.f90 File Reference	65
6.19.1	Function/Subroutine Documentation	65
6.19.1.1	resample	65
6.20	src/tests/alltests.f90 File Reference	66
6.20.1	Function/Subroutine Documentation	66
6.20.1.1	alltests	66
6.21	src/tests/test_h.f90 File Reference	67
6.21.1	Function/Subroutine Documentation	67
6.21.1.1	test_h	67
6.22	src/tests/test_hqhtr.f90 File Reference	68
6.22.1	Function/Subroutine Documentation	68
6.22.1.1	test_hqhtr	68
6.23	src/tests/test_q.f90 File Reference	68
6.23.1	Function/Subroutine Documentation	69
6.23.1.1	test_q	69
6.24	src/tests/test_r.f90 File Reference	69
6.24.1	Function/Subroutine Documentation	69
6.24.1.1	test_r	69
6.25	src/tests/tests.f90 File Reference	70
6.25.1	Function/Subroutine Documentation	70



6.25.1.1	<a href="#">h_tests</a>	70
6.25.1.2	<a href="#">hqhtr_tests</a>	71
6.25.1.3	<a href="#">q_tests</a>	72
6.25.1.4	<a href="#">r_tests</a>	72
6.26	<a href="#">src/utls/comms.f90 File Reference</a>	73
6.27	<a href="#">src/utls/data_io.f90 File Reference</a>	73
6.27.1	<a href="#">Function/Subroutine Documentation</a>	73
6.27.1.1	<a href="#">get_observation_data</a>	73
6.27.1.2	<a href="#">output_from_pf</a>	74
6.27.1.3	<a href="#">save_observation_data</a>	74
6.27.1.4	<a href="#">save_truth</a>	75
6.28	<a href="#">src/utls/diagnostics.f90 File Reference</a>	75
6.28.1	<a href="#">Function/Subroutine Documentation</a>	75
6.28.1.1	<a href="#">diagnostics</a>	75
6.28.1.2	<a href="#">trajectories</a>	76
6.29	<a href="#">src/utls/genQ.f90 File Reference</a>	76
6.29.1	<a href="#">Function/Subroutine Documentation</a>	76
6.29.1.1	<a href="#">genq</a>	77
6.30	<a href="#">src/utls/histogram.f90 File Reference</a>	77
6.31	<a href="#">src/utls/quicksort.f90 File Reference</a>	77
6.31.1	<a href="#">Function/Subroutine Documentation</a>	77
6.31.1.1	<a href="#">insertionsort_d</a>	77
6.31.1.2	<a href="#">quicksort_d</a>	78
6.32	<a href="#">src/utls/random_d.f90 File Reference</a>	78
<b>Index</b>		<b>79</b>



# Chapter 1

## EMPIRE Data Assimilation Documentation

### Author

Philip A. Browne

### Date

Time-stamp: <2014-09-23 12:07:45 pbrowne>

## 1.1 Downloading

These codes are hosted on [www.bitbucket.org](http://www.bitbucket.org) and can be attained with the following commands:

```
git clone git@bitbucket.org:pbrowne/empire-data-assimilation.git
```

or

```
wget https://bitbucket.org/pbrowne/empire-data-assimilation/get/aa31fdfc3912.zip && gunzip aa31fdfc3912.zip
```

## 1.2 Compiling

### 1.2.1 Compilation of the source code

The Makefile must be edited for the specific compiler setup. In the main directory you will find the file `Makefile`.

Edit the variables as follows:

- `FC` The fortran compiler
- `FCOPTS` The options for the fortran compiler
- `LIB_LIST` The libraries to be called. Note this must include BLAS

To compile the source code, simply then type the command

```
make
```

If successful, the following executables are created in the `bin/` folder:

- `empire`

- [alltests](#)
- [test\\_h](#)
- [test\\_hqhtr](#)
- [test\\_q](#)
- [test\\_r](#)

To remove the object and executable files if compilation fails for some reason, run the following:

```
make clean
```

### 1.2.2 Compilation of the documentation

Documentation of the code is automatically generated using Doxygen, dot and pdflatex.

All of these packages must be installed for the following to work.

```
make docs
```

This will make an html webpage for the code, the mainpage for which is located in doc/html/index.html.

A latex version of the documentation will be built to the file doc/latex/refman.pdf.

To simply make the html version of the documentation (if pdflatex is not available) then use the command

```
make doc_html
```

## 1.3 Customising for specific models

*This is where the science and all the effort should happen!!*

The file [model\\_specific.f90](#) should be edited for the specific model which you wish to use. This contains a number of subroutines which need to be adapted for the model and the observation network. We list these subsequently.

- [configure\\_model](#) This is called early in the code and can be used to read in any data from files before subsequently using them in the below operations.
- [h](#) This is the observation operator
- [ht](#) This is the transpose of the observation operator
- [r](#) This is the observation error covariance matrix  $R$
- [rhalf](#) This is the square root of the observation error covariance matrix  $R^{\frac{1}{2}}$
- [solve\\_r](#) This is a linear solve with the observation error covariance matrix, i.e. given  $b$ , find  $x$  such that  $Rx = b$  or indeed,  $x = R^{-1}b$
- [q](#) This is the model error covariance matrix  $Q$
- [qhalf](#) This is the square root model error covariance matrix  $Q^{\frac{1}{2}}$
- [solve\\_hqht\\_plus\\_r](#) This is a linear solve with the matrix  $(HQH^T + R)$

Not all of these subroutines will be required for each filtering method you wish to use, so it may be advantageous to only implement the necessary ones.

## 1.4 Testing

You can test your user supplied routines by running the test codes found in the folder bin/.

These are by no means full-proof ways of ensuring that you have implemented things correctly, but should at least check what you have done for logical consistency.

For example, they will test if  $HH^T x = x$ , and if  $Q^{\frac{1}{2}} Q^{\frac{1}{2}} x = Qx$  for various different vectors  $x$ .

## 1.5 Linking to your model using EMPIRE

**Todo** Write a bit about how to put the MPI commands into the model, or point to where that is. [www.met.-reading.ac.uk/~darc/empire](http://www.met.rdg.ac.uk/~darc/empire)

## 1.6 Running

For example, to run **N\_MDL** copies of the model with **N\_DA** copies of empire, then the following are possible:

```
mpirun -np N_MDL model_executable : -np N_DA empire
```

```
aprun -n N_MDL -N N_MDL model_executable : -n N_DA -N N_DA empire
```

**Todo** I have to talk about how pf\_parameters.dat works.

## 1.7 Bug Reports and Functionality Requests

While the code is not too large, you may email me the issue or request [here](#).

However there is a webpage set up for this:

<https://bitbucket.org/pbrowne/empire-data-assimilation/issues>



## Chapter 2

## Todo List

page [EMPIRE Data Assimilation Documentation](#)

Write a bit about how to put the MPI commands into the model, or point to where that is. [www.met.-reading.ac.uk/~darc/empire](http://www.met.rdg.ac.uk/~darc/empire)

I have to talk about how pf\_parameters.dat works.





## Chapter 3

# Data Type Index

### 3.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">comms</a>	Module containing EMPIRE coupling data . . . . .	11
<a href="#">histogram_data</a>	Module to control what variables are used to generate rank histograms . . . . .	13
<a href="#">hqht_plus_r</a>	. . . . .	14
<a href="#">pf_control</a>	Module to hold all the information to control the the main program . . . . .	14
<a href="#">pf_control::pf_control_type</a>	. . . . .	17
<a href="#">qdata</a>	Module as a place to store user specified data for $Q$ . . . . .	20
<a href="#">random</a>	A module for random number generation from the following distributions: . . . . .	22
<a href="#">rdata</a>	Module to hold user supplied data for $R$ observation error covariance matrix . . . . .	31
<a href="#">sizes</a>	Module that stores the dimension of observation and state spaces . . . . .	33



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">model_specific.f90</a>	35
<a href="#">src/controllers/old_pf_couple.f90</a>	45
<a href="#">src/controllers/pf_control.f90</a>	46
<a href="#">src/controllers/pf_couple.f90</a>	46
<a href="#">src/controllers/sizes.f90</a>	47
<a href="#">src/data/Qdata.f90</a>	47
<a href="#">src/data/Rdata.f90</a>	48
<a href="#">src/filters/eakf_analysis.f90</a>	48
<a href="#">src/filters/enkf_specific.f90</a>	49
<a href="#">src/filters/equivalent_weights_step.f90</a>	51
<a href="#">src/filters/etkf_analysis.f90</a>	52
<a href="#">src/filters/proposal_filter.f90</a>	53
<a href="#">src/filters/sir_filter.f90</a>	53
<a href="#">src/filters/stochastic_model.f90</a>	54
<a href="#">src/operations/gen_rand.f90</a>	55
<a href="#">src/operations/operator_wrappers.f90</a>	59
<a href="#">src/operations/perturb_particle.f90</a>	63
<a href="#">src/operations/resample.f90</a>	65
<a href="#">src/tests/alltests.f90</a>	66
<a href="#">src/tests/test_h.f90</a>	67
<a href="#">src/tests/test_hqhtr.f90</a>	68
<a href="#">src/tests/test_q.f90</a>	68
<a href="#">src/tests/test_r.f90</a>	69
<a href="#">src/tests/tests.f90</a>	70
<a href="#">src/utills/comms.f90</a>	73
<a href="#">src/utills/data_io.f90</a>	73
<a href="#">src/utills/diagnostics.f90</a>	75
<a href="#">src/utills/genQ.f90</a>	76
<a href="#">src/utills/histogram.f90</a>	77
<a href="#">src/utills/quicksort.f90</a>	77
<a href="#">src/utills/random_d.f90</a>	78



## Chapter 5

# Data Type Documentation

### 5.1 comms Module Reference

Module containing EMPIRE coupling data.

#### Public Member Functions

- subroutine [allocate\\_data](#)
- subroutine [deallocate\\_data](#)
- subroutine [initialise\\_mpi](#)

*subroutine to make EMPIRE connections and saves details into [pf\\_control](#) module*

#### Public Attributes

- integer [cpl\\_mpi\\_comm](#)
- integer [mype\\_id](#)
- integer [myrank](#)
- integer [nproc](#)
- integer [pf\\_mpi\\_comm](#)
- integer [pfrank](#)
- integer [npfs](#)
- integer, dimension(:), allocatable [gblcount](#)
- integer, dimension(:), allocatable [gbldisp](#)

#### 5.1.1 Detailed Description

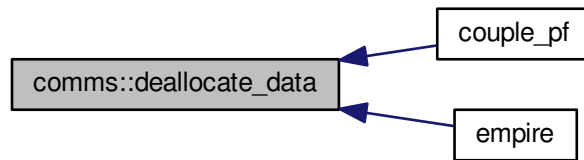
Module containing EMPIRE coupling data.

#### 5.1.2 Member Function/Subroutine Documentation

##### 5.1.2.1 subroutine `comms::allocate_data` ( )

### 5.1.2.2 subroutine comms::deallocate\_data ( )

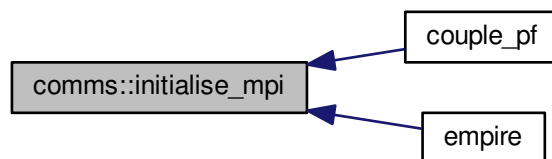
Here is the caller graph for this function:



### 5.1.2.3 subroutine comms::initialise\_mpi ( )

subroutine to make EMPIRE connections and saves details into `pf_control` module

Here is the caller graph for this function:



## 5.1.3 Member Data Documentation

5.1.3.1 integer `comms::cpl_mpi_comm`

5.1.3.2 integer, dimension(:), allocatable `comms::gblcount`

5.1.3.3 integer, dimension(:), allocatable `comms::gbldisp`

5.1.3.4 integer `comms::mype_id`

5.1.3.5 integer `comms::myrank`

5.1.3.6 integer `comms::npfs`

5.1.3.7 integer `comms::nproc`

5.1.3.8 integer `comms::pf_mpi_comm`

## 5.1.3.9 integer comms::pfrank

The documentation for this module was generated from the following file:

- [src/utils/comms.f90](#)

## 5.2 histogram\_data Module Reference

Module to control what variables are used to generate rank histograms.

### Public Member Functions

- subroutine [load\\_histogram\\_data](#)  
*subroutine to read from variables\_hist.dat which variables to be used to make the rank histograms*
- subroutine [kill\\_histogram\\_data](#)  
*subroutine to clean up arrays used in rank histograms*

### Public Attributes

- integer, dimension(:), allocatable [rank\\_hist\\_list](#)
- integer, dimension(:), allocatable [rank\\_hist\\_nums](#)
- integer [rhl\\_n](#)
- integer [rhn\\_n](#)

### 5.2.1 Detailed Description

Module to control what variables are used to generate rank histograms.

### 5.2.2 Member Function/Subroutine Documentation

#### 5.2.2.1 subroutine histogram\_data::kill\_histogram\_data ( )

subroutine to clean up arrays used in rank histograms

#### 5.2.2.2 subroutine histogram\_data::load\_histogram\_data ( )

subroutine to read from variables\_hist.dat which variables to be used to make the rank histograms

### 5.2.3 Member Data Documentation

#### 5.2.3.1 integer, dimension(:), allocatable histogram\_data::rank\_hist\_list

#### 5.2.3.2 integer, dimension(:), allocatable histogram\_data::rank\_hist\_nums

#### 5.2.3.3 integer histogram\_data::rhl\_n

#### 5.2.3.4 integer histogram\_data::rhn\_n

The documentation for this module was generated from the following file:

- [src/utils/histogram.f90](#)

## 5.3 hqht\_plus\_r Module Reference

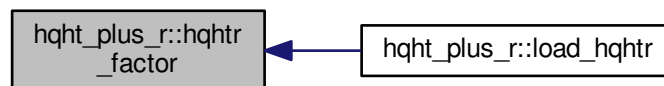
### Public Member Functions

- subroutine [load\\_hqhtr](#)
- subroutine [hqhtr\\_factor](#)
- subroutine [kill\\_hqhtr](#)

#### 5.3.1 Member Function/Subroutine Documentation

##### 5.3.1.1 subroutine hqht\_plus\_r::hqhtr\_factor ( )

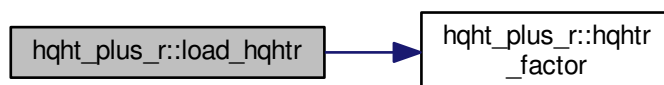
Here is the caller graph for this function:



##### 5.3.1.2 subroutine hqht\_plus\_r::kill\_hqhtr ( )

##### 5.3.1.3 subroutine hqht\_plus\_r::load\_hqhtr ( )

Here is the call graph for this function:



The documentation for this module was generated from the following file:

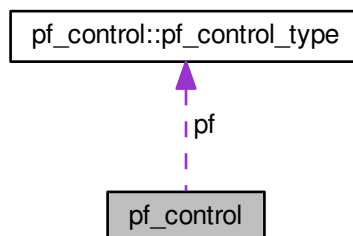
- [src/data/Rdata.f90](#)

## 5.4 pf\_control Module Reference

module to hold all the information to control the the main program



Collaboration diagram for pf\_control:



## Data Types

- type `pf_control_type`

## Public Member Functions

- subroutine `set_pf_controls`
- subroutine `allocate_pf`
- subroutine `deallocate_pf`

## Public Attributes

- type(`pf_control_type`) `pf`

*the derived data type holding all controlling data*

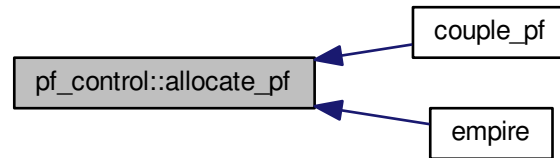
### 5.4.1 Detailed Description

module to hold all the information to control the the main program

### 5.4.2 Member Function/Subroutine Documentation

#### 5.4.2.1 subroutine pf\_control::allocate\_pf ( )

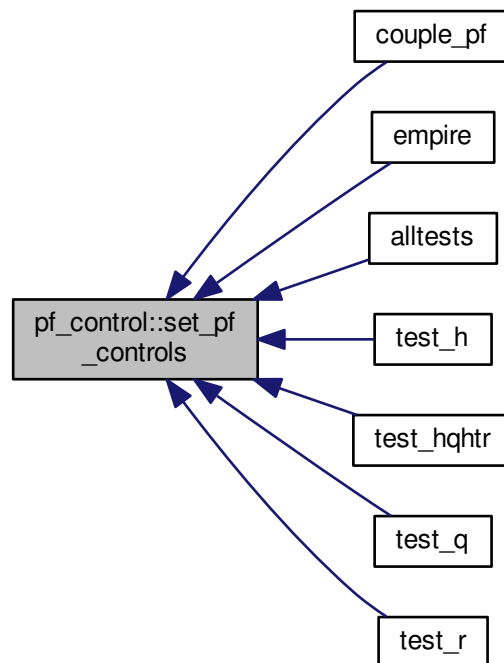
Here is the caller graph for this function:



#### 5.4.2.2 subroutine pf\_control::deallocate\_pf ( )

#### 5.4.2.3 subroutine pf\_control::set\_pf\_controls ( )

Here is the caller graph for this function:



### 5.4.3 Member Data Documentation

## 5.4.3.1 type(pf\_control\_type) pf\_control::pf

the derived data type holding all controlling data

The documentation for this module was generated from the following file:

- src/controllers/pf\_control.f90

## 5.5 pf\_control::pf\_control\_type Type Reference

## Public Attributes

- integer [nens](#)  
*the total number of ensemble members*
- real(kind=kind(1.0d0)), dimension(:), allocatable [weight](#)  
*the negative log of the weights of the particles*
- integer [time\\_obs](#)  
*the number of observations we will assimilate*
- integer [time\\_bwn\\_obs](#)  
*the number of model timesteps between observations*
- real(kind=kind(1.0d0)) [nudgefac](#)  
*the nudging factor*
- logical [gen\\_data](#)  
*true generates synthetic obs for a twin experiment*
- logical [gen\\_q](#)  
*true attempts to build up Q from long model run*
- logical [human\\_readable](#)  
*unused*
- integer [timestep](#) = 0  
*the current timestep as the model progresses*
- real(kind=kind(1.0d0)), dimension(:, :), allocatable [psi](#)  
*state vector of ensemble members on this mpi process*
- real(kind=kind(1.0d0)), dimension(:), allocatable [mean](#)  
*mean state vector*
- real(kind=kind(1.0d0)) [nfac](#)  
*standard deviation of normal distribution in mixture density*
- real(kind=kind(1.0d0)) [ufac](#)  
*half width of the uniform distribution in mixture density*
- real(kind=kind(1.0d0)) [efac](#)
- real(kind=kind(1.0d0)) [keep](#)  
*proportion of particles to keep in EWPF EW step*
- real(kind=kind(1.0d0)) [time](#)  
*dunno*
- real(kind=kind(1.0d0)) [qscale](#)  
*scalar to multiply Q by*
- integer [couple\\_root](#)  
*empire master processor*
- logical [use\\_talagrand](#)  
*switch if true outputs rank histograms*
- logical [use\\_weak](#)  
*switch unused*

- logical `use_mean`  
*switch if true outputs ensemble mean*
- logical `use_var`  
*switch if true outputs ensemble variance*
- logical `use_traj`  
*switch if true outputs trajectories*
- logical `use_rmse`  
*switch if true outputs Root Mean Square Errors*
- integer, dimension(:,:), allocatable `talagrand`  
*storage for rank histograms*
- integer `count`  
*number of ensemble members associated with this MPI process*
- integer, dimension(:), allocatable `particles`  
*particles associates with this MPI process*
- character(2) `type`  
*which filter to use*
- character(1) `init`  
*which method to initialise ensemble*

### 5.5.1 Member Data Documentation

#### 5.5.1.1 integer `pf_control::pf_control_type::count`

number of ensemble members associated with this MPI process

#### 5.5.1.2 integer `pf_control::pf_control_type::couple_root`

empire master processor

#### 5.5.1.3 real(kind=kind(1.0d0)) `pf_control::pf_control_type::efac`

#### 5.5.1.4 logical `pf_control::pf_control_type::gen_data`

true generates synthetic obs for a twin experiment

#### 5.5.1.5 logical `pf_control::pf_control_type::gen_q`

true attempts to build up  $Q$  from long model run

#### 5.5.1.6 logical `pf_control::pf_control_type::human_readable`

unused

#### 5.5.1.7 character(1) `pf_control::pf_control_type::init`

which method to initialise ensemble

#### 5.5.1.8 real(kind=kind(1.0d0)) `pf_control::pf_control_type::keep`

proportion of particles to keep in EWPF EW step

5.5.1.9 `real(kind=kind(1.0d0)), dimension(:), allocatable pf_control::pf_control_type::mean`

mean state vector

5.5.1.10 `integer pf_control::pf_control_type::nens`

the total number of ensemble members

5.5.1.11 `real(kind=kind(1.0d0)) pf_control::pf_control_type::nfac`

standard deviation of normal distribution in mixture density

5.5.1.12 `real(kind=kind(1.0d0)) pf_control::pf_control_type::nudgefac`

the nudging factor

5.5.1.13 `integer, dimension(:), allocatable pf_control::pf_control_type::particles`

particles associates with this MPI process

5.5.1.14 `real(kind=kind(1.0d0)), dimension(:, :), allocatable pf_control::pf_control_type::psi`

state vector of ensemble members on this mpi process

5.5.1.15 `real(kind=kind(1.0d0)) pf_control::pf_control_type::qscale`

scalar to multiply Q by

5.5.1.16 `integer, dimension(:, :), allocatable pf_control::pf_control_type::talagrand`

storage for rank histograms

5.5.1.17 `real(kind=kind(1.0d0)) pf_control::pf_control_type::time`

dunno

5.5.1.18 `integer pf_control::pf_control_type::time_bwn_obs`

the number of model timesteps between observations

5.5.1.19 `integer pf_control::pf_control_type::time_obs`

the number of observations we will assimilate

5.5.1.20 `integer pf_control::pf_control_type::timestep = 0`

the current timestep as the model progresses

5.5.1.21 `character(2) pf_control::pf_control_type::type`

which filter to use

5.5.1.22 `real(kind=kind(1.0d0)) pf_control::pf_control_type::ufac`

half width of the uniform distribution in mixture density

5.5.1.23 `logical pf_control::pf_control_type::use_mean`

switch if true outputs ensemble mean

5.5.1.24 `logical pf_control::pf_control_type::use_rmse`

switch if true outputs Root Mean Square Errors

5.5.1.25 `logical pf_control::pf_control_type::use_talagrand`

switch if true outputs rank histograms

5.5.1.26 `logical pf_control::pf_control_type::use_traj`

switch if true outputs trajectories

5.5.1.27 `logical pf_control::pf_control_type::use_var`

switch if true outputs ensemble variance

5.5.1.28 `logical pf_control::pf_control_type::use_weak`

switch unused

5.5.1.29 `real(kind=kind(1.0d0)), dimension(:), allocatable pf_control::pf_control_type::weight`

the negative log of the weights of the particles

The documentation for this type was generated from the following file:

- [src/controllers/pf\\_control.f90](#)

## 5.6 qdata Module Reference

Module as a place to store user specified data for  $Q$ .

### Public Member Functions

- subroutine [loadq](#)  
*Subroutine to load in user data for  $Q$ .*
- subroutine [killq](#)

## Public Attributes

- integer [qn](#)
- integer [qne](#)
- integer, dimension(:), allocatable [qrow](#)
- integer, dimension(:), allocatable [qcol](#)
- real(kind=kind(1.0d0)), dimension(:), allocatable [qval](#)
- real(kind=kind(1.0d0)), dimension(:), allocatable [qdiag](#)
- real(kind=kind(1.0d0)) [qscale](#)

### 5.6.1 Detailed Description

Module as a place to store user specified data for  $Q$ .

- the model error covariance matrix

### 5.6.2 Member Function/Subroutine Documentation

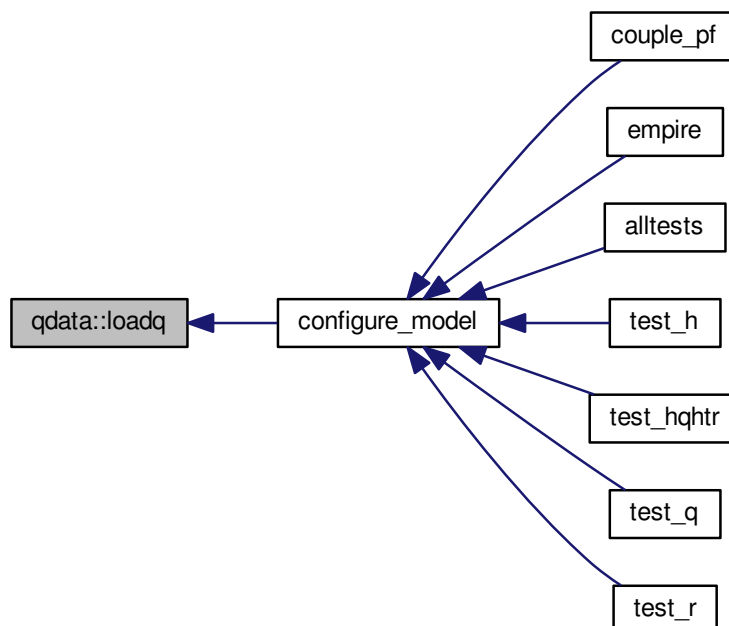
#### 5.6.2.1 subroutine qdata::killq ( )

Subroutine to deallocate user data for  $Q$

#### 5.6.2.2 subroutine qdata::loadq ( )

Subroutine to load in user data for  $Q$ .

Here is the caller graph for this function:



### 5.6.3 Member Data Documentation

5.6.3.1 integer, dimension(:), allocatable qdata::qcol

5.6.3.2 real(kind=kind(1.0d0)), dimension(:), allocatable qdata::qdiag

5.6.3.3 integer qdata::qn

5.6.3.4 integer qdata::qne

5.6.3.5 integer, dimension(:), allocatable qdata::qrow

5.6.3.6 real(kind=kind(1.0d0)) qdata::qscale

5.6.3.7 real(kind=kind(1.0d0)), dimension(:), allocatable qdata::qval

The documentation for this module was generated from the following file:

- src/data/[Qdata.f90](#)

## 5.7 random Module Reference

A module for random number generation from the following distributions:

### Public Member Functions

- real(kind=kind(1.0d+0)) function [random\\_normal](#) ()  
*function to get random normal with zero mean and stdev 1*
- real(kind=kind(1.0d+0)) function [random\\_gamma](#) (s, first)
- real(kind=kind(1.0d+0)) function [random\\_gamma1](#) (s, first)
- real(kind=kind(1.0d+0)) function [random\\_gamma2](#) (s, first)
- real(kind=kind(1.0d+0)) function [random\\_chisq](#) (ndf, first)
- real(kind=kind(1.0d+0)) function [random\\_exponential](#) ()
- real(kind=kind(1.0d+0)) function [random\\_weibull](#) (a)
- real(kind=kind(1.0d+0)) function [random\\_beta](#) (aa, bb, first)
- real(kind=kind(1.0d+0)) function [random\\_t](#) (m)
- subroutine [random\\_mvnorm](#) (n, h, d, f, first, x, ier)
- real(kind=kind(1.0d+0)) function [random\\_inv\\_gauss](#) (h, b, first)
- integer function [random\\_poisson](#) (mu, first)
- integer function [random\\_binomial1](#) (n, p, first)
- real(kind=kind(1.0d+0)) function [bin\\_prob](#) (n, p, r)
- real([dp](#)) function [lngamma](#) (x)
- integer function [random\\_binomial2](#) (n, pp, first)
- integer function [random\\_neg\\_binomial](#) (sk, p)
- real(kind=kind(1.0d+0)) function [random\\_von\\_mises](#) (k, first)
- real(kind=kind(1.0d+0)) function [random\\_cauchy](#) ()
- subroutine [random\\_order](#) (order, n)
- subroutine [seed\\_random\\_number](#) (iounit)

### Public Attributes

- integer, parameter [dp](#) = SELECTED\_REAL\_KIND(12, 60)



### 5.7.1 Detailed Description

A module for random number generation from the following distributions:

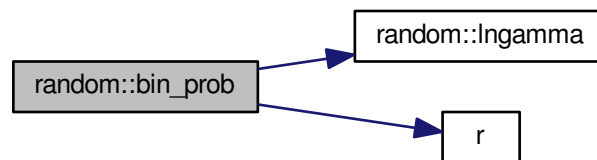
Distribution Function/subroutine name

Normal (Gaussian) random\_normal Gamma random\_gamma Chi-squared random\_chisq Exponential random\_exponential Weibull random\_Weibull Beta random\_beta t random\_t Multivariate normal random\_mvnorm Generalized inverse Gaussian random\_inv\_gauss Poisson random\_Poisson Binomial random\_binomial1 \* random\_binomial2 \* Negative binomial random\_neg\_binomial von Mises random\_von\_Mises Cauchy random\_Cauchy

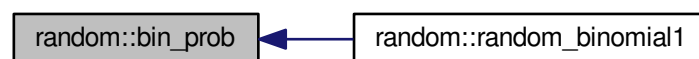
### 5.7.2 Member Function/Subroutine Documentation

5.7.2.1 `real(kind=kind(1.0d+0)) function random::bin_prob ( integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) p, integer, intent(in) r )`

Here is the call graph for this function:

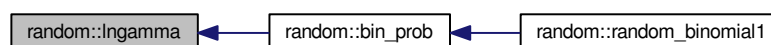


Here is the caller graph for this function:



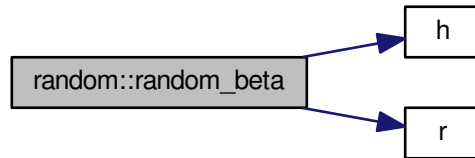
5.7.2.2 `real(dp) function random::lngamma ( real(dp), intent(in) x )`

Here is the caller graph for this function:



5.7.2.3 `real(kind=kind(1.0d+0)) function random::random_beta ( real(kind=kind(1.0d+0)), intent(in) aa, real(kind=kind(1.0d+0)), intent(in) bb, logical, intent(in) first )`

Here is the call graph for this function:



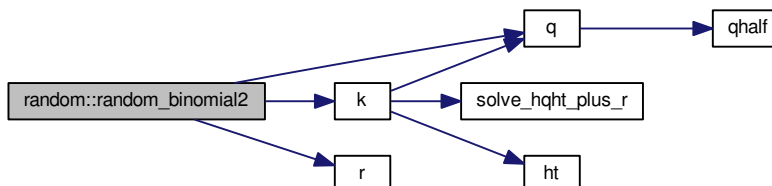
5.7.2.4 `integer function random::random_binomial1 ( integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) p, logical, intent(in) first )`

Here is the call graph for this function:



5.7.2.5 `integer function random::random_binomial2 ( integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) pp, logical, intent(in) first )`

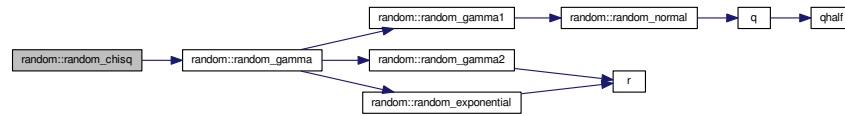
Here is the call graph for this function:



5.7.2.6 `real(kind=kind(1.0d+0)) function random::random_cauchy ( )`

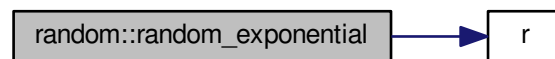
5.7.2.7 `real(kind=kind(1.0d+0)) function random::random_chisq ( integer, intent(in) ndf, logical, intent(in) first )`

Here is the call graph for this function:

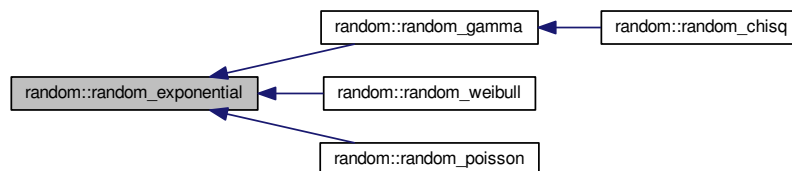


5.7.2.8 `real(kind=kind(1.0d+0)) function random::random_exponential ( )`

Here is the call graph for this function:

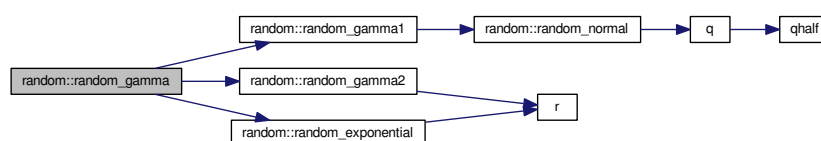


Here is the caller graph for this function:



5.7.2.9 `real(kind=kind(1.0d+0)) function random::random_gamma ( real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first )`

Here is the call graph for this function:

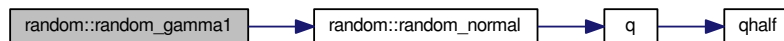


Here is the caller graph for this function:

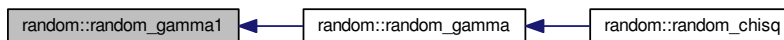


5.7.2.10 `real(kind=kind(1.0d+0)) function random::random_gamma1 ( real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first )`

Here is the call graph for this function:

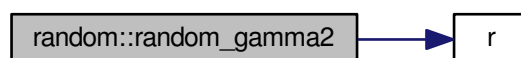


Here is the caller graph for this function:

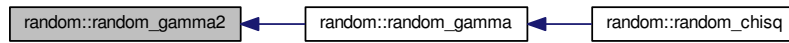


5.7.2.11 `real(kind=kind(1.0d+0)) function random::random_gamma2 ( real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first )`

Here is the call graph for this function:

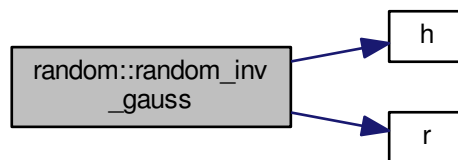


Here is the caller graph for this function:



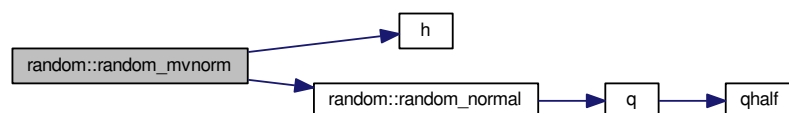
5.7.2.12 `real(kind=kind(1.0d+0)) function random::random_inv_gauss ( real(kind=kind(1.0d+0)), intent(in) h,  
real(kind=kind(1.0d+0)), intent(in) b, logical, intent(in) first )`

Here is the call graph for this function:



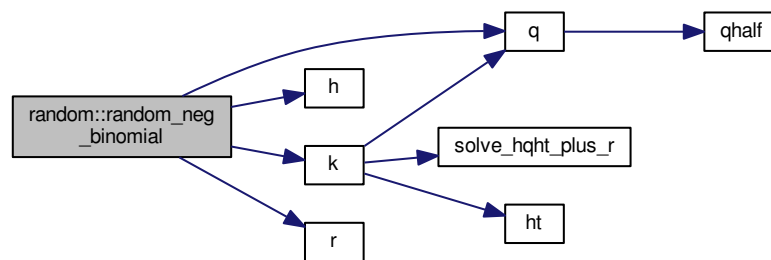
5.7.2.13 `subroutine random::random_mvnorm ( integer, intent(in) n, real(kind=kind(1.0d+0)), dimension(:), intent(in) h,  
real(kind=kind(1.0d+0)), dimension(:), intent(in) d, real(kind=kind(1.0d+0)), dimension(:), intent(inout) f, logical,  
intent(in) first, real(kind=kind(1.0d+0)), dimension(:), intent(out) x, integer, intent(out) ier )`

Here is the call graph for this function:



5.7.2.14 integer function `random::random_neg_binomial` ( `real(kind=kind(1.0d+0))`, intent(in) `sk`, `real(kind=kind(1.0d+0))`, intent(in) `p` )

Here is the call graph for this function:



5.7.2.15 `real(kind=kind(1.0d+0))` function `random::random_normal` ( )

function to get random normal with zero mean and stdev 1

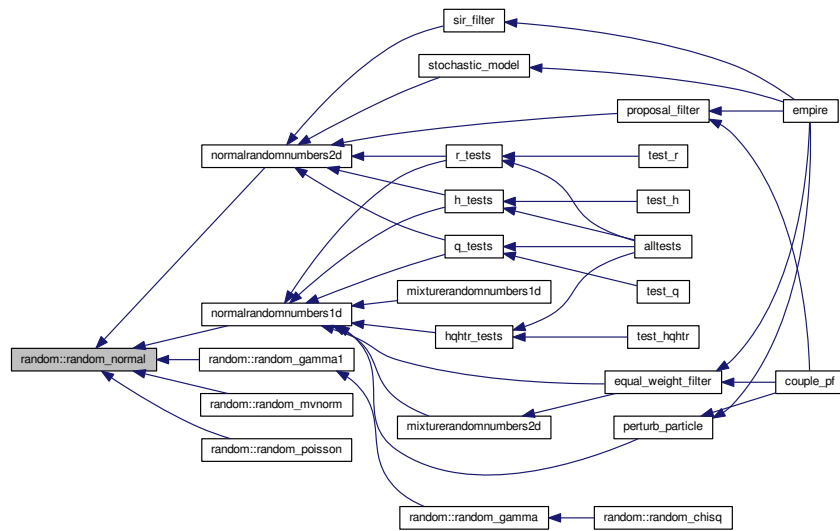
#### Returns

`fn_val`

Here is the call graph for this function:

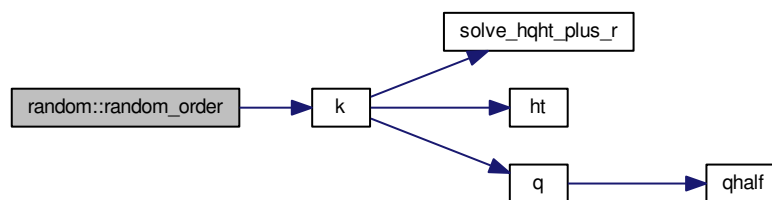


Here is the caller graph for this function:



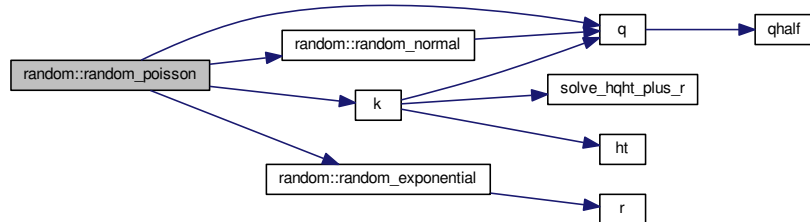
5.7.2.16 subroutine `random::random_order` ( integer, dimension(n), intent(out) *order*, integer, intent(in) *n* )

Here is the call graph for this function:



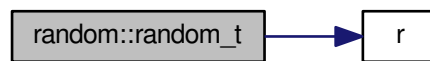
5.7.2.17 integer function random::random\_poisson ( real(kind=kind(1.0d+0)), intent(in) *mu*, logical, intent(in) *first* )

Here is the call graph for this function:



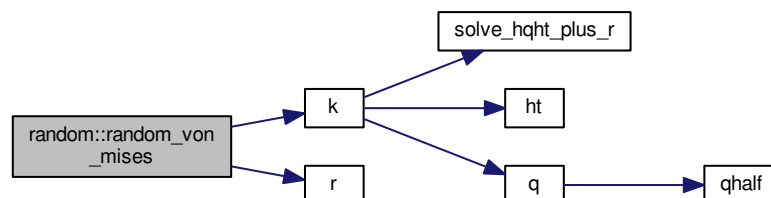
5.7.2.18 real(kind=kind(1.0d+0)) function random::random\_t ( integer, intent(in) *m* )

Here is the call graph for this function:



5.7.2.19 real(kind=kind(1.0d+0)) function random::random\_von\_mises ( real(kind=kind(1.0d+0)), intent(in) *k*, logical, intent(in) *first* )

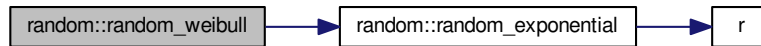
Here is the call graph for this function:





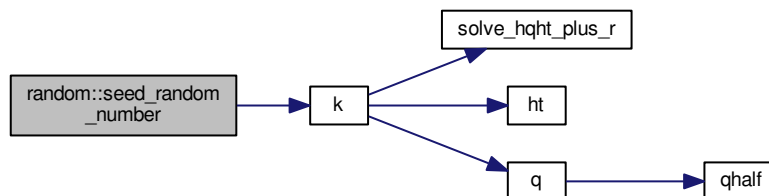
5.7.2.20 `real(kind=kind(1.0d+0)) function random::random_weibull ( real(kind=kind(1.0d+0)), intent(in) a )`

Here is the call graph for this function:



5.7.2.21 `subroutine random::seed_random_number ( integer, intent(in) iounit )`

Here is the call graph for this function:



## 5.7.3 Member Data Documentation

5.7.3.1 `integer, parameter random::dp = SELECTED_REAL_KIND(12, 60)`

The documentation for this module was generated from the following file:

- [src/utils/random\\_d.f90](#)

## 5.8 rdata Module Reference

Module to hold user supplied data for  $R$  observation error covariance matrix.

### Public Member Functions

- subroutine [loadr](#)  
Subroutine to load data for  $R$ .
- subroutine [killr](#)

### Public Attributes

- integer [rn](#)
- integer [rne](#)

- integer, dimension(:), allocatable [row](#)
- integer, dimension(:), allocatable [rcol](#)
- real(kind=kind(1.0d0)), dimension(:), allocatable [rval](#)
- real(kind=kind(1.0d0)), dimension(:), allocatable [rdiag](#)

### 5.8.1 Detailed Description

Module to hold user supplied data for  $R$  observation error covariance matrix.

### 5.8.2 Member Function/Subroutine Documentation

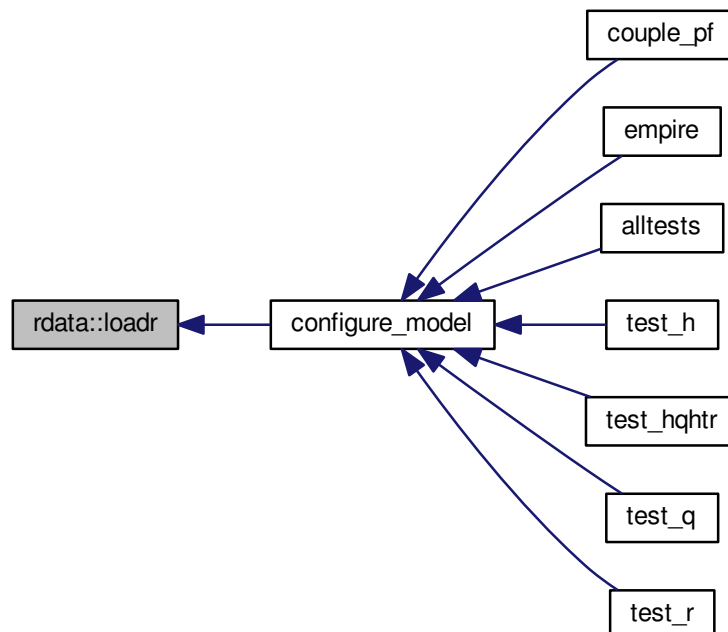
#### 5.8.2.1 subroutine `rdata::killr ( )`

SUbroutine to deallocate R data

#### 5.8.2.2 subroutine `rdata::loadr ( )`

Subroutine to load data for R.

Here is the caller graph for this function:



### 5.8.3 Member Data Documentation

#### 5.8.3.1 integer, dimension(:), allocatable `rdata::rcol`

#### 5.8.3.2 real(kind=kind(1.0d0)), dimension(:), allocatable `rdata::rdiag`

5.8.3.3 integer rdata::rn

5.8.3.4 integer rdata::rne

5.8.3.5 integer, dimension(:), allocatable rdata::rrow

5.8.3.6 real(kind=kind(1.0d0)), dimension(:), allocatable rdata::rval

The documentation for this module was generated from the following file:

- src/data/[Rdata.f90](#)

## 5.9 sizes Module Reference

Module that stores the dimension of observation and state spaces.

### Public Attributes

- integer [obs\\_dim](#)  
*size of the observation space*
- integer [state\\_dim](#)  
*dimension of the model*

### 5.9.1 Detailed Description

Module that stores the dimension of observation and state spaces.

### 5.9.2 Member Data Documentation

5.9.2.1 integer sizes::obs\_dim

size of the observation space

5.9.2.2 integer sizes::state\_dim

dimension of the model

The documentation for this module was generated from the following file:

- src/controllers/[sizes.f90](#)



## Chapter 6

# File Documentation

### 6.1 model\_specific.f90 File Reference

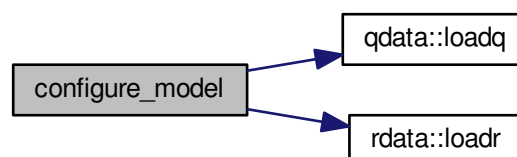
#### Functions/Subroutines

- subroutine [configure\\_model](#)
- subroutine [solve\\_r](#) (y, v, t)
- subroutine [solve\\_hqht\\_plus\\_r](#) (y, v, t)
- subroutine [q](#) (nrhs, x, Qx)
- subroutine [qhalf](#) (nrhs, x, Qx)
- subroutine [r](#) (nrhs, y, Ry, t)
- subroutine [rhalf](#) (nrhs, y, Ry, t)
- subroutine [h](#) (x, hx, t)
- subroutine [ht](#) (y, x, t)

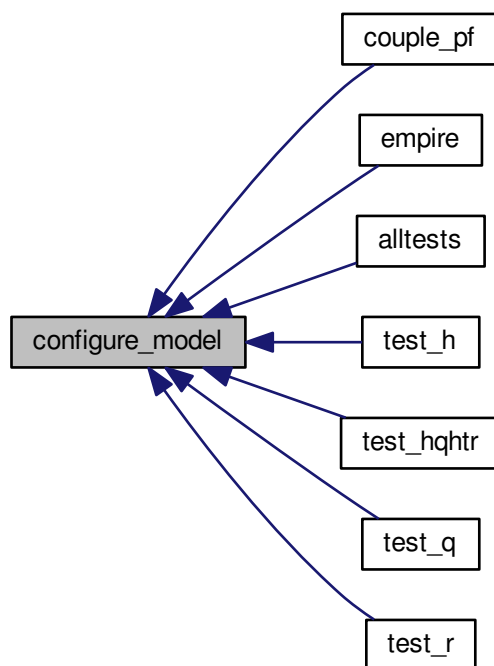
#### 6.1.1 Function/Subroutine Documentation

##### 6.1.1.1 subroutine [configure\\_model](#) ( )

Here is the call graph for this function:

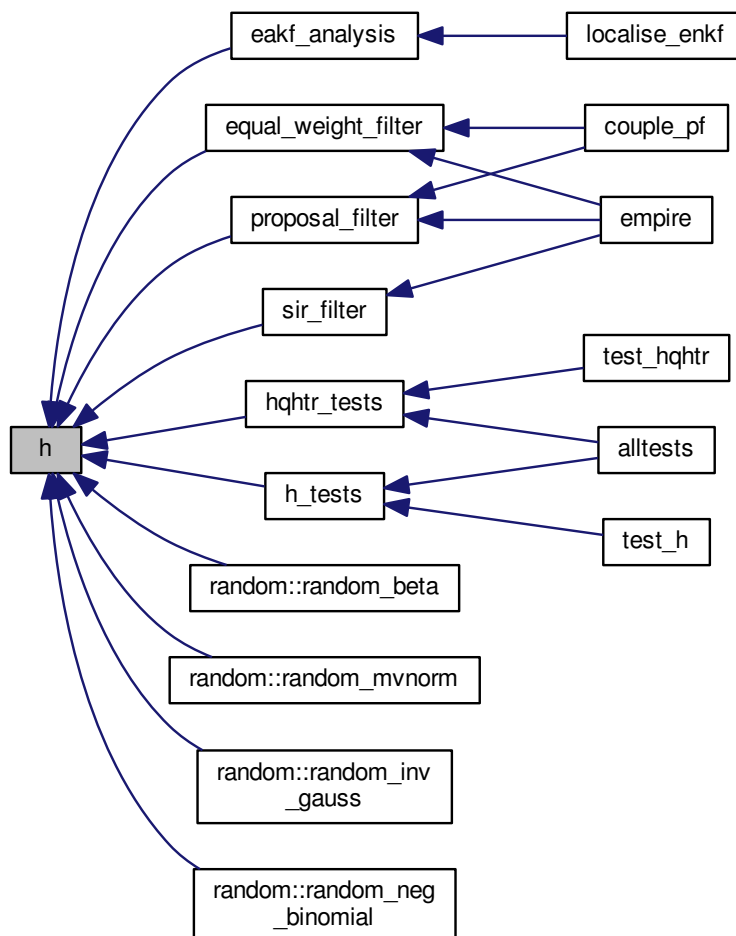


Here is the caller graph for this function:



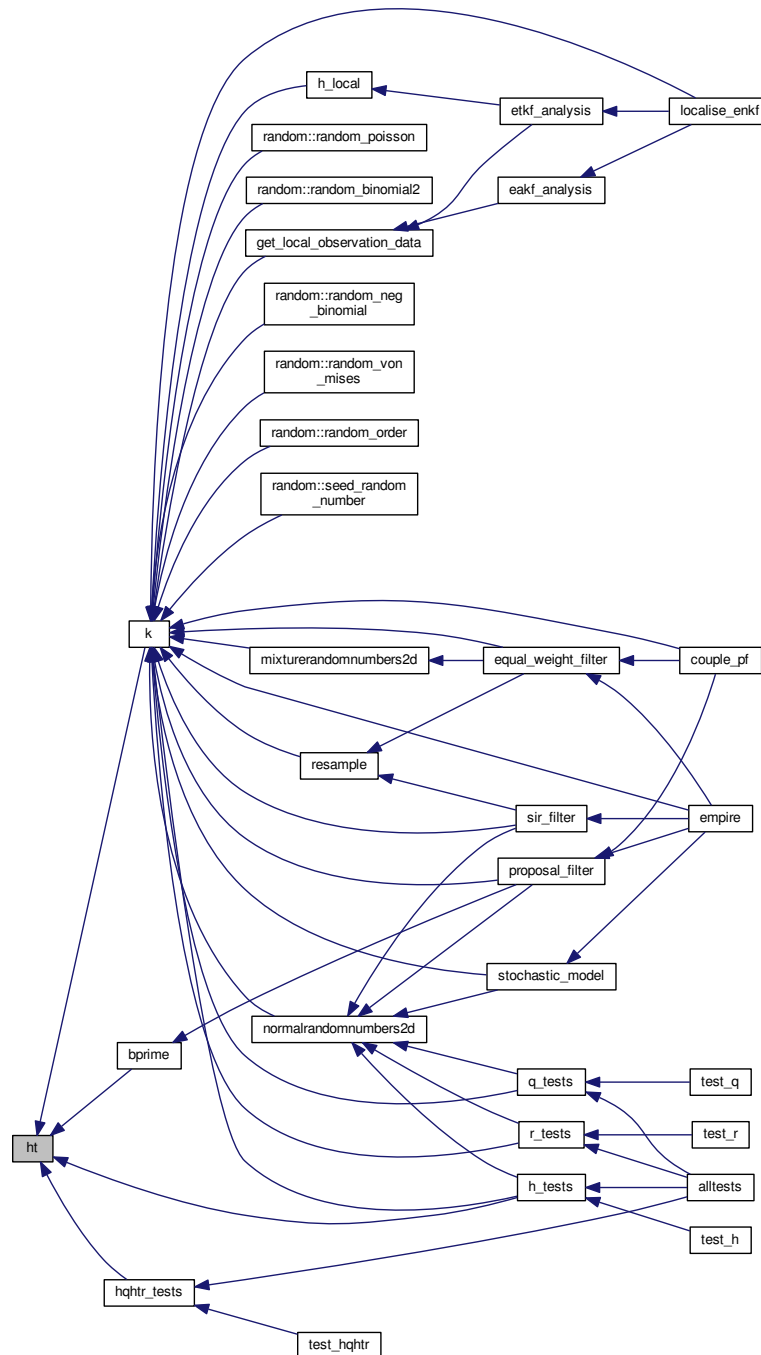
6.1.1.2 subroutine h ( real(kind=rk), dimension(state\_dim,pf%count), intent(in) x, real(kind=rk), dimension(obs\_dim,pf%count),  
intent(out) hx, integer, intent(in) t )

Here is the caller graph for this function:



6.1.1.3 subroutine ht ( real(kind=rk), dimension(obs\_dim,pf%count), intent(in) y, real(kind=rk), dimension(state\_dim,pf%count), intent(out) x, integer, intent(in) t )

Here is the caller graph for this function:



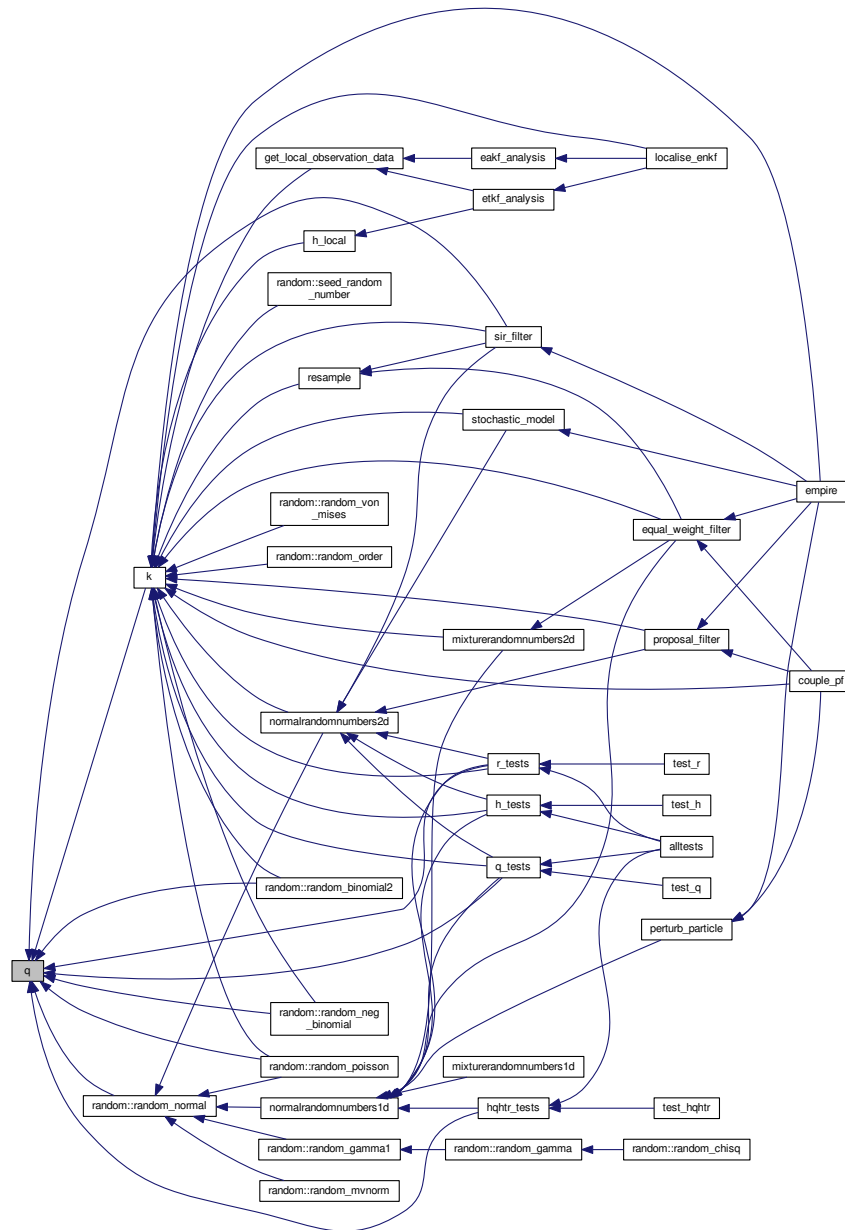


6.1.1.4 subroutine q ( integer, intent(in) *nrhs*, real(kind=rk), dimension(state\_dim,nrhs), intent(in) *x*, real(kind=rk), dimension(state\_dim,nrhs), intent(out) *Qx* )

Here is the call graph for this function:

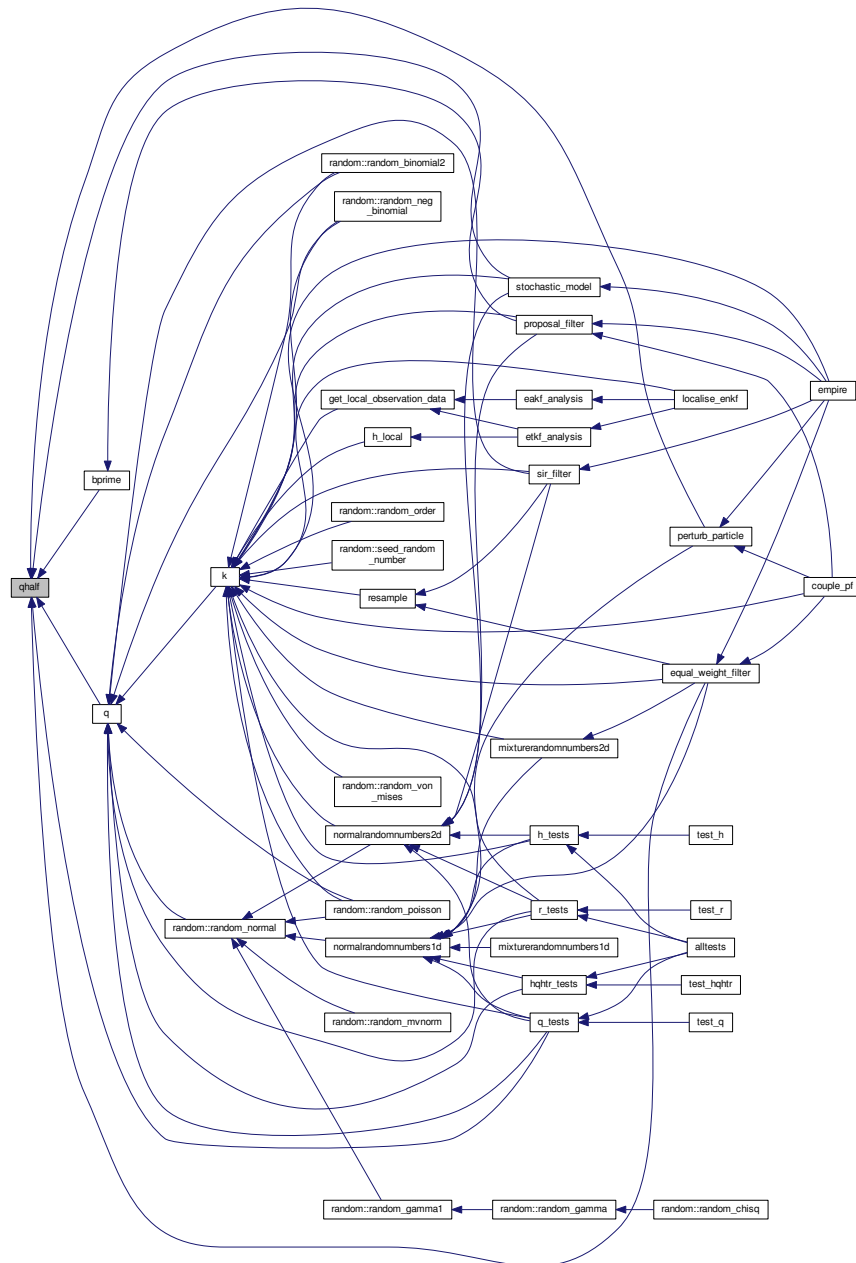


Here is the caller graph for this function:



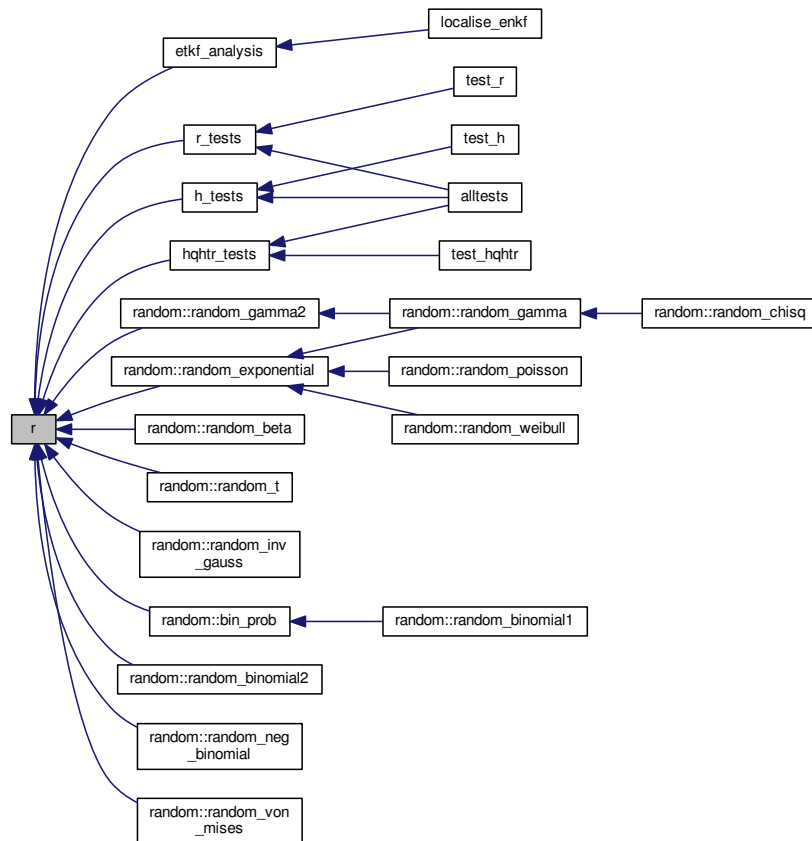
6.1.1.5 subroutine qhalf ( integer, intent(in) nrhs, real(kind=rk), dimension(state\_dim,nrhs), intent(in) x, real(kind=rk), dimension(state\_dim,nrhs), intent(out) Qx )

Here is the caller graph for this function:



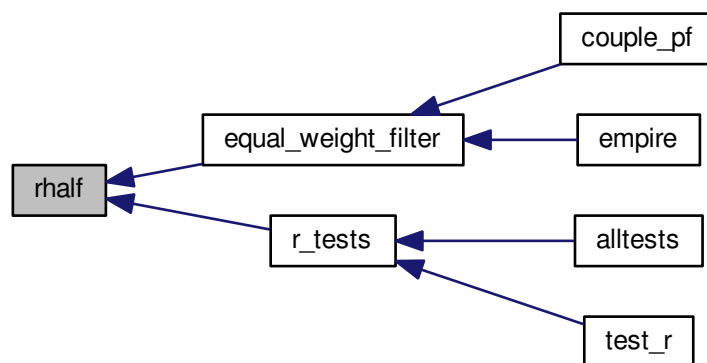
6.1.1.6 subroutine `r` ( integer, intent(in) *nrhs*, real(kind=rk), dimension(obs\_dim,nrhs), intent(in) *y*, real(kind=rk), dimension(obs\_dim,nrhs), intent(out) *Ry*, integer, intent(in) *t* )

Here is the caller graph for this function:



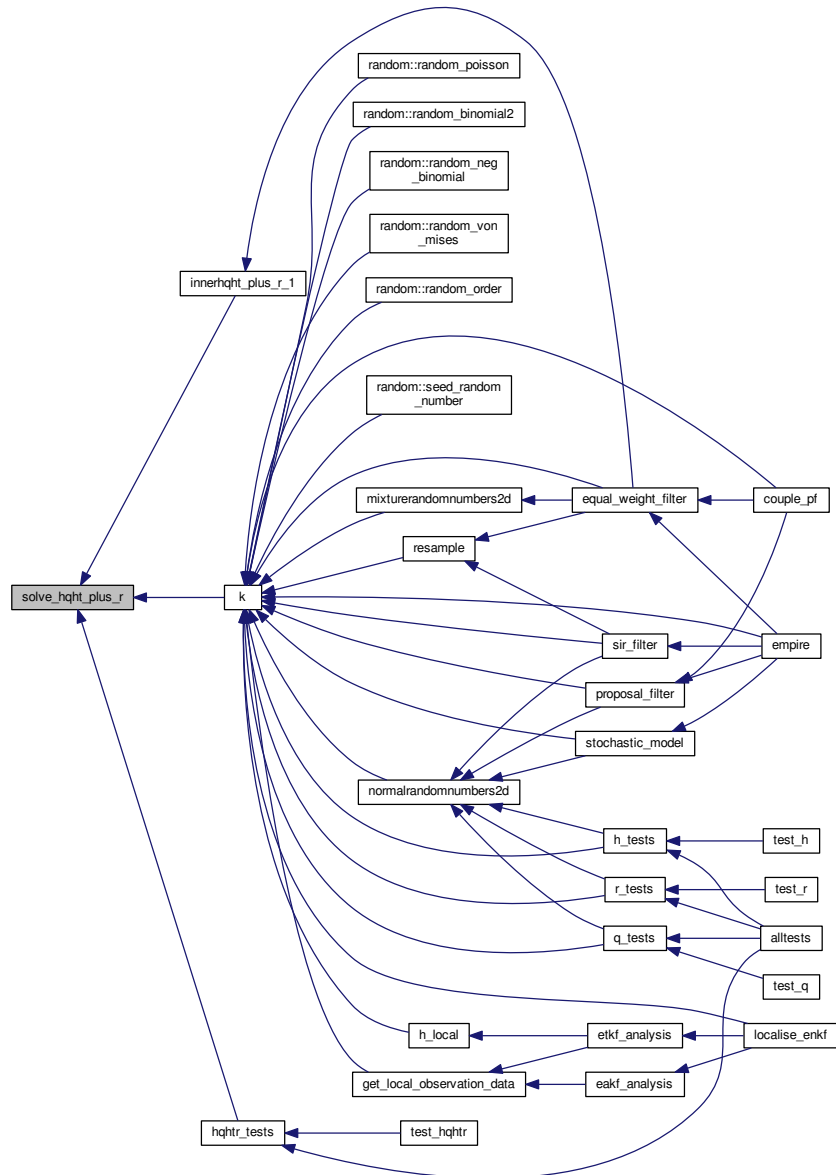
6.1.1.7 subroutine rhalf ( integer, intent(in) *nrhs*, real(kind=rk), dimension(obs\_dim,nrhs), intent(in) *y*, real(kind=rk), dimension(obs\_dim,nrhs), intent(out) *Ry*, integer, intent(in) *t* )

Here is the caller graph for this function:



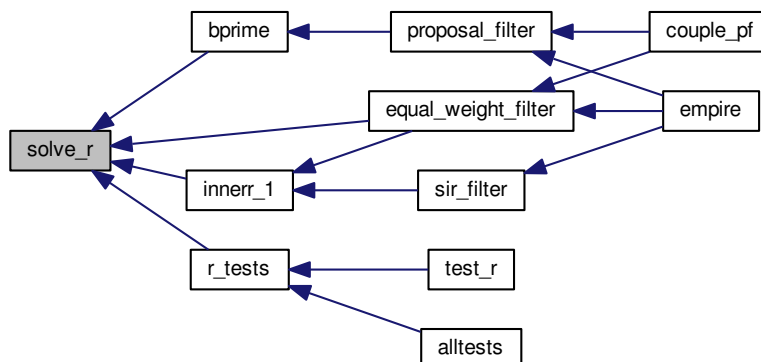
6.1.1.8 subroutine solve\_hqht\_plus\_r ( real(kind=rk), dimension(obs\_dim), intent(in) y, real(kind=rk), dimension(obs\_dim), intent(out) v, integer, intent(in) t )

Here is the caller graph for this function:



6.1.1.9 subroutine solve\_r ( real(kind=rk), dimension(obs\_dim,pf%count), intent(in) y, real(kind=rk),  
dimension(obs\_dim,pf%count), intent(out) v, integer, intent(in) t )

Here is the caller graph for this function:



## 6.2 src/controllers/old\_pf\_couple.f90 File Reference

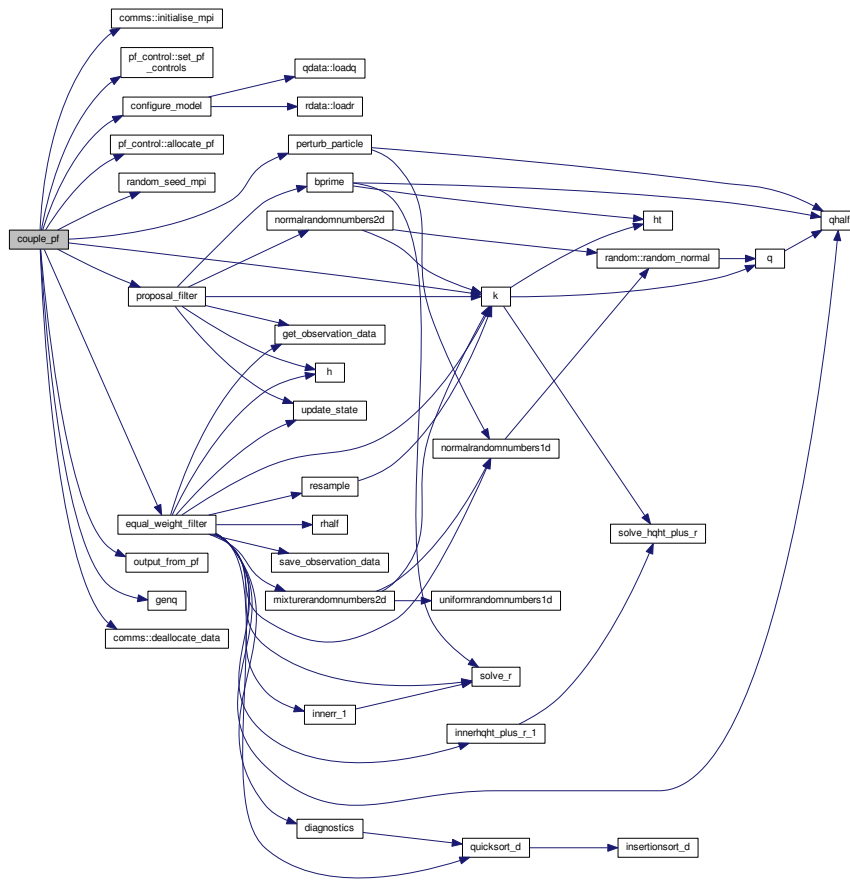
### Functions/Subroutines

- program [couple\\_pf](#)

### 6.2.1 Function/Subroutine Documentation

### 6.2.1.1 program couple\_pf ( )

Here is the call graph for this function:



## 6.3 src/controllers/pf\_control.f90 File Reference

### Data Types

- module [pf\\_control](#)  
module to hold all the information to control the the main program
- type [pf\\_control::pf\\_control\\_type](#)

## 6.4 src/controllers/pf\_couple.f90 File Reference

### Functions/Subroutines

- program [empire](#)  
the main program

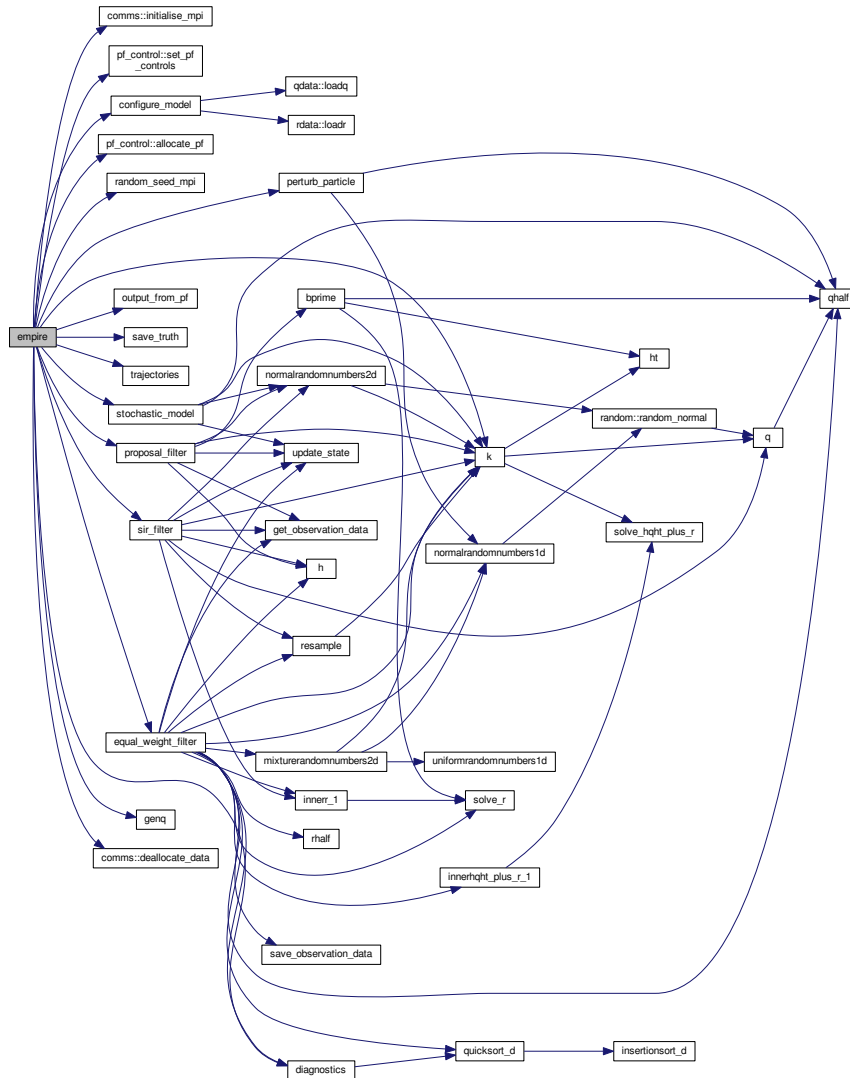
### 6.4.1 Function/Subroutine Documentation



## 6.4.1.1 program empire ( )

the main program

Here is the call graph for this function:



## 6.5 src/controllers/sizes.f90 File Reference

## Data Types

- module [sizes](#)

*Module that stores the dimension of observation and state spaces.*

## 6.6 src/data/Qdata.f90 File Reference

## Data Types

- module [qdata](#)

*Module as a place to store user specified data for  $Q$ .*

## 6.7 src/data/Rdata.f90 File Reference

### Data Types

- module [rdata](#)

*Module to hold user supplied data for  $R$  observation error covariance matrix.*

- module [hqht\\_plus\\_r](#)

## 6.8 src/DOC\_README.txt File Reference

## 6.9 src/filters/eakf\_analysis.f90 File Reference

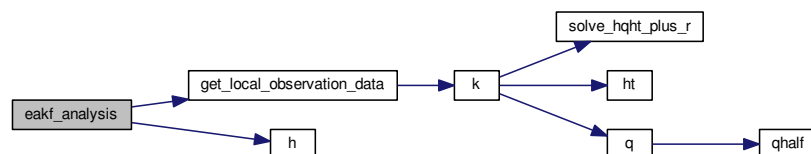
### Functions/Subroutines

- subroutine [eakf\\_analysis](#) (num\_hor, num\_ver, this\_hor, this\_ver, boundary, x, N, stateDim, obsDim, rho)

#### 6.9.1 Function/Subroutine Documentation

6.9.1.1 subroutine `eakf_analysis` ( integer, intent(in) *num\_hor*, integer, intent(in) *num\_ver*, integer, intent(in) *this\_hor*, integer, intent(in) *this\_ver*, integer, intent(in) *boundary*, real(kind=rk), dimension(statedim,n), intent(inout) *x*, integer, intent(in) *N*, integer, intent(in) *stateDim*, integer, intent(in) *obsDim*, real(kind=rk), intent(in) *rho* )

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.10 src/filters/enkf\_specific.f90 File Reference

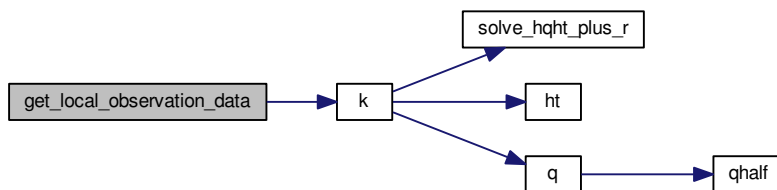
### Functions/Subroutines

- subroutine [h\\_local](#) (num\_hor, num\_ver, this\_hor, this\_ver, boundary, nrhs, stateDim, x, obsDim, y)
- subroutine [solve\\_rhalf\\_local](#) (num\_hor, num\_ver, this\_hor, this\_ver, boundary, nrhs, obsDim, y, v)
- subroutine [get\\_local\\_observation\\_data](#) (num\_hor, num\_ver, this\_hor, this\_ver, boundary, obsDim, y)
- subroutine [localise\\_enkf](#) (enkf\_analysis)

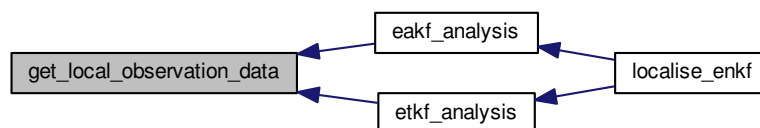
### 6.10.1 Function/Subroutine Documentation

6.10.1.1 subroutine `get_local_observation_data` ( integer, intent(in) *num\_hor*, integer, intent(in) *num\_ver*, integer, intent(in) *this\_hor*, integer, intent(in) *this\_ver*, integer, intent(in) *boundary*, integer, intent(in) *obsDim*, real(kind=rk), dimension(obsdim), intent(out) *y* )

Here is the call graph for this function:

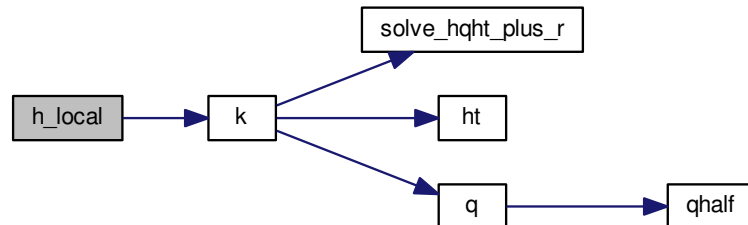


Here is the caller graph for this function:

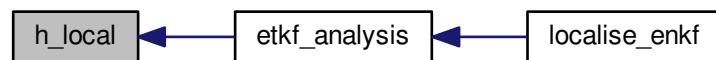


6.10.1.2 subroutine `h_local` ( integer, intent(in) *num\_hor*, integer, intent(in) *num\_ver*, integer, intent(in) *this\_hor*, integer, intent(in) *this\_ver*, integer, intent(in) *boundary*, integer, intent(in) *nrhs*, integer, intent(in) *stateDim*, real(kind=rk), dimension(statedim,nrhs), intent(in) *x*, integer, intent(in) *obsDim*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *y* )

Here is the call graph for this function:

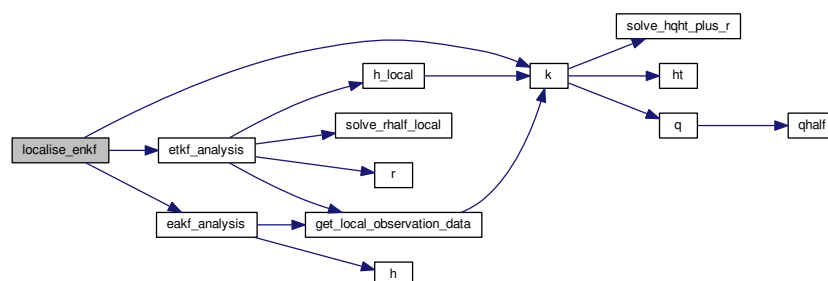


Here is the caller graph for this function:



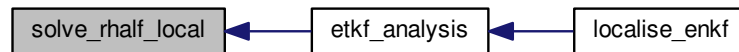
6.10.1.3 subroutine `localise_enkf` ( integer, intent(in) *enkf\_analysis* )

Here is the call graph for this function:



6.10.1.4 subroutine solve\_rhalf\_local ( integer, intent(in) num\_hor, integer, intent(in) num\_ver, integer, intent(in) this\_hor, integer, intent(in) this\_ver, integer, intent(in) boundary, integer, intent(in) nrhs, integer, intent(in) obsDim, real(kind=rk), dimension(obsdim,nrhs), intent(in) y, real(kind=rk), dimension(obsdim,nrhs), intent(out) v )

Here is the caller graph for this function:



## 6.11 src/filters/equivalent\_weights\_step.f90 File Reference

### Functions/Subroutines

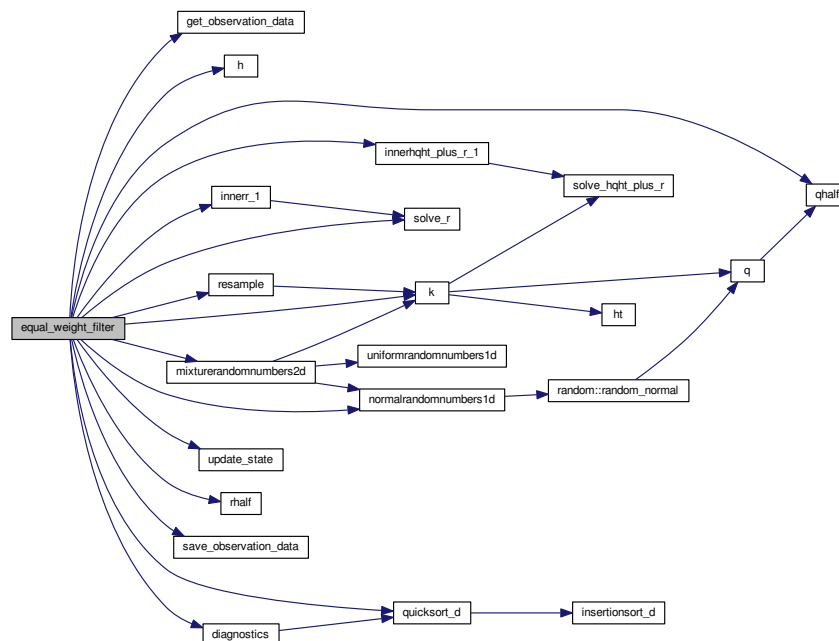
- subroutine [equal\\_weight\\_filter](#)  
subroutine to do the equivalent weights step

#### 6.11.1 Function/Subroutine Documentation

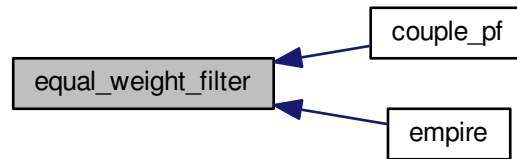
##### 6.11.1.1 subroutine equal\_weight\_filter ( )

subroutine to do the equivalent weights step

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.12 src/filters/etkf\_analysis.f90 File Reference

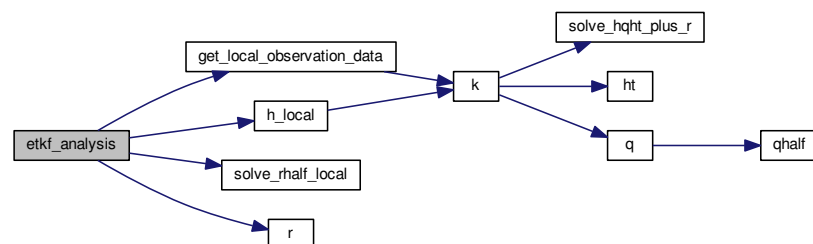
### Functions/Subroutines

- subroutine [etkf\\_analysis](#) (num\_hor, num\_ver, this\_hor, this\_ver, boundary, x, N, stateDim, obsDim, rho)

### 6.12.1 Function/Subroutine Documentation

6.12.1.1 subroutine `etkf_analysis` ( integer, intent(in) *num\_hor*, integer, intent(in) *num\_ver*, integer, intent(in) *this\_hor*, integer, intent(in) *this\_ver*, integer, intent(in) *boundary*, real(kind=rk), dimension(statedim,n), intent(inout) *x*, integer, intent(in) *N*, integer, intent(in) *stateDim*, integer, intent(in) *obsDim*, real(kind=rk), intent(in) *rho* )

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.13 src/filters/proposal\_filter.f90 File Reference

### Functions/Subroutines

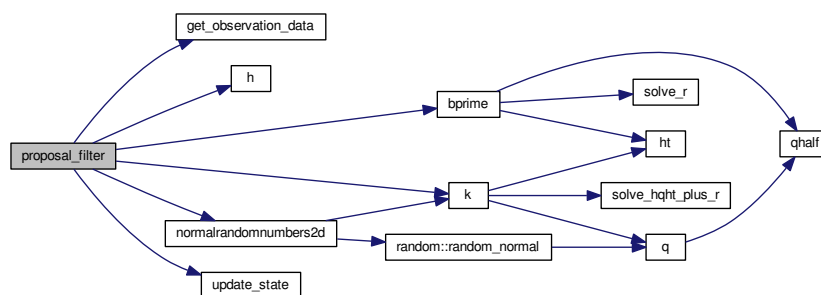
- subroutine [proposal\\_filter](#)  
*Subroutine to perform nudging in the proposal step of EWPF.*

### 6.13.1 Function/Subroutine Documentation

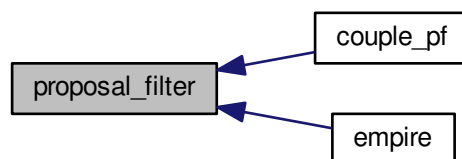
#### 6.13.1.1 subroutine proposal\_filter ( )

Subroutine to perform nudging in the proposal step of EWPF.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.14 src/filters/sir\_filter.f90 File Reference

### Functions/Subroutines

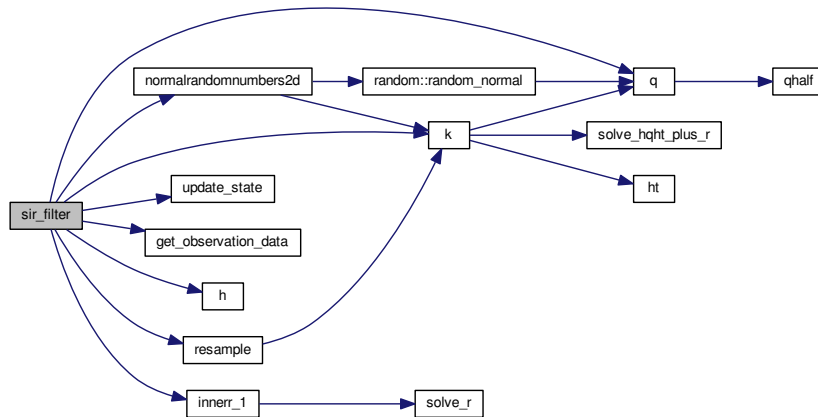
- subroutine [sir\\_filter](#)  
*Subroutine to perform SIR filter (Sequential Importance Resampling)*

### 6.14.1 Function/Subroutine Documentation

#### 6.14.1.1 subroutine sir\_filter ( )

Subroutine to perform SIR filter (Sequential Importance Resampling)

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.15 src/filters/stochastic\_model.f90 File Reference

### Functions/Subroutines

- subroutine [stochastic\\_model](#)  
*subroutine to simply move the model forward in time one timestep PAB 21-05-2013*
- subroutine [check\\_scaling](#) (x, fx, b, scales)

### 6.15.1 Function/Subroutine Documentation

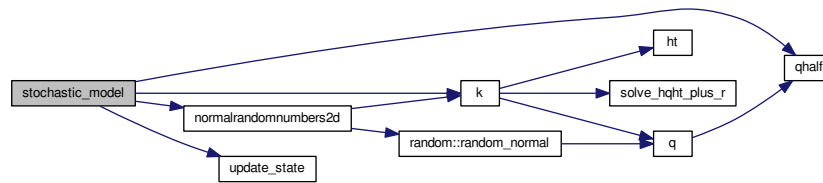
6.15.1.1 subroutine [check\\_scaling](#) ( real(kind=rk), dimension(state\_dim), intent(in) x, real(kind=rk), dimension(state\_dim), intent(in) fx, real(kind=rk), dimension(state\_dim), intent(in) b, real(kind=rk), dimension(9), intent(inout) scales )

6.15.1.2 subroutine [stochastic\\_model](#) ( )

subroutine to simply move the model forward in time one timestep PAB 21-05-2013



Here is the call graph for this function:



Here is the caller graph for this function:



## 6.16 src/operations/gen\_rand.f90 File Reference

### Functions/Subroutines

- subroutine [uniformrandomnumbers1d](#) (minv, maxv, n, phi)  
*generate one dimension of uniform random numbers*
- subroutine [normalrandomnumbers1d](#) (mean, stdev, n, phi)  
*generate one dimension of Normal random numbers*
- subroutine [normalrandomnumbers2d](#) (mean, stdev, n, k, phi)  
*generate two dimensional Normal random numbers*
- subroutine [mixturerandomnumbers1d](#) (mean, stdev, ufac, epsi, n, phi, uniform)  
*generate one dimensional vector drawn from mixture density*
- subroutine [mixturerandomnumbers2d](#) (mean, stdev, ufac, epsi, n, k, phi, uniform)  
*generate two dimensional vector, each drawn from mixture density*
- subroutine [random\\_seed\\_mpi](#) (pfid)  
*Subroutine to set the random seed across MPI threads.*

### 6.16.1 Function/Subroutine Documentation

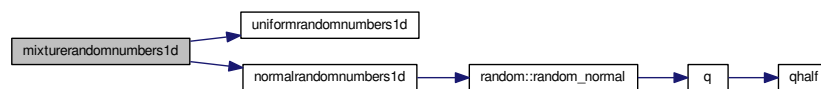
- 6.16.1.1 subroutine [mixturerandomnumbers1d](#) ( real(kind=kind(1.0d0)), intent(in) *mean*, real(kind=kind(1.0d0)), intent(in) *stdev*, real(kind=kind(1.0d0)), intent(in) *ufac*, real(kind=kind(1.0d0)), intent(in) *epsi*, integer, intent(in) *n*, real(kind=kind(1.0d0)), dimension(n), intent(out) *phi*, logical, intent(out) *uniform* )

generate one dimensional vector drawn from mixture density

## Parameters

in	<i>mean</i>	Mean of normal distribution
in	<i>stdev</i>	Standard deviation of normal distribution
in	<i>ufac</i>	half-width of uniform distribution that is centered on the mean
in	<i>epsi</i>	Proportion controlling mixture draw. if random_number > epsi then draw from uniform, else normal
in	<i>n</i>	size of output vector
out	<i>phi</i>	n dimensional mixture random numbers
out	<i>uniform</i>	True if mixture drawn from uniform. False if drawn from normal

Here is the call graph for this function:



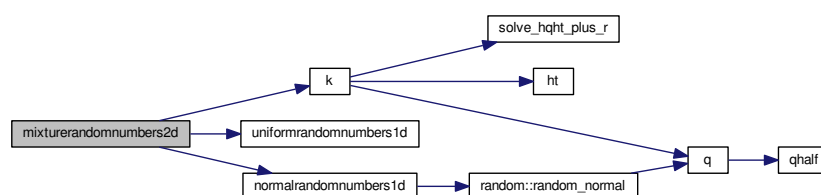
6.16.1.2 subroutine mixturerandomnumbers2d ( real(kind=kind(1.0d0)), intent(in) *mean*, real(kind=kind(1.0d0)), intent(in) *stdev*, real(kind=kind(1.0d0)), intent(in) *ufac*, real(kind=kind(1.0d0)), intent(in) *epsi*, integer, intent(in) *n*, integer, intent(in) *k*, real(kind=kind(1.0d0)), dimension(n,k), intent(out) *phi*, logical, dimension(k), intent(out) *uniform* )

generate two dimensional vector, each drawn from mixture density

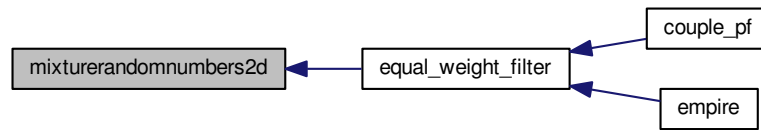
## Parameters

in	<i>mean</i>	Mean of normal distribution
in	<i>stdev</i>	Standard deviation of normal distribution
in	<i>ufac</i>	half-width of uniform distribution that is centered on the mean
in	<i>epsi</i>	Proportion controlling mixture draw. if random_number > epsi then draw from uniform, else normal
in	<i>n</i>	first dimension of output vector
in	<i>n</i>	second dimension of output vector
out	<i>phi</i>	n,k dimensional mixture random numbers
out	<i>uniform</i>	k dimensional logical with uniform(i) True if phi(:,i) drawn from uniform. False if drawn from normal

Here is the call graph for this function:



Here is the caller graph for this function:



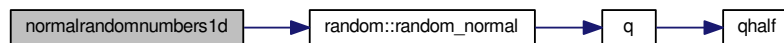
6.16.1.3 subroutine `normalrandomnumbers1d` ( `real(kind=rk), intent(in) mean`, `real(kind=rk), intent(in) stdev`, `integer, intent(in) n`, `real(kind=rk), dimension(n), intent(out) phi` )

generate one dimension of Normal random numbers

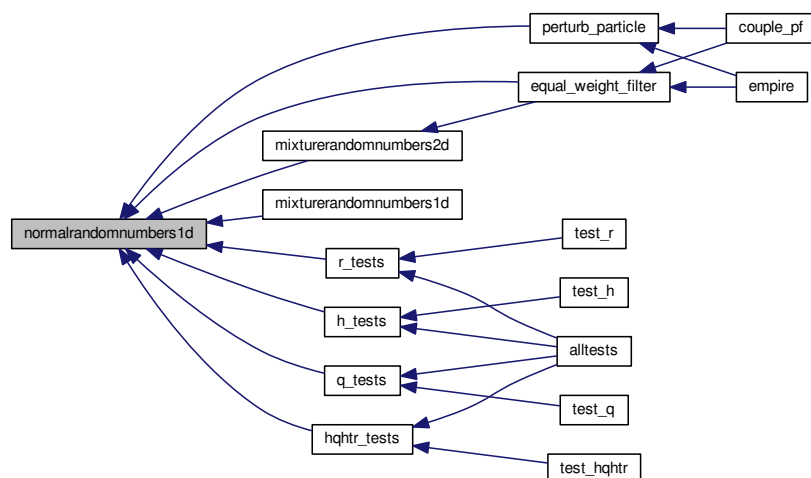
#### Parameters

in	<i>n</i>	size of output vector
in	<i>mean</i>	mean of normal distribution
in	<i>stdev</i>	Standard Deviation of normal distribution
out	<i>phi</i>	n dimensional normal random numbers

Here is the call graph for this function:



Here is the caller graph for this function:



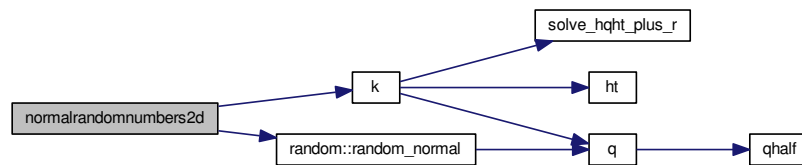
6.16.1.4 subroutine `normalrandomnumbers2d` ( `real(kind=rk), intent(in) mean`, `real(kind=rk), intent(in) stdev`, `integer, intent(in) n`, `integer, intent(in) k`, `real(kind=rk), dimension(n,k), intent(out) phi` )

generate two dimensional Normal random numbers

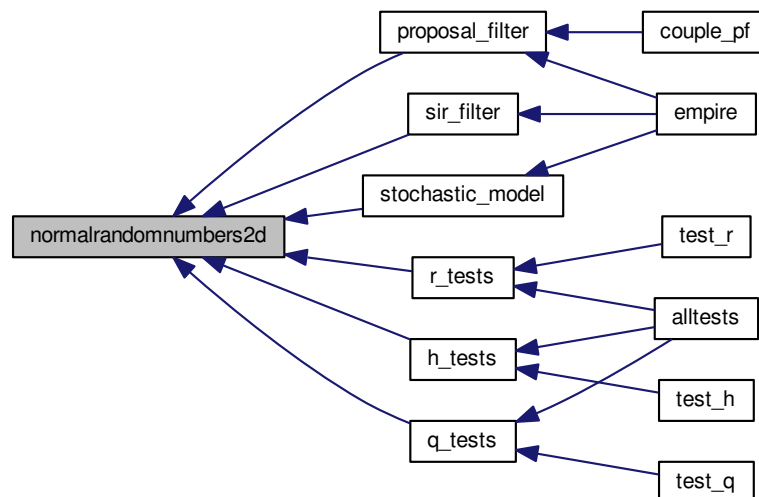
#### Parameters

in	<i>n</i>	first dimension of output vector
in	<i>n</i>	second dimension of output vector
in	<i>mean</i>	mean of normal distribution
in	<i>stdev</i>	Standard Deviation of normal distribution
out	<i>phi</i>	n,k dimensional normal random numbers

Here is the call graph for this function:



Here is the caller graph for this function:



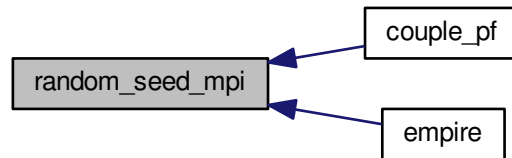
6.16.1.5 subroutine `random_seed_mpi` ( `integer, intent(in) pfid` )

Subroutine to set the random seed across MPI threads.

## Parameters

in	<i>pfid</i>	The process identifier of the MPI process
----	-------------	---

Here is the caller graph for this function:



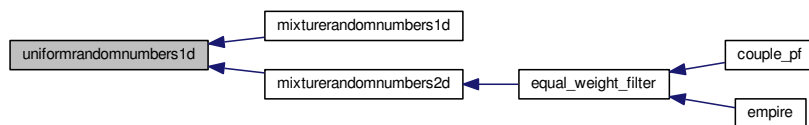
6.16.1.6 subroutine `uniformrandomnumbers1d` ( `real(kind=rk)`, intent(in) *minv*, `real(kind=rk)`, intent(in) *maxv*, integer, intent(in) *n*, `real(kind=rk)`, dimension(*n*), intent(out) *phi* )

generate one dimension of uniform random numbers

## Parameters

in	<i>n</i>	size of output vector
in	<i>minv</i>	minimum value of uniform distribution
in	<i>maxv</i>	maximum value of uniform distribution
out	<i>phi</i>	n dimensional uniform random numbers

Here is the caller graph for this function:



## 6.17 src/operations/operator\_wrappers.f90 File Reference

## Functions/Subroutines

- subroutine `k` (*y*, *x*)  
Subroutine to apply  $K$  to a vector  $y$  in observation space where  $K := QH^T(QH^T + R)^{-1}$ .
- subroutine `innerr_1` (*y*, *w*)  
subroutine to compute the inner product with  $R^{-1}$
- subroutine `innerhqht_plus_r_1` (*y*, *w*)  
subroutine to compute the inner product with  $(HQH^T + R)^{-1}$
- subroutine `bprime` (*y*, *x*, `QHtR_1y`, `normaln`, *betan*)  
subroutine to calculate nudging term and correlated random errors efficiently

### 6.17.1 Function/Subroutine Documentation

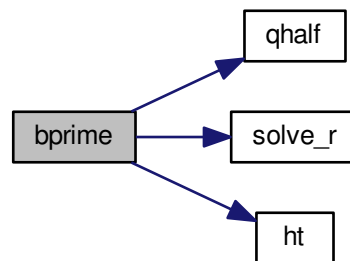
6.17.1.1 subroutine `bprime` ( `real(kind=rk)`, `dimension(obs_dim,pf%count)`, `intent(in)` `y`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out)` `x`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out)` `QHtR_1y`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(in)` `normaln`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out)` `betan` )

subroutine to calculate nudging term and correlated random errors efficiently

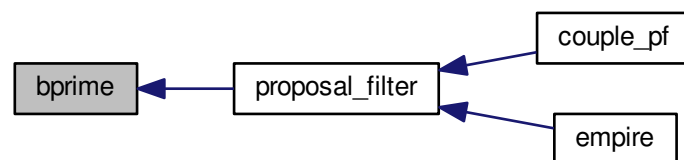
#### Parameters

in	<code>y</code>	( <code>obs_dim,pf%count</code> ) vectors of innovations $y - H(x^{n-1})$
out	<code>x</code>	( <code>state_dim,pf%count</code> ) vectors of $pQ^{\frac{1}{2}}H^TR^{-1}[y - H(x^{n-1})]$
out	<code>QHtR_1y</code>	( <code>state_dim,pf%count</code> ) vectors of $pQH^TR^{-1}[y - H(x^{n-1})]$
in	<code>normaln</code>	( <code>state_dim,pf%count</code> ) uncorrelated random vectors such that $\text{normaln}(:,i) \sim \mathcal{N}(0, I)$
out	<code>betan</code>	( <code>state_dim,pf%count</code> ) correlated random vectors such that $\text{betan}(:,i) \sim \mathcal{N}(0, Q)$

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.1.2 subroutine `innerhqht_plus_r_1` ( `real(kind=rk)`, `dimension(obs_dim)`, `intent(in)` `y`, `real(kind=rk)`, `intent(out)` `w` )

subroutine to compute the inner product with  $(HQH^T + R)^{-1}$

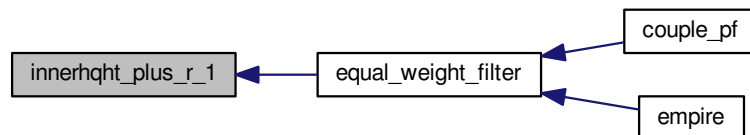
## Parameters

in	$y$	vector in observation space
out	$w$	scalar with value $y^T R^{-1} y$

Here is the call graph for this function:



Here is the caller graph for this function:



**6.17.1.3** subroutine `innerr_1` ( `real(kind=rk)`, `dimension(obs_dim,pf%count)`, `intent(in) y`, `real(kind=rk)`, `dimension(pf%count)`, `intent(out) w` )

subroutine to compute the inner product with  $R^{-1}$

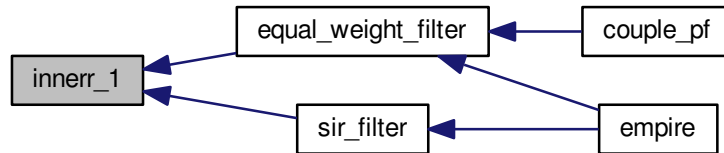
## Parameters

in	$y$	multiple vectors in observation space (pf%count of them)
out	$w$	multiple scalars (pf%count) where $w(i)$ has the value $y(:,i)^T R^{-1} y(:,i)$

Here is the call graph for this function:



Here is the caller graph for this function:



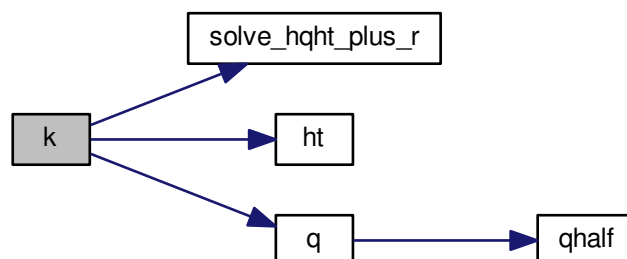
6.17.1.4 subroutine `k` ( `real(kind=rk)`, `dimension(obs_dim,pf%count)`, `intent(in) y`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out) x` )

Subroutine to apply  $K$  to a vector  $y$  in observation space where  $K := QH^T(HQH^T + R)^{-1}$ .

Parameters

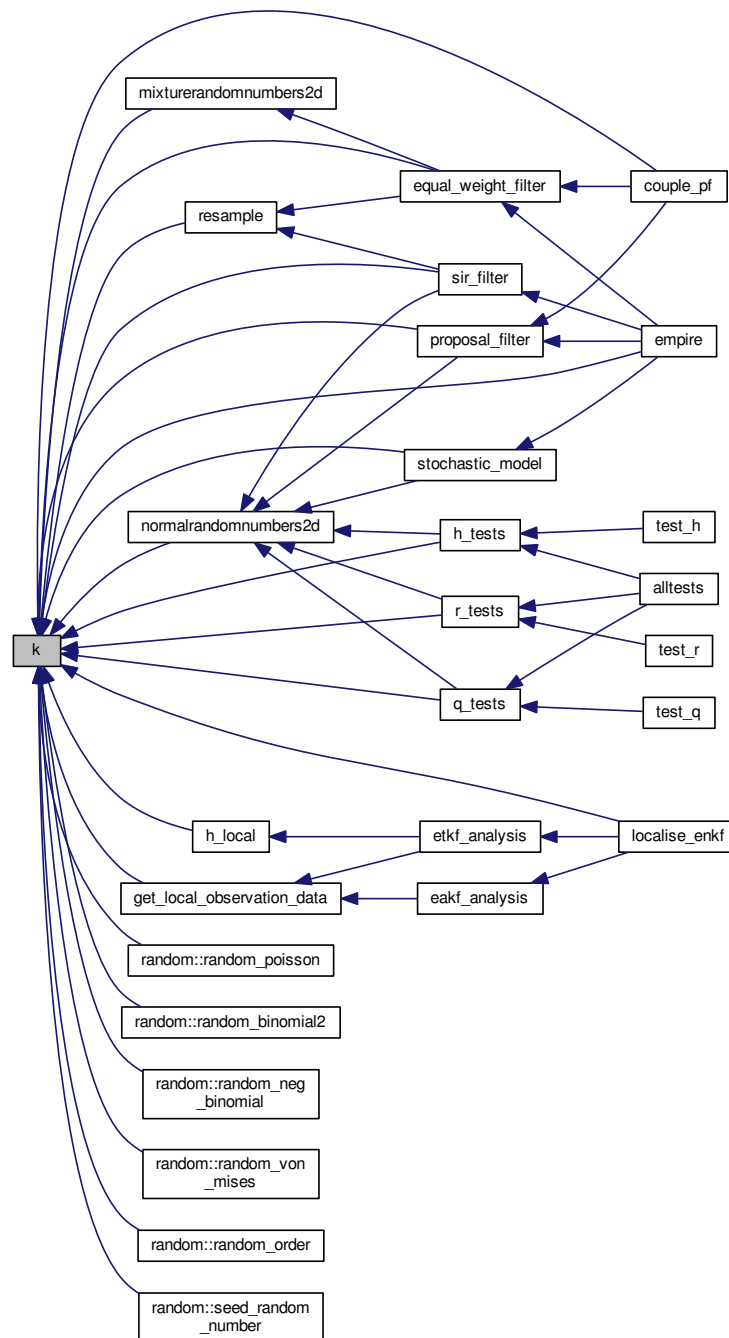
<code>in</code>	<code>y</code>	vector in observation space
<code>out</code>	<code>x</code>	vector in state space

Here is the call graph for this function:





Here is the caller graph for this function:



## 6.18 src/operations/perturb\_particle.f90 File Reference

### Functions/Subroutines

- subroutine `perturb_particle` (x)

Subroutine to perturb state vector with normal random vector drawn from  $\mathcal{N}(0, Q)$ .

- subroutine `update_state` (state, fps, kgain, betan)

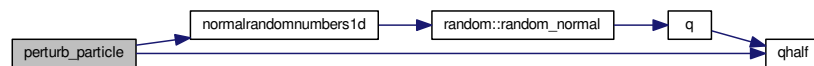
Subroutine to update the state.

## 6.18.1 Function/Subroutine Documentation

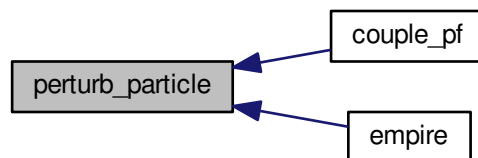
### 6.18.1.1 subroutine `perturb_particle` ( real(kind=rk), dimension(state\_dim), intent(inout) x )

Subroutine to perturb state vector with normal random vector drawn from  $\mathcal{N}(0, Q)$ .

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.18.1.2 subroutine `update_state` ( real(kind=rk), dimension(state\_dim), intent(out) state, real(kind=rk), dimension(state\_dim), intent(in) fps, real(kind=rk), dimension(state\_dim), intent(in) kgain, real(kind=rk), dimension(state\_dim), intent(inout) betan )

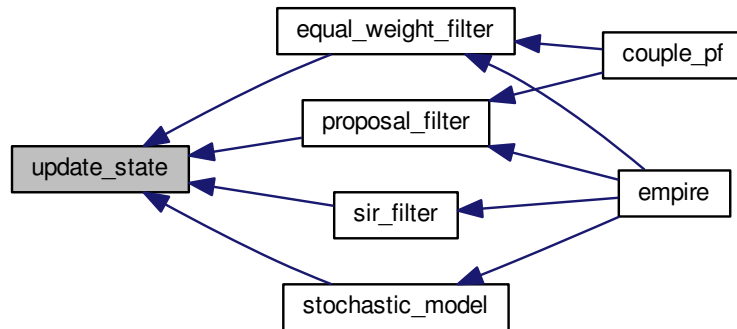
Subroutine to update the state.

This can be changed for the specific model if it needs to be

#### Parameters

in	<i>fps</i>	deterministic model update $f(x^{n-1})$
in	<i>kgain</i>	nudging term
in, out	<i>betan</i>	Stochastic term
out	<i>state</i>	The updated state vector

Here is the caller graph for this function:



## 6.19 src/operations/resample.f90 File Reference

### Functions/Subroutines

- subroutine [resample](#)

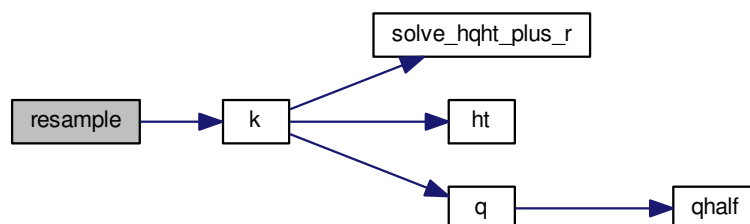
*Subroutine to perform Universal Importance Resampling.*

### 6.19.1 Function/Subroutine Documentation

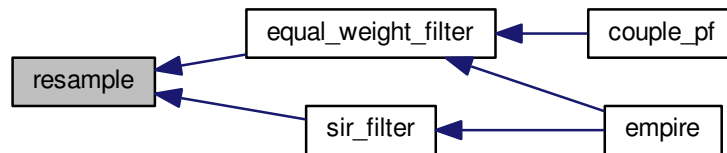
#### 6.19.1.1 subroutine resample ( )

Subroutine to perform Universal Importance Resampling.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.20 src/tests/alltests.f90 File Reference

### Functions/Subroutines

- program [alltests](#)

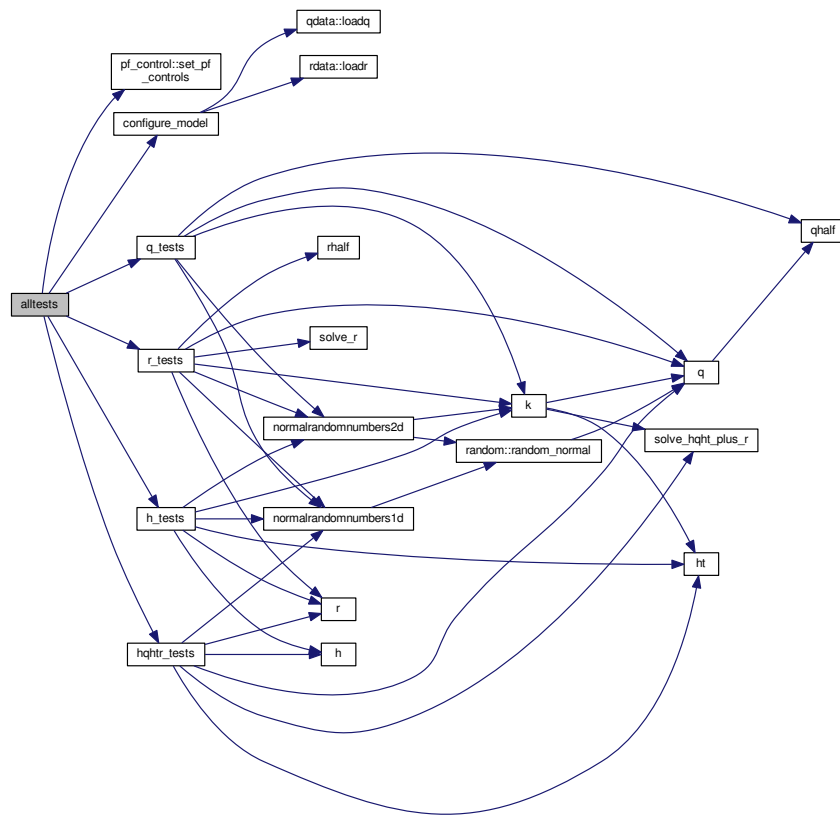
*program to run all tests of user specific functions*

### 6.20.1 Function/Subroutine Documentation

#### 6.20.1.1 program alltests ( )

program to run all tests of user specific functions

Here is the call graph for this function:



## 6.21 src/tests/test\_h.f90 File Reference

### Functions/Subroutines

- program [test\\_h](#)

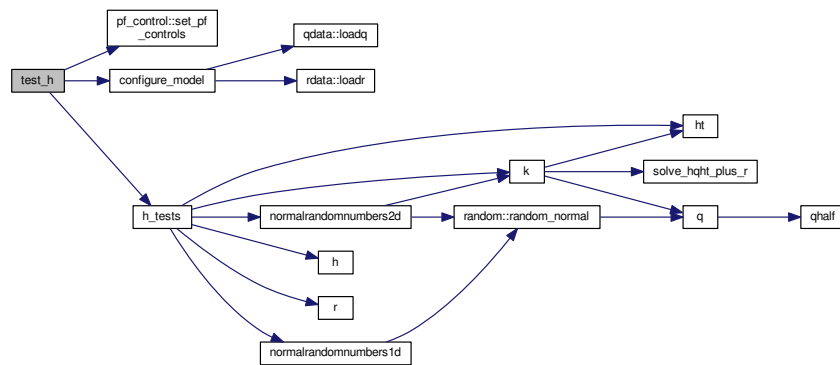
*program to run tests of user supplied observation operator*

#### 6.21.1 Function/Subroutine Documentation

##### 6.21.1.1 program test\_h ( )

program to run tests of user supplied observation operator

Here is the call graph for this function:



## 6.22 src/tests/test\_hqht.f90 File Reference

### Functions/Subroutines

- program [test\\_hqht](#)  
*program to run tests of user supplied linear solve*

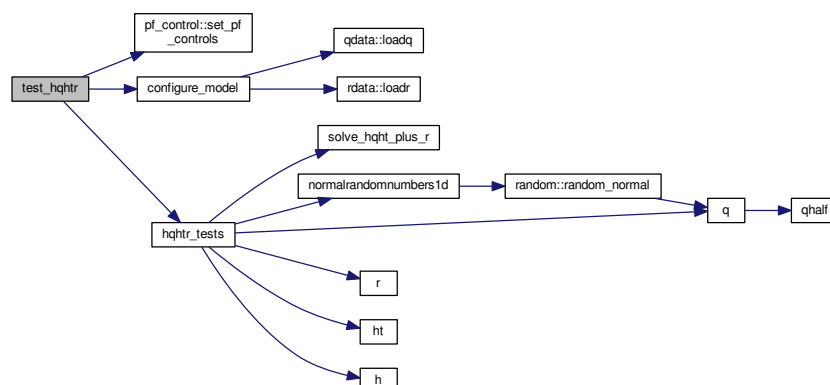
### 6.22.1 Function/Subroutine Documentation

#### 6.22.1.1 program test\_hqht ( )

program to run tests of user supplied linear solve

$$(HQH^T + R)^{-1}$$

Here is the call graph for this function:



## 6.23 src/tests/test\_q.f90 File Reference

## Functions/Subroutines

- program [test\\_q](#)

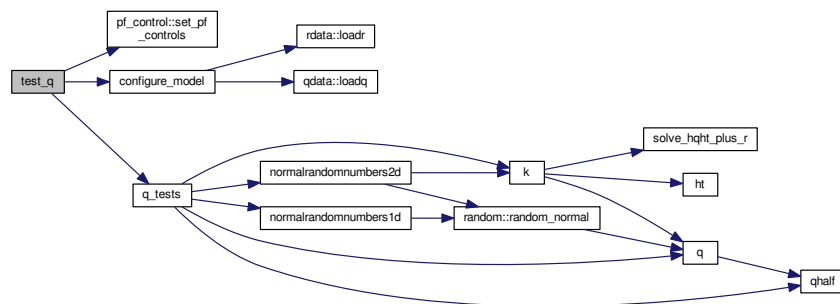
*program to run tests of user supplied model error covariance matrix*

### 6.23.1 Function/Subroutine Documentation

#### 6.23.1.1 program test\_q ( )

program to run tests of user supplied model error covariance matrix

Here is the call graph for this function:



## 6.24 src/tests/test\_r.f90 File Reference

## Functions/Subroutines

- program [test\\_r](#)

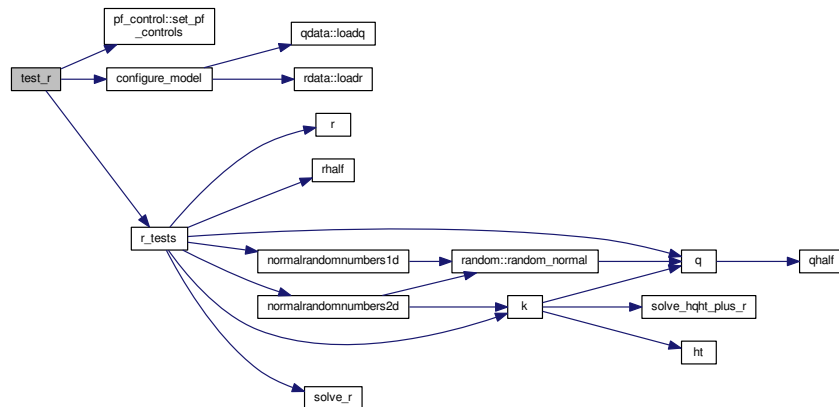
*program to run all tests of user supplied observation error covariance matrix/*

### 6.24.1 Function/Subroutine Documentation

#### 6.24.1.1 program test\_r ( )

program to run all tests of user supplied observation error covariance matrix/

Here is the call graph for this function:



## 6.25 src/tests/tests.f90 File Reference

### Functions/Subroutines

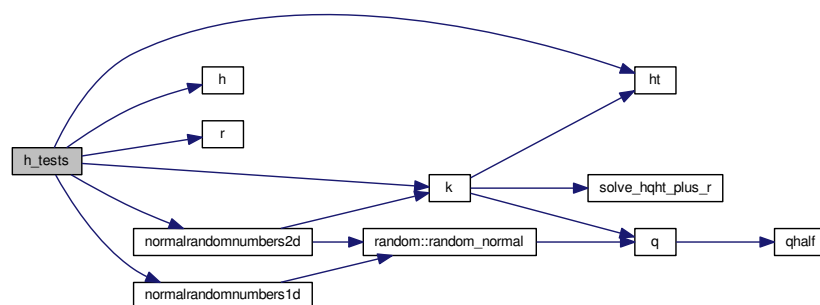
- subroutine [h\\_tests](#) ()
- subroutine [r\\_tests](#) ()
- subroutine [q\\_tests](#) ()
- subroutine [hqhtr\\_tests](#) ()

### 6.25.1 Function/Subroutine Documentation

#### 6.25.1.1 subroutine [h\\_tests](#) ( )

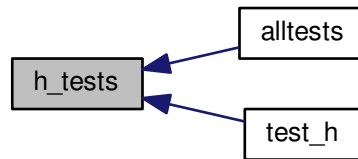
These are some tests to check that the observation operator is implemented correctly

Here is the call graph for this function:





Here is the caller graph for this function:

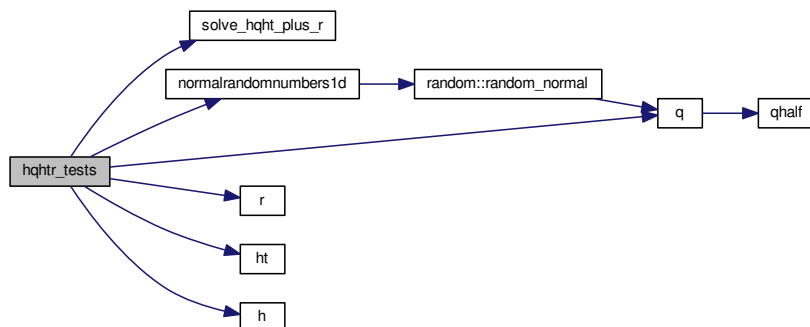


#### 6.25.1.2 subroutine hqhtr\_tests ( )

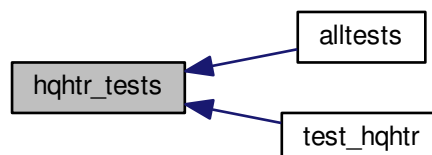
These are some tests to check that the linear solve operator is implemented correctly

This should check the operation  $(HQH^T + R)^{-1}$  is working

Here is the call graph for this function:



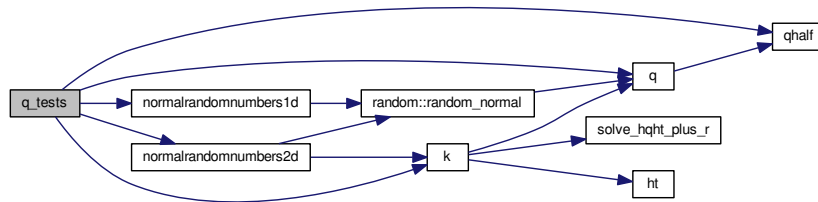
Here is the caller graph for this function:



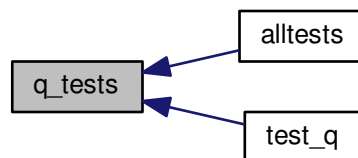
### 6.25.1.3 subroutine q\_tests ( )

These are some tests to check that the model error covariance matrix is implemented correctly

Here is the call graph for this function:



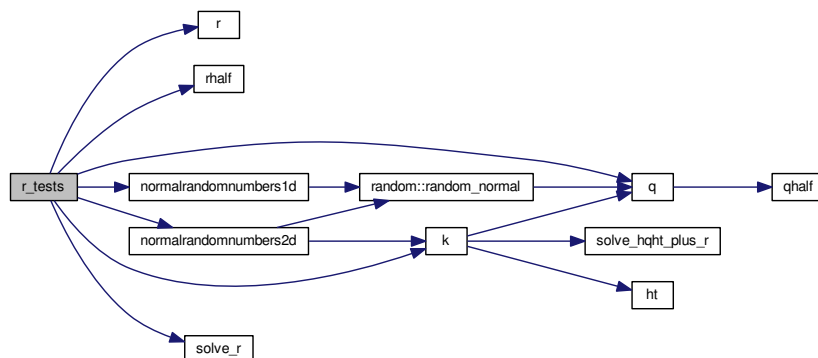
Here is the caller graph for this function:



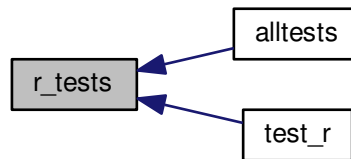
### 6.25.1.4 subroutine r\_tests ( )

These are some tests to check that the observation error covariance matrix is implemented correctly

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.26 src/utls/comms.f90 File Reference

### Data Types

- module `comms`  
*Module containing EMPIRE coupling data.*

## 6.27 src/utls/data\_io.f90 File Reference

### Functions/Subroutines

- subroutine `get_observation_data` (y)  
*Subroutine to read observation from a file  
Uses pftimestep to determine which observation to read.*
- subroutine `save_observation_data` (y)  
*Subroutine to save observation to a file  
Uses pftimestep to determine which observation to save.*
- subroutine `save_truth` (x)  
*Subroutine to save truth to a file*
- subroutine `output_from_pf`  
*subroutine to ouput data from the filter*

### 6.27.1 Function/Subroutine Documentation

#### 6.27.1.1 subroutine `get_observation_data` ( real(kind=rk), dimension(obs\_dim), intent(out) y )

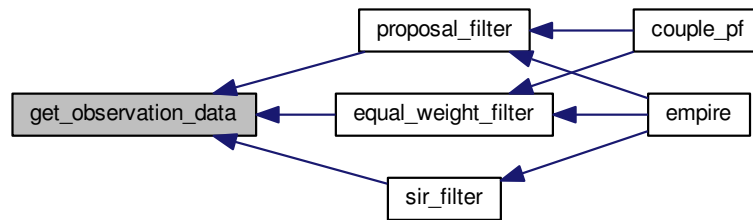
Subroutine to read observation from a file

Uses pftimestep to determine which observation to read.

#### Parameters

out	y	The observation
-----	---	-----------------

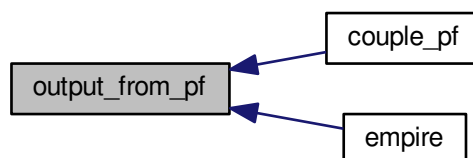
Here is the caller graph for this function:



#### 6.27.1.2 subroutine output\_from\_pf ( )

subroutine to output data from the filter

Here is the caller graph for this function:



#### 6.27.1.3 subroutine save\_observation\_data ( real(kind=rk), dimension(obs\_dim), intent(in) y )

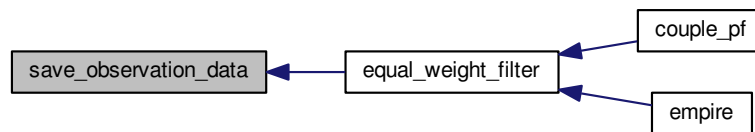
Subroutine to save observation to a file

Uses `pftimestep` to determine which observation to save.

**Parameters**

<code>in</code>	<code>y</code>	The observation
-----------------	----------------	-----------------

Here is the caller graph for this function:



#### 6.27.1.4 subroutine `save_truth` ( `real(kind=rk)`, `dimension(state_dim)`, `intent(in) x` )

Subroutine to save truth to a file

.

##### Parameters

<code>in</code>	<code>x</code>	The state vector
-----------------	----------------	------------------

Here is the caller graph for this function:



## 6.28 src/utls/diagnostics.f90 File Reference

### Functions/Subroutines

- subroutine [diagnostics](#)

*Subroutine to give output diagnostics such as rank histograms and trajectories.*

- subroutine [trajectories](#)

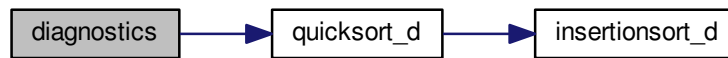
*subroutine to output trajectories*

### 6.28.1 Function/Subroutine Documentation

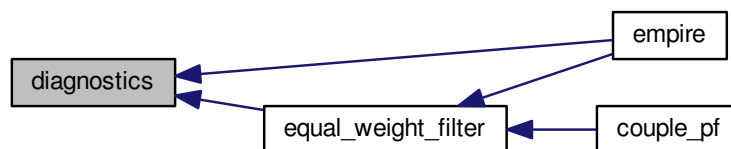
#### 6.28.1.1 subroutine `diagnostics` ( )

Subroutine to give output diagnostics such as rank histograms and trajectories.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.28.1.2 subroutine trajectories ( )

subroutine to output trajectories

Here is the caller graph for this function:



## 6.29 src/utls/genQ.f90 File Reference

### Functions/Subroutines

- subroutine [genq](#)

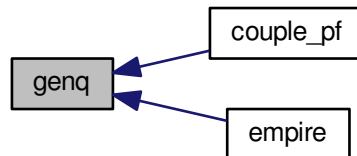
*Subroutine to estimate Q from a long model run.*

#### 6.29.1 Function/Subroutine Documentation

## 6.29.1.1 subroutine genq ( )

Subroutine to estimate Q from a long model run.

Here is the caller graph for this function:



## 6.30 src/utls/histogram.f90 File Reference

## Data Types

- module [histogram\\_data](#)

*Module to control what variables are used to generate rank histograms.*

## 6.31 src/utls/quicksort.f90 File Reference

## Functions/Subroutines

- recursive subroutine [quicksort\\_d](#) (a, na)

*subroutine to sort using the quicksort algorithm*

- subroutine [insertionsort\\_d](#) (A, nA)

*subroutine to sort using the insertionsort algorithm*

## 6.31.1 Function/Subroutine Documentation

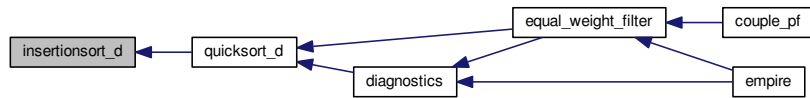
## 6.31.1.1 subroutine insertionsort\_d ( real(kind=kind(1.0d0)), dimension(na), intent(inout) A, integer, intent(in) nA )

subroutine to sort using the insertionsort algorithm

## Parameters

<code>in, out</code>	<code>a</code>	array of doubles to be sorted
<code>in</code>	<code>na</code>	dimension of array a

Here is the caller graph for this function:



#### 6.31.1.2 recursive subroutine `quicksort_d` ( `real(kind=kind(1.0d0))`, `dimension(na)`, `intent(inout) a`, `integer, intent(in) na` )

subroutine to sort using the quicksort algorithm

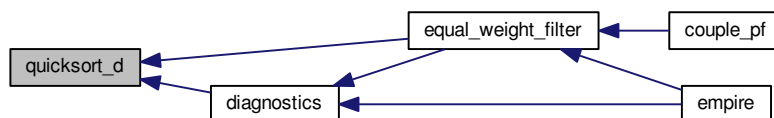
##### Parameters

<code>in, out</code>	<code>a</code>	array of doubles to be sorted
<code>in</code>	<code>na</code>	dimension of array <code>a</code>

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.32 `src/utils/random_d.f90` File Reference

### Data Types

- module [random](#)

*A module for random number generation from the following distributions:*



# Index

- allocate\_data
  - comms, 11
- allocate\_pf
  - pf\_control, 15
- alltests
  - alltests.f90, 66
- alltests.f90
  - alltests, 66
- bin\_prob
  - random, 23
- bprime
  - operator\_wrappers.f90, 60
- check\_scaling
  - stochastic\_model.f90, 54
- comms, 11
  - allocate\_data, 11
  - cpl\_mpi\_comm, 12
  - deallocate\_data, 11
  - gblcount, 12
  - gbldisp, 12
  - initialise\_mpi, 12
  - mype\_id, 12
  - myrank, 12
  - npfs, 12
  - nproc, 12
  - pf\_mpi\_comm, 12
  - pfrank, 12
- configure\_model
  - model\_specific.f90, 35
- count
  - pf\_control::pf\_control\_type, 18
- couple\_pf
  - old\_pf\_couple.f90, 45
- couple\_root
  - pf\_control::pf\_control\_type, 18
- cpl\_mpi\_comm
  - comms, 12
- data\_io.f90
  - get\_observation\_data, 73
  - output\_from\_pf, 74
  - save\_observation\_data, 74
  - save\_truth, 75
- deallocate\_data
  - comms, 11
- deallocate\_pf
  - pf\_control, 16
- diagnostics
  - diagnostics.f90, 75
- diagnostics.f90
  - diagnostics, 75
  - trajectories, 76
- dp
  - random, 31
- eakf\_analysis
  - eakf\_analysis.f90, 48
- eakf\_analysis.f90
  - eakf\_analysis, 48
- efac
  - pf\_control::pf\_control\_type, 18
- empire
  - pf\_couple.f90, 46
- enkf\_specific.f90
  - get\_local\_observation\_data, 49
  - h\_local, 49
  - localise\_enkf, 50
  - solve\_rhalf\_local, 50
- equal\_weight\_filter
  - equivalent\_weights\_step.f90, 51
- equivalent\_weights\_step.f90
  - equal\_weight\_filter, 51
- etkf\_analysis
  - etkf\_analysis.f90, 52
- etkf\_analysis.f90
  - etkf\_analysis, 52
- gblcount
  - comms, 12
- gbldisp
  - comms, 12
- gen\_data
  - pf\_control::pf\_control\_type, 18
- gen\_q
  - pf\_control::pf\_control\_type, 18
- gen\_rand.f90
  - mixture\_randomnumbers1d, 55
  - mixture\_randomnumbers2d, 56
  - normal\_randomnumbers1d, 57
  - normal\_randomnumbers2d, 57
  - random\_seed\_mpi, 58
  - uniform\_randomnumbers1d, 59
- genQ.f90
  - genq, 76
- genq
  - genQ.f90, 76
- get\_local\_observation\_data
  - enkf\_specific.f90, 49

- get\_observation\_data
  - data\_io.f90, 73
- h
  - model\_specific.f90, 36
- h\_local
  - enkf\_specific.f90, 49
- h\_tests
  - tests.f90, 70
- histogram\_data, 13
  - kill\_histogram\_data, 13
  - load\_histogram\_data, 13
  - rank\_hist\_list, 13
  - rank\_hist\_nums, 13
  - rhl\_n, 13
  - rhn\_n, 13
- hqht\_plus\_r, 14
  - hqhtr\_factor, 14
  - kill\_hqhtr, 14
  - load\_hqhtr, 14
- hqhtr\_factor
  - hqht\_plus\_r, 14
- hqhtr\_tests
  - tests.f90, 71
- ht
  - model\_specific.f90, 37
- human\_readable
  - pf\_control::pf\_control\_type, 18
- init
  - pf\_control::pf\_control\_type, 18
- initialise\_mpi
  - comms, 12
- innerhqht\_plus\_r\_1
  - operator\_wrappers.f90, 60
- innerr\_1
  - operator\_wrappers.f90, 61
- insertionsort\_d
  - quicksort.f90, 77
- k
  - operator\_wrappers.f90, 62
- keep
  - pf\_control::pf\_control\_type, 18
- kill\_histogram\_data
  - histogram\_data, 13
- kill\_hqhtr
  - hqht\_plus\_r, 14
- killq
  - qdata, 21
- killr
  - rdata, 32
- lngamma
  - random, 23
- load\_histogram\_data
  - histogram\_data, 13
- load\_hqhtr
  - hqht\_plus\_r, 14
- loadq
  - qdata, 21
- loadr
  - rdata, 32
- localise\_enkf
  - enkf\_specific.f90, 50
- mean
  - pf\_control::pf\_control\_type, 18
- mixture\_randomnumbers1d
  - gen\_rand.f90, 55
- mixture\_randomnumbers2d
  - gen\_rand.f90, 56
- model\_specific.f90, 35
  - configure\_model, 35
  - h, 36
  - ht, 37
  - q, 38
  - qhalf, 40
  - r, 41
  - rhalf, 42
  - solve\_hqht\_plus\_r, 43
  - solve\_r, 44
- mype\_id
  - comms, 12
- myrank
  - comms, 12
- nens
  - pf\_control::pf\_control\_type, 19
- nfac
  - pf\_control::pf\_control\_type, 19
- normal\_randomnumbers1d
  - gen\_rand.f90, 57
- normal\_randomnumbers2d
  - gen\_rand.f90, 57
- npfs
  - comms, 12
- nproc
  - comms, 12
- nudgefac
  - pf\_control::pf\_control\_type, 19
- obs\_dim
  - sizes, 33
- old\_pf\_couple.f90
  - couple\_pf, 45
- operator\_wrappers.f90
  - bprime, 60
  - innerhqht\_plus\_r\_1, 60
  - innerr\_1, 61
  - k, 62
- output\_from\_pf
  - data\_io.f90, 74
- particles
  - pf\_control::pf\_control\_type, 19
- perturb\_particle
  - perturb\_particle.f90, 64

- perturb\_particle.f90
  - perturb\_particle, 64
  - update\_state, 64
- pf
  - pf\_control, 16
- pf\_control, 14
  - allocate\_pf, 15
  - deallocate\_pf, 16
  - pf, 16
  - set\_pf\_controls, 16
- pf\_control::pf\_control\_type, 17
  - count, 18
  - couple\_root, 18
  - efac, 18
  - gen\_data, 18
  - gen\_q, 18
  - human\_readable, 18
  - init, 18
  - keep, 18
  - mean, 18
  - nens, 19
  - nfac, 19
  - nudgefac, 19
  - particles, 19
  - psi, 19
  - qscale, 19
  - talagrand, 19
  - time, 19
  - time\_bwn\_obs, 19
  - time\_obs, 19
  - timestep, 19
  - type, 19
  - ufac, 20
  - use\_mean, 20
  - use\_rmse, 20
  - use\_talagrand, 20
  - use\_traj, 20
  - use\_var, 20
  - use\_weak, 20
  - weight, 20
- pf\_couple.f90
  - empire, 46
- pf\_mpi\_comm
  - comms, 12
- pfrank
  - comms, 12
- proposal\_filter
  - proposal\_filter.f90, 53
- proposal\_filter.f90
  - proposal\_filter, 53
- psi
  - pf\_control::pf\_control\_type, 19
- q
  - model\_specific.f90, 38
- q\_tests
  - tests.f90, 71
- qcol
  - qdata, 22
- qdata, 20
  - killq, 21
  - loadq, 21
  - qcol, 22
  - qdiag, 22
  - qn, 22
  - qne, 22
  - qrow, 22
  - qscale, 22
  - qval, 22
- qdiag
  - qdata, 22
- qhalf
  - model\_specific.f90, 40
- qn
  - qdata, 22
- qne
  - qdata, 22
- qrow
  - qdata, 22
- qscale
  - pf\_control::pf\_control\_type, 19
  - qdata, 22
- quicksort.f90
  - insertionsort\_d, 77
  - quicksort\_d, 78
- quicksort\_d
  - quicksort.f90, 78
- qval
  - qdata, 22
- r
  - model\_specific.f90, 41
- r\_tests
  - tests.f90, 72
- random, 22
  - bin\_prob, 23
  - dp, 31
  - lngamma, 23
  - random\_beta, 23
  - random\_binomial1, 24
  - random\_binomial2, 24
  - random\_cauchy, 24
  - random\_chisq, 24
  - random\_exponential, 25
  - random\_gamma, 25
  - random\_gamma1, 26
  - random\_gamma2, 26
  - random\_inv\_gauss, 27
  - random\_mvnorm, 27
  - random\_neg\_binomial, 27
  - random\_normal, 28
  - random\_order, 29
  - random\_poisson, 29
  - random\_t, 30
  - random\_von\_mises, 30
  - random\_weibull, 30
  - seed\_random\_number, 31
- random\_beta

- random, 23
- random\_binomial1
  - random, 24
- random\_binomial2
  - random, 24
- random\_cauchy
  - random, 24
- random\_chisq
  - random, 24
- random\_exponential
  - random, 25
- random\_gamma
  - random, 25
- random\_gamma1
  - random, 26
- random\_gamma2
  - random, 26
- random\_inv\_gauss
  - random, 27
- random\_mvnorm
  - random, 27
- random\_neg\_binomial
  - random, 27
- random\_normal
  - random, 28
- random\_order
  - random, 29
- random\_poisson
  - random, 29
- random\_seed\_mpi
  - gen\_rand.f90, 58
- random\_t
  - random, 30
- random\_von\_mises
  - random, 30
- random\_weibull
  - random, 30
- rank\_hist\_list
  - histogram\_data, 13
- rank\_hist\_nums
  - histogram\_data, 13
- rcol
  - rdata, 32
- rdata, 31
  - killr, 32
  - loadr, 32
  - rcol, 32
  - rdiag, 32
  - rn, 32
  - rne, 33
  - rrow, 33
  - rval, 33
- rdiag
  - rdata, 32
- resample
  - resample.f90, 65
- resample.f90
  - resample, 65
- rhalf
  - model\_specific.f90, 42
- rhl\_n
  - histogram\_data, 13
- rhn\_n
  - histogram\_data, 13
- rn
  - rdata, 32
- rne
  - rdata, 33
- rrow
  - rdata, 33
- rval
  - rdata, 33
- save\_observation\_data
  - data\_io.f90, 74
- save\_truth
  - data\_io.f90, 75
- seed\_random\_number
  - random, 31
- set\_pf\_controls
  - pf\_control, 16
- sir\_filter
  - sir\_filter.f90, 53
- sir\_filter.f90
  - sir\_filter, 53
- sizes, 33
  - obs\_dim, 33
  - state\_dim, 33
- solve\_hqht\_plus\_r
  - model\_specific.f90, 43
- solve\_r
  - model\_specific.f90, 44
- solve\_rhalf\_local
  - enkf\_specific.f90, 50
- src/DOC\_README.txt, 48
- src/controllers/old\_pf\_couple.f90, 45
- src/controllers/pf\_control.f90, 46
- src/controllers/pf\_couple.f90, 46
- src/controllers/sizes.f90, 47
- src/data/Qdata.f90, 47
- src/data/Rdata.f90, 48
- src/filters/eakf\_analysis.f90, 48
- src/filters/enkf\_specific.f90, 49
- src/filters/equivalent\_weights\_step.f90, 51
- src/filters/etkf\_analysis.f90, 52
- src/filters/proposal\_filter.f90, 53
- src/filters/sir\_filter.f90, 53
- src/filters/stochastic\_model.f90, 54
- src/operations/gen\_rand.f90, 55
- src/operations/operator\_wrappers.f90, 59
- src/operations/perturb\_particle.f90, 63
- src/operations/resample.f90, 65
- src/tests/alltests.f90, 66
- src/tests/test\_h.f90, 67
- src/tests/test\_hqhtr.f90, 68
- src/tests/test\_q.f90, 68
- src/tests/test\_r.f90, 69

- src/tests/tests.f90, [70](#)
- src/utls/comms.f90, [73](#)
- src/utls/data\_io.f90, [73](#)
- src/utls/diagnostics.f90, [75](#)
- src/utls/genQ.f90, [76](#)
- src/utls/histogram.f90, [77](#)
- src/utls/quicksort.f90, [77](#)
- src/utls/random\_d.f90, [78](#)
- state\_dim
  - sizes, [33](#)
- stochastic\_model
  - stochastic\_model.f90, [54](#)
- stochastic\_model.f90
  - check\_scaling, [54](#)
  - stochastic\_model, [54](#)
- talagrand
  - pf\_control::pf\_control\_type, [19](#)
- test\_h
  - test\_h.f90, [67](#)
- test\_h.f90
  - test\_h, [67](#)
- test\_hqhtr
  - test\_hqhtr.f90, [68](#)
- test\_hqhtr.f90
  - test\_hqhtr, [68](#)
- test\_q
  - test\_q.f90, [69](#)
- test\_q.f90
  - test\_q, [69](#)
- test\_r
  - test\_r.f90, [69](#)
- test\_r.f90
  - test\_r, [69](#)
- tests.f90
  - h\_tests, [70](#)
  - hqhtr\_tests, [71](#)
  - q\_tests, [71](#)
  - r\_tests, [72](#)
- time
  - pf\_control::pf\_control\_type, [19](#)
- time\_bwn\_obs
  - pf\_control::pf\_control\_type, [19](#)
- time\_obs
  - pf\_control::pf\_control\_type, [19](#)
- timestep
  - pf\_control::pf\_control\_type, [19](#)
- trajectories
  - diagnostics.f90, [76](#)
- type
  - pf\_control::pf\_control\_type, [19](#)
- ufac
  - pf\_control::pf\_control\_type, [20](#)
- uniformrandomnumbers1d
  - gen\_rand.f90, [59](#)
- update\_state
  - perturb\_particle.f90, [64](#)
- use\_mean
  - pf\_control::pf\_control\_type, [20](#)
- use\_rmse
  - pf\_control::pf\_control\_type, [20](#)
- use\_talagrand
  - pf\_control::pf\_control\_type, [20](#)
- use\_traj
  - pf\_control::pf\_control\_type, [20](#)
- use\_var
  - pf\_control::pf\_control\_type, [20](#)
- use\_weak
  - pf\_control::pf\_control\_type, [20](#)
- weight
  - pf\_control::pf\_control\_type, [20](#)