

EMPIRE DA

0.1

Generated by Doxygen 1.8.8

Wed Nov 12 2014 13:43:35



# 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	Examples . . . . .	3
1.8	Bug Reports and Functionality Requests . . . . .	4
<b>2</b>	<b>Data Type Index</b>	<b>5</b>
2.1	Data Types List . . . . .	5
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List . . . . .	7
<b>4</b>	<b>Data Type Documentation</b>	<b>9</b>
4.1	comms Module Reference . . . . .	9
4.1.1	Detailed Description . . . . .	9
4.1.2	Member Function/Subroutine Documentation . . . . .	10
4.1.2.1	allocate_data . . . . .	10
4.1.2.2	deallocate_data . . . . .	10
4.1.2.3	initialise_mpi . . . . .	10
4.1.3	Member Data Documentation . . . . .	10
4.1.3.1	cpl_mpi_comm . . . . .	10
4.1.3.2	gblcount . . . . .	10
4.1.3.3	gbldisp . . . . .	11
4.1.3.4	mype_id . . . . .	11
4.1.3.5	myrank . . . . .	11
4.1.3.6	npfs . . . . .	11

4.1.3.7	<a href="#">nproc</a>	11
4.1.3.8	<a href="#">pf_mpi_comm</a>	11
4.1.3.9	<a href="#">pfrank</a>	11
4.2	<a href="#">histogram_data Module Reference</a>	11
4.2.1	<a href="#">Detailed Description</a>	12
4.2.2	<a href="#">Member Function/Subroutine Documentation</a>	12
4.2.2.1	<a href="#">kill_histogram_data</a>	12
4.2.2.2	<a href="#">load_histogram_data</a>	12
4.2.3	<a href="#">Member Data Documentation</a>	12
4.2.3.1	<a href="#">rank_hist_list</a>	12
4.2.3.2	<a href="#">rank_hist_nums</a>	12
4.2.3.3	<a href="#">rhl_n</a>	12
4.2.3.4	<a href="#">rhn_n</a>	12
4.3	<a href="#">hqht_plus_r Module Reference</a>	13
4.3.1	<a href="#">Detailed Description</a>	13
4.3.2	<a href="#">Member Function/Subroutine Documentation</a>	13
4.3.2.1	<a href="#">hqhtr_factor</a>	13
4.3.2.2	<a href="#">kill_hqhtr</a>	13
4.3.2.3	<a href="#">load_hqhtr</a>	13
4.4	<a href="#">pf_control Module Reference</a>	14
4.4.1	<a href="#">Detailed Description</a>	14
4.4.2	<a href="#">Member Function/Subroutine Documentation</a>	15
4.4.2.1	<a href="#">allocate_pf</a>	15
4.4.2.2	<a href="#">deallocate_pf</a>	15
4.4.2.3	<a href="#">parse_pf_parameters</a>	15
4.4.2.4	<a href="#">set_pf_controls</a>	16
4.4.3	<a href="#">Member Data Documentation</a>	17
4.4.3.1	<a href="#">pf</a>	17
4.5	<a href="#">pf_control::pf_control_type Type Reference</a>	17
4.5.1	<a href="#">Detailed Description</a>	19
4.5.2	<a href="#">Member Data Documentation</a>	19
4.5.2.1	<a href="#">count</a>	19
4.5.2.2	<a href="#">couple_root</a>	19
4.5.2.3	<a href="#">efac</a>	19
4.5.2.4	<a href="#">gen_data</a>	19
4.5.2.5	<a href="#">gen_q</a>	19
4.5.2.6	<a href="#">human_readable</a>	19
4.5.2.7	<a href="#">init</a>	20
4.5.2.8	<a href="#">keep</a>	20
4.5.2.9	<a href="#">len</a>	20

4.5.2.10	mean	20
4.5.2.11	nens	20
4.5.2.12	nfac	20
4.5.2.13	nudgefac	20
4.5.2.14	particles	20
4.5.2.15	psi	21
4.5.2.16	qscale	21
4.5.2.17	rho	21
4.5.2.18	talagrand	21
4.5.2.19	time	21
4.5.2.20	time_bwn_obs	21
4.5.2.21	time_obs	21
4.5.2.22	timestep	21
4.5.2.23	type	21
4.5.2.24	ufac	22
4.5.2.25	use_mean	22
4.5.2.26	use_rmse	22
4.5.2.27	use_talagrand	22
4.5.2.28	use_traj	22
4.5.2.29	use_var	22
4.5.2.30	use_weak	22
4.5.2.31	weight	22
4.6	qdata Module Reference	23
4.6.1	Detailed Description	23
4.6.2	Member Function/Subroutine Documentation	23
4.6.2.1	killq	23
4.6.2.2	loadq	23
4.6.3	Member Data Documentation	24
4.6.3.1	qcol	24
4.6.3.2	qdiag	24
4.6.3.3	qn	24
4.6.3.4	qne	24
4.6.3.5	qrow	24
4.6.3.6	qscale	24
4.6.3.7	qval	25
4.7	random Module Reference	25
4.7.1	Detailed Description	25
4.7.2	Member Function/Subroutine Documentation	26
4.7.2.1	bin_prob	26
4.7.2.2	lngamma	26

4.7.2.3	<a href="#">random_beta</a>	26
4.7.2.4	<a href="#">random_binomial1</a>	26
4.7.2.5	<a href="#">random_binomial2</a>	27
4.7.2.6	<a href="#">random_cauchy</a>	27
4.7.2.7	<a href="#">random_chisq</a>	27
4.7.2.8	<a href="#">random_exponential</a>	27
4.7.2.9	<a href="#">random_gamma</a>	28
4.7.2.10	<a href="#">random_gamma1</a>	28
4.7.2.11	<a href="#">random_gamma2</a>	29
4.7.2.12	<a href="#">random_inv_gauss</a>	29
4.7.2.13	<a href="#">random_mvnorm</a>	29
4.7.2.14	<a href="#">random_neg_binomial</a>	29
4.7.2.15	<a href="#">random_normal</a>	29
4.7.2.16	<a href="#">random_order</a>	30
4.7.2.17	<a href="#">random_poisson</a>	30
4.7.2.18	<a href="#">random_t</a>	30
4.7.2.19	<a href="#">random_von_mises</a>	30
4.7.2.20	<a href="#">random_weibull</a>	31
4.7.2.21	<a href="#">seed_random_number</a>	31
4.7.3	<a href="#">Member Data Documentation</a>	31
4.7.3.1	<a href="#">dp</a>	31
4.8	<a href="#">rdata Module Reference</a>	31
4.8.1	<a href="#">Detailed Description</a>	31
4.8.2	<a href="#">Member Function/Subroutine Documentation</a>	32
4.8.2.1	<a href="#">killr</a>	32
4.8.2.2	<a href="#">loadr</a>	32
4.8.3	<a href="#">Member Data Documentation</a>	32
4.8.3.1	<a href="#">rcol</a>	32
4.8.3.2	<a href="#">rdiag</a>	32
4.8.3.3	<a href="#">rn</a>	32
4.8.3.4	<a href="#">rne</a>	33
4.8.3.5	<a href="#">rrow</a>	33
4.8.3.6	<a href="#">rval</a>	33
4.9	<a href="#">sizes Module Reference</a>	33
4.9.1	<a href="#">Detailed Description</a>	33
4.9.2	<a href="#">Member Data Documentation</a>	33
4.9.2.1	<a href="#">obs_dim</a>	33
4.9.2.2	<a href="#">state_dim</a>	33

5.1	model_specific.f90 File Reference	35
5.1.1	Function/Subroutine Documentation	35
5.1.1.1	configure_model	35
5.1.1.2	dist_st_ob	36
5.1.1.3	h	37
5.1.1.4	ht	38
5.1.1.5	q	39
5.1.1.6	qhalf	40
5.1.1.7	r	41
5.1.1.8	rhalf	41
5.1.1.9	solve_hqht_plus_r	42
5.1.1.10	solve_r	42
5.1.1.11	solve_rhalf	43
5.2	src/controllers/pf_control.f90 File Reference	44
5.3	src/controllers/pf_couple.f90 File Reference	44
5.3.1	Function/Subroutine Documentation	44
5.3.1.1	empire	44
5.4	src/controllers/pf_parameters.dat File Reference	45
5.5	src/controllers/sizes.f90 File Reference	45
5.6	src/data/Qdata.f90 File Reference	46
5.7	src/data/Rdata.f90 File Reference	46
5.8	src/DOC_README.txt File Reference	46
5.9	src/filters/deterministic_model.f90 File Reference	46
5.9.1	Function/Subroutine Documentation	46
5.9.1.1	deterministic_model	46
5.10	src/filters/eakf_analysis.f90 File Reference	46
5.10.1	Function/Subroutine Documentation	47
5.10.1.1	eakf_analysis	47
5.11	src/filters/enkf_specific.f90 File Reference	47
5.11.1	Function/Subroutine Documentation	47
5.11.1.1	get_local_observation_data	47
5.11.1.2	h_local	48
5.11.1.3	localise_enkf	48
5.11.1.4	solve_rhalf_local	49
5.12	src/filters/equivalent_weights_step.f90 File Reference	49
5.12.1	Function/Subroutine Documentation	49
5.12.1.1	equal_weight_filter	49
5.13	src/filters/etkf_analysis.f90 File Reference	50
5.13.1	Function/Subroutine Documentation	50
5.13.1.1	etkf_analysis	51

5.14	src/filters/letkf_analysis.f90 File Reference	51
5.14.1	Function/Subroutine Documentation	51
5.14.1.1	letkf_analysis	51
5.15	src/filters/proposal_filter.f90 File Reference	52
5.15.1	Function/Subroutine Documentation	52
5.15.1.1	proposal_filter	52
5.16	src/filters/sir_filter.f90 File Reference	53
5.16.1	Function/Subroutine Documentation	53
5.16.1.1	sir_filter	53
5.17	src/filters/stochastic_model.f90 File Reference	54
5.17.1	Function/Subroutine Documentation	54
5.17.1.1	check_scaling	54
5.17.1.2	stochastic_model	55
5.18	src/operations/gen_rand.f90 File Reference	55
5.18.1	Function/Subroutine Documentation	55
5.18.1.1	mixturerandomnumbers1d	56
5.18.1.2	mixturerandomnumbers2d	57
5.18.1.3	normalrandomnumbers1d	58
5.18.1.4	normalrandomnumbers2d	59
5.18.1.5	random_seed_mpi	60
5.18.1.6	uniformrandomnumbers1d	60
5.19	src/operations/operator_wrappers.f90 File Reference	61
5.19.1	Function/Subroutine Documentation	61
5.19.1.1	bprime	61
5.19.1.2	innerhqht_plus_r_1	62
5.19.1.3	innerr_1	63
5.19.1.4	k	63
5.20	src/operations/perturb_particle.f90 File Reference	64
5.20.1	Function/Subroutine Documentation	64
5.20.1.1	perturb_particle	64
5.20.1.2	update_state	65
5.21	src/operations/resample.f90 File Reference	66
5.21.1	Function/Subroutine Documentation	66
5.21.1.1	resample	66
5.22	src/tests/alltests.f90 File Reference	67
5.22.1	Function/Subroutine Documentation	67
5.22.1.1	alltests	67
5.23	src/tests/test_h.f90 File Reference	67
5.23.1	Function/Subroutine Documentation	68
5.23.1.1	test_h	68



5.24	src/tests/test_hqhtr.f90 File Reference	68
5.24.1	Function/Subroutine Documentation	68
5.24.1.1	test_hqhtr	68
5.25	src/tests/test_q.f90 File Reference	69
5.25.1	Function/Subroutine Documentation	69
5.25.1.1	test_q	69
5.26	src/tests/test_r.f90 File Reference	70
5.26.1	Function/Subroutine Documentation	70
5.26.1.1	test_r	70
5.27	src/tests/tests.f90 File Reference	71
5.27.1	Function/Subroutine Documentation	71
5.27.1.1	h_tests	71
5.27.1.2	hqhtr_tests	72
5.27.1.3	q_tests	73
5.27.1.4	r_tests	74
5.28	src/utls/comms.f90 File Reference	75
5.29	src/utls/data_io.f90 File Reference	75
5.29.1	Function/Subroutine Documentation	76
5.29.1.1	get_observation_data	76
5.29.1.2	get_state	76
5.29.1.3	output_from_pf	77
5.29.1.4	save_observation_data	77
5.29.1.5	save_state	77
5.29.1.6	save_truth	78
5.30	src/utls/diagnostics.f90 File Reference	78
5.30.1	Function/Subroutine Documentation	78
5.30.1.1	diagnostics	78
5.30.1.2	trajectories	79
5.31	src/utls/genQ.f90 File Reference	79
5.31.1	Function/Subroutine Documentation	79
5.31.1.1	genq	79
5.32	src/utls/histogram.f90 File Reference	80
5.33	src/utls/quicksort.f90 File Reference	80
5.33.1	Function/Subroutine Documentation	80
5.33.1.1	insertionsort_d	80
5.33.1.2	quicksort_d	81
5.34	src/utls/random_d.f90 File Reference	82



# Chapter 1

## EMPIRE Data Assimilation Documentation

### Author

Philip A. Browne [p.browne@reading.ac.uk](mailto:p.browne@reading.ac.uk)

### Date

Time-stamp: <2014-10-08 16:40:26 pbrowne>

## 1.1 Downloading

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

```
1 git clone https://www.bitbucket.org/pbrowne/empire-data-assimilation.git
```

To upgrade to the latest versions of the codes, use the following command:

```
1 git pull https://www.bitbucket.org/pbrowne/empire-data-assimilation.git
```

### Copyright

These codes are distributed under the GNU GPU v3 License. See LICENSE.txt.

## 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

This has been tested with gfortran 4.8.2, crayftn 8.2.6 and ifort 14.0.1.106

- `FCOPTS` The options for the fortran compiler
- `LIB_LIST` The libraries to be called. Note this must include BLAS and LAPACK
- `MODFLAG` The flag to specify where module files should be placed by the fortran compiler. Examples are

- gfortran: -J
- ifort: -module
- crayftn: -em -J
- pgfortran: -module

To compile the source code, simply then type the command

```
1 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:

```
1 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.

```
1 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

```
1 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,  $x = R^{-1}b$
- **solve\_rhalf** This is a linear solve with the square root of the observation error covariance matrix, i.e. given  $b$ , find  $x$  such that  $R^{\frac{1}{2}}x = b$  or indeed,  $x = R^{-\frac{1}{2}}b$
- **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  $(HQH^T + R)$
- **dist\_st\_ob** This specifies the distance between a an element of the state vector and an element of the observation vector

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

Full instructions on how to put the EMPIRE MPI commands into a new model can be found at [www.met.rdg.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:

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

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

The empire executable is controlled by the namelist data file [pf\\_parameters.dat](#). As such, this file should be put in the directory where empire is executed.

## 1.7 Examples

In the directory `examples` there is currently one example of how to use EMPIRE, specifically with the Lorenz 1996 model. In the directory you will find an example [model\\_specific.f90](#) file setup for that model, along with a file `instructions.txt` which will lead you step by step through how to run a twin experiment.

## 1.8 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

# Data Type Index

### 2.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">comms</a>	Module containing EMPIRE coupling data . . . . .	9
<a href="#">histogram_data</a>	Module to control what variables are used to generate rank histograms . . . . .	11
<a href="#">hqht_plus_r</a>	. . . . .	13
<a href="#">pf_control</a>	Module <a href="#">pf_control</a> holds 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 $Q$ . . . . .	23
<a href="#">random</a>	A module for random number generation from the following distributions: . . . . .	25
<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 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">model_specific.f90</a>	35
<a href="#">src/controllers/pf_control.f90</a>	44
<a href="#">src/controllers/pf_couple.f90</a>	44
<a href="#">src/controllers/pf_parameters.dat</a>	45
<a href="#">src/controllers/sizes.f90</a>	45
<a href="#">src/data/Qdata.f90</a>	46
<a href="#">src/data/Rdata.f90</a>	46
<a href="#">src/filters/deterministic_model.f90</a>	46
<a href="#">src/filters/eakf_analysis.f90</a>	46
<a href="#">src/filters/enkf_specific.f90</a>	47
<a href="#">src/filters/ekf_analysis.f90</a>	49
<a href="#">src/filters/etkf_analysis.f90</a>	50
<a href="#">src/filters/letkf_analysis.f90</a>	51
<a href="#">src/filters/proposal_filter.f90</a>	52
<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>	61
<a href="#">src/operations/perturb_particle.f90</a>	64
<a href="#">src/operations/resample.f90</a>	66
<a href="#">src/tests/alltests.f90</a>	67
<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>	69
<a href="#">src/tests/test_r.f90</a>	70
<a href="#">src/tests/tests.f90</a>	71
<a href="#">src/utls/comms.f90</a>	75
<a href="#">src/utls/data_io.f90</a>	75
<a href="#">src/utls/diagnostics.f90</a>	78
<a href="#">src/utls/genQ.f90</a>	79
<a href="#">src/utls/histogram.f90</a>	80
<a href="#">src/utls/quicksort.f90</a>	80
<a href="#">src/utls/random_d.f90</a>	82



## Chapter 4

# Data Type Documentation

### 4.1 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](#)  
*the communicator between the empire codes and the model master nodes*
- integer [mype\\_id](#)  
*the rank of this process on MPI\_COMM\_WORLD*
- integer [myrank](#)  
*the rank of this process on CPL\_MPI\_COMM*
- integer [nproc](#)  
*the total number of processes*
- integer [pf\\_mpi\\_comm](#)  
*the communicator between DA processes*
- integer [pfrank](#)  
*the rank of this process on PF\_MPI\_COMM*
- integer [npfs](#)  
*the total number of DA processes*
- integer, dimension(:), allocatable [gblcount](#)  
*the number of ensemble members associated with each DA process*
- integer, dimension(:), allocatable [gbldisp](#)  
*the displacements of each each ensemble member relative to pfrank=0. VERY useful for mpi\_gatherv and mpi\_↔  
scatterv on pf\_mpi\_comm*

#### 4.1.1 Detailed Description

Module containing EMPIRE coupling data.

Definition at line 30 of file comms.f90.

## 4.1.2 Member Function/Subroutine Documentation

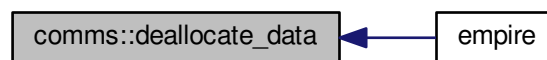
### 4.1.2.1 subroutine `comms::allocate_data ( )`

Definition at line 47 of file `comms.f90`.

### 4.1.2.2 subroutine `comms::deallocate_data ( )`

Definition at line 53 of file `comms.f90`.

Here is the caller graph for this function:

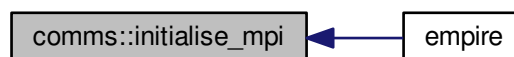


### 4.1.2.3 subroutine `comms::initialise_mpi ( )`

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

Definition at line 60 of file `comms.f90`.

Here is the caller graph for this function:



## 4.1.3 Member Data Documentation

### 4.1.3.1 integer `comms::cpl_mpi_comm`

the communicator between the empire codes and the model master nodes

Definition at line 31 of file `comms.f90`.

### 4.1.3.2 integer, dimension(:), allocatable `comms::gblcount`

the number of ensemble members associated with each DA process

Definition at line 39 of file `comms.f90`.

**4.1.3.3 integer dimension(:), allocatable comms::gbldisp**

the displacements of each ensemble member relative to pfrank=0. VERY useful for mpi\_gatherv and mpi\_↔scaterv on pf\_mpi\_comm

Definition at line 41 of file comms.f90.

**4.1.3.4 integer comms::mype\_id**

the rank of this process on MPI\_COMM\_WORLD

Definition at line 33 of file comms.f90.

**4.1.3.5 integer comms::myrank**

the rank of this process on CPL\_MPI\_COMM

Definition at line 34 of file comms.f90.

**4.1.3.6 integer comms::npfs**

the total number of DA processes

Definition at line 38 of file comms.f90.

**4.1.3.7 integer comms::nproc**

the total number of processes

Definition at line 35 of file comms.f90.

**4.1.3.8 integer comms::pf\_mpi\_comm**

the communicator between DA processes

Definition at line 36 of file comms.f90.

**4.1.3.9 integer comms::pfrank**

the rank of this process on PF\_MPI\_COMM

Definition at line 37 of file comms.f90.

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

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

## 4.2 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](#)

### 4.2.1 Detailed Description

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

Definition at line 29 of file histogram.f90.

### 4.2.2 Member Function/Subroutine Documentation

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

subroutine to clean up arrays used in rank histograms

Definition at line 57 of file histogram.f90.

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

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

Definition at line 37 of file histogram.f90.

### 4.2.3 Member Data Documentation

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

Definition at line 30 of file histogram.f90.

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

Definition at line 31 of file histogram.f90.

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

Definition at line 32 of file histogram.f90.

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

Definition at line 32 of file histogram.f90.

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

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

## 4.3 hqht\_plus\_r Module Reference

### Public Member Functions

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

#### 4.3.1 Detailed Description

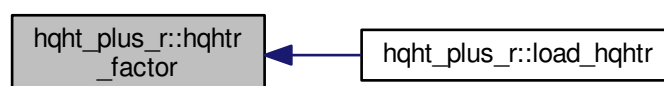
Definition at line 59 of file Rdata.f90.

#### 4.3.2 Member Function/Subroutine Documentation

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

Definition at line 69 of file Rdata.f90.

Here is the caller graph for this function:



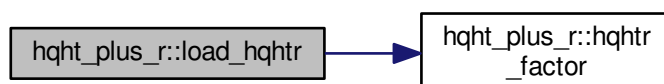
##### 4.3.2.2 subroutine hqht\_plus\_r::kill\_hqhtr ( )

Definition at line 74 of file Rdata.f90.

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

Definition at line 65 of file Rdata.f90.

Here is the call graph for this function:



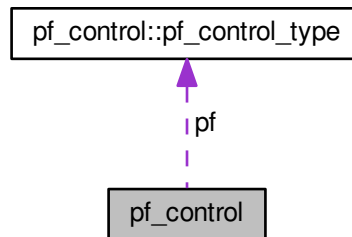
The documentation for this module was generated from the following file:

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

## 4.4 pf\_control Module Reference

module [pf\\_control](#) holds 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 to ensure [pf\\_control](#) data is ok*
- subroutine [parse\\_pf\\_parameters](#)  
*subroutine to read the namelist file and save it to pf datatype Here we read [pf\\_parameters.dat](#)*
- subroutine [allocate\\_pf](#)  
*subroutine to allocate space for the filtering code*
- subroutine [deallocate\\_pf](#)  
*subroutine to deallocate space for the filtering code*

### Public Attributes

- type([pf\\_control\\_type](#)), save [pf](#)  
*the derived data type holding all controlling data*

#### 4.4.1 Detailed Description

module [pf\\_control](#) holds all the information to control the the main program

Definition at line 29 of file [pf\\_control.f90](#).



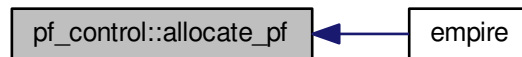
## 4.4.2 Member Function/Subroutine Documentation

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

subroutine to allocate space for the filtering code

Definition at line 343 of file pf\_control.f90.

Here is the caller graph for this function:



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

subroutine to deallocate space for the filtering code

Definition at line 365 of file pf\_control.f90.

### 4.4.2.3 subroutine pf\_control::parse\_pf\_parameters ( )

subroutine to read the namelist file and save it to pf datatype Here we read [pf\\_parameters.dat](#)

[pf\\_parameters.dat](#) is a fortran namelist file. As such, within it there must be a line beginning

&pf\_params

To make it (probably) work, ensure there is a forward slash on the penultimate line and a blank line to end the file

This is just the fortran standard for namelists though.

On to the content...in any order, the [pf\\_parameters.dat](#) may contain the following things:

Integers:

- [time\\_obs](#)
- [time\\_bwn\\_obs](#)

Reals, double precision:

- [nudgefac](#)
- [nfac](#)
- [ufac](#)
- Qscale
- [keep](#)
- [rho](#)
- [len](#)

2 Characters:

- [type](#)

1 Character:

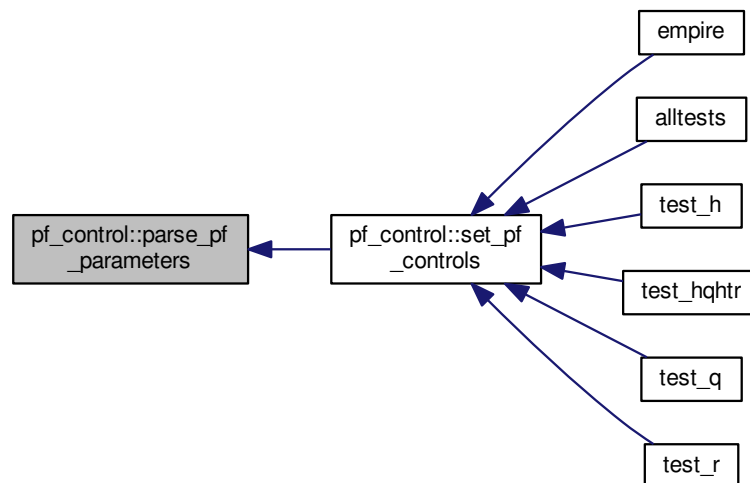
- [init](#)

Logicals:

- [gen\\_Q](#)
- [gen\\_data](#)
- [use\\_talagrand](#)
- [use\\_weak](#)
- [use\\_var](#)
- [use\\_traj](#)
- [use\\_rmse](#)
- [human\\_readable](#)

Definition at line 163 of file pf\_control.f90.

Here is the caller graph for this function:



#### 4.4.2.4 subroutine `pf_control::set_pf_controls` ( )

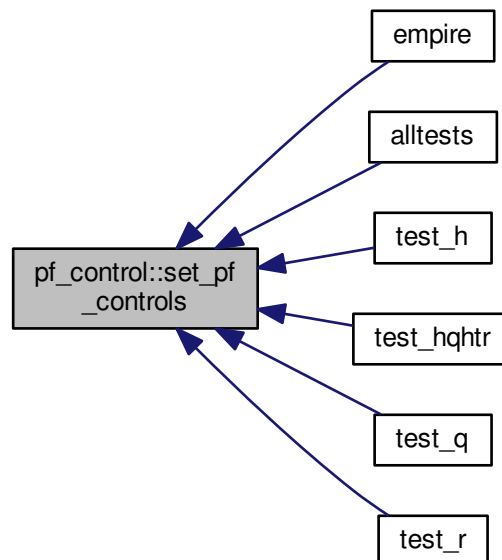
subroutine to ensure [pf\\_control](#) data is ok

Definition at line 96 of file pf\_control.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



### 4.4.3 Member Data Documentation

#### 4.4.3.1 `type(pf_control_type)`, save `pf_control::pf`

the derived data type holding all controlling data

Definition at line 91 of file `pf_control.f90`.

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

- `src/controllers/pf_control.f90`

## 4.5 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*
- real(kind=kind(1.0d0)) `rho`  
*enkf inflation factor so that  $P_f = (1 + \rho)P_f$*
- real(kind=kind(1.0d0)) `len`  
 *$R$  localisation length scale.*
- 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 currently this has a number of options:*
- character(1) `init`  
*which method to initialise ensemble currently this has a number of options:*

#### 4.5.1 Detailed Description

Definition at line 31 of file pf\_control.f90.

#### 4.5.2 Member Data Documentation

##### 4.5.2.1 integer pf\_control::pf\_control\_type::count

number of ensemble members associated with this MPI process

Definition at line 64 of file pf\_control.f90.

##### 4.5.2.2 integer pf\_control::pf\_control\_type::couple\_root

empire master processor

Definition at line 56 of file pf\_control.f90.

##### 4.5.2.3 real(kind=kind(1.0d0)) pf\_control::pf\_control\_type::efac

Definition at line 46 of file pf\_control.f90.

##### 4.5.2.4 logical pf\_control::pf\_control\_type::gen\_data

true generates synthetic obs for a twin experiment

Definition at line 37 of file pf\_control.f90.

##### 4.5.2.5 logical pf\_control::pf\_control\_type::gen\_q

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

Definition at line 38 of file pf\_control.f90.

##### 4.5.2.6 logical pf\_control::pf\_control\_type::human\_readable

unused

Definition at line 40 of file pf\_control.f90.

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

which method to initialise ensemble currently this has a number of options:

- N – perturb around the model initial conditions with random noise distributed  $\mathcal{N}(0, I)$
- P – perturb around the model initial conditions with random noise distributed  $\mathcal{N}(0, Q)$
- R – read model states from `rst` folder where each ensemble member is stored in the file `rst/##.state`
- S – read model states from `start` folder where each ensemble member is stored in the file `start/##.state`

Definition at line 74 of file `pf_control.f90`.

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

proportion of particles to keep in EWPF EW step

Definition at line 47 of file `pf_control.f90`.

#### 4.5.2.9 `real(kind=kind(1.0d0)) pf_control::pf_control_type::len`

R localisation length scale.

Definition at line 54 of file `pf_control.f90`.

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

mean state vector

Definition at line 43 of file `pf_control.f90`.

#### 4.5.2.11 `integer pf_control::pf_control_type::nens`

the total number of ensemble members

Definition at line 32 of file `pf_control.f90`.

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

standard deviation of normal distribution in mixture density

Definition at line 44 of file `pf_control.f90`.

#### 4.5.2.13 `real(kind=kind(1.0d0)) pf_control::pf_control_type::nudgef`

the nudging factor

Definition at line 36 of file `pf_control.f90`.

#### 4.5.2.14 `integer, dimension(:), allocatable pf_control::pf_control_type::particles`

particles associates with this MPI process

Definition at line 65 of file `pf_control.f90`.

**4.5.2.15 real(kind=kind(1.0d0)), dimension(:,:), allocatable pf\_control::pf\_control\_type::psi**

state vector of ensemble members on this mpi process

Definition at line 42 of file pf\_control.f90.

**4.5.2.16 real(kind=kind(1.0d0)) pf\_control::pf\_control\_type::qscale**

scalar to multiply Q by

Definition at line 49 of file pf\_control.f90.

**4.5.2.17 real(kind=kind(1.0d0)) pf\_control::pf\_control\_type::rho**

enkf inflation factor so that  $P_f = (1 + \rho)P_f$

Definition at line 51 of file pf\_control.f90.

**4.5.2.18 integer, dimension(:,:), allocatable pf\_control::pf\_control\_type::talagrand**

storage for rank histograms

Definition at line 63 of file pf\_control.f90.

**4.5.2.19 real(kind=kind(1.0d0)) pf\_control::pf\_control\_type::time**

dunno

Definition at line 48 of file pf\_control.f90.

**4.5.2.20 integer pf\_control::pf\_control\_type::time\_bwn\_obs**

the number of model timesteps between observations

Definition at line 35 of file pf\_control.f90.

**4.5.2.21 integer pf\_control::pf\_control\_type::time\_obs**

the number of observations we will assimilate

Definition at line 34 of file pf\_control.f90.

**4.5.2.22 integer pf\_control::pf\_control\_type::timestep = 0**

the current timestep as the model progresses

Definition at line 41 of file pf\_control.f90.

**4.5.2.23 character(2) pf\_control::pf\_control\_type::type**

which filter to use currently this has a number of options:

- SE – a stochastic ensemble
- SI – the SIR filter
- ET – the L-ETKF

- EW – the Equivalent Weights particle filter

Definition at line 66 of file `pf_control.f90`.

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

half width of the uniform distribution in mixture density

Definition at line 45 of file `pf_control.f90`.

#### 4.5.2.25 `logical pf_control::pf_control_type::use_mean`

switch if true outputs ensemble mean

Definition at line 59 of file `pf_control.f90`.

#### 4.5.2.26 `logical pf_control::pf_control_type::use_rmse`

switch if true outputs Root Mean Square Errors

Definition at line 62 of file `pf_control.f90`.

#### 4.5.2.27 `logical pf_control::pf_control_type::use_talagrand`

switch if true outputs rank histograms

Definition at line 57 of file `pf_control.f90`.

#### 4.5.2.28 `logical pf_control::pf_control_type::use_traj`

switch if true outputs trajectories

Definition at line 61 of file `pf_control.f90`.

#### 4.5.2.29 `logical pf_control::pf_control_type::use_var`

switch if true outputs ensemble variance

Definition at line 60 of file `pf_control.f90`.

#### 4.5.2.30 `logical pf_control::pf_control_type::use_weak`

switch unused

Definition at line 58 of file `pf_control.f90`.

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

the negative log of the weights of the particles

Definition at line 33 of file `pf_control.f90`.

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

- `src/controllers/pf_control.f90`



## 4.6 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](#)

#### 4.6.1 Detailed Description

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

- the model error covariance matrix

Definition at line 30 of file Qdata.f90.

#### 4.6.2 Member Function/Subroutine Documentation

##### 4.6.2.1 subroutine `qdata::killq` ( )

SUbroutine to deallocate user data for  $Q$

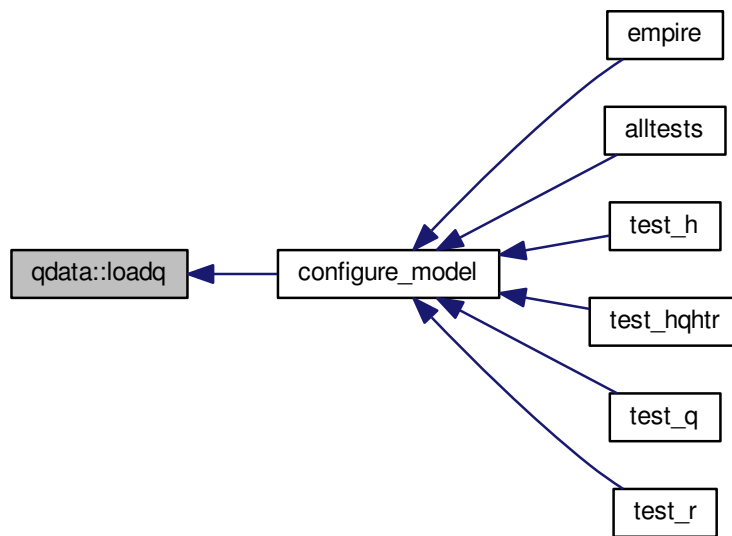
Definition at line 44 of file Qdata.f90.

##### 4.6.2.2 subroutine `qdata::loadq` ( )

Subroutine to load in user data for  $Q$ .

Definition at line 38 of file Qdata.f90.

Here is the caller graph for this function:



### 4.6.3 Member Data Documentation

#### 4.6.3.1 integer, dimension(:), allocatable qdata::qcol

Definition at line 33 of file Qdata.f90.

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

Definition at line 34 of file Qdata.f90.

#### 4.6.3.3 integer qdata::qn

Definition at line 32 of file Qdata.f90.

#### 4.6.3.4 integer qdata::qne

Definition at line 32 of file Qdata.f90.

#### 4.6.3.5 integer, dimension(:), allocatable qdata::qrow

Definition at line 33 of file Qdata.f90.

#### 4.6.3.6 real(kind=kind(1.0d0)) qdata::qscale

Definition at line 35 of file Qdata.f90.

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

Definition at line 34 of file Qdata.f90.

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

- `src/data/Qdata.f90`

## 4.7 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)`

#### 4.7.1 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` Gener-  
alized 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`

Definition at line 22 of file `random_d.f90`.

## 4.7.2 Member Function/Subroutine Documentation

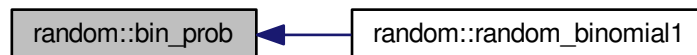
4.7.2.1 `real(kind=kind(1.0d+0)) function random::bin_prob ( integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) p, integer, intent(in) r )`

Definition at line 1000 of file `random_d.f90`.

Here is the call graph for this function:



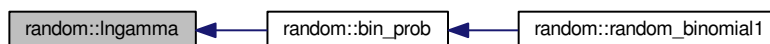
Here is the caller graph for this function:



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

Definition at line 1018 of file `random_d.f90`.

Here is the caller graph for this function:



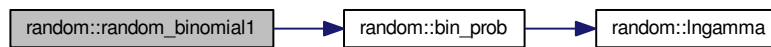
4.7.2.3 `real(kind=kind(1.0d+0)) function random::random_beta ( real(kind=kind(1.0d+0)), intent(in) aa, real(kind=kind(1.0d+0)), intent(in) bb, logical, intent(in) first )`

Definition at line 371 of file `random_d.f90`.

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

Definition at line 923 of file `random_d.f90`.

Here is the call graph for this function:



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

Definition at line 1082 of file `random_d.f90`.

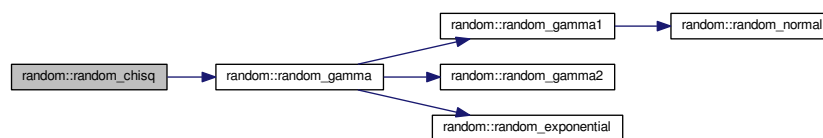
4.7.2.6 real(kind=kind(1.0d+0)) function `random::random_cauchy` ( )

Definition at line 1517 of file `random_d.f90`.

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

Definition at line 308 of file `random_d.f90`.

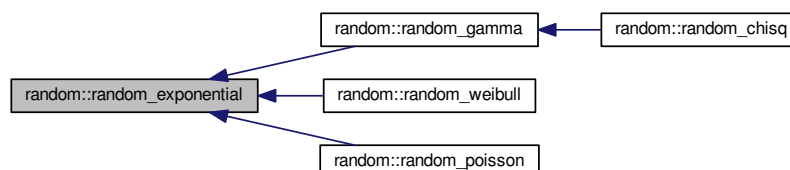
Here is the call graph for this function:



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

Definition at line 324 of file `random_d.f90`.

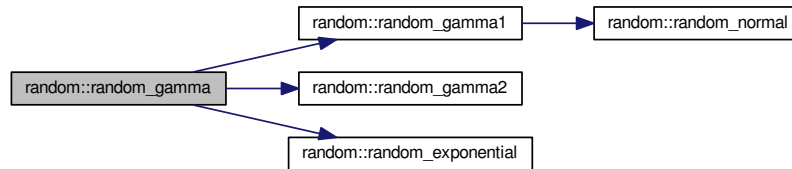
Here is the caller graph for this function:



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

Definition at line 154 of file `random_d.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



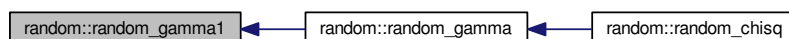
4.7.2.10 `real(kind=kind(1.0d+0)) function random::random_gamma1 ( real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first )`

Definition at line 189 of file `random_d.f90`.

Here is the call graph for this function:



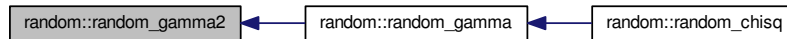
Here is the caller graph for this function:



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

Definition at line 238 of file `random_d.f90`.

Here is the caller graph for this function:



4.7.2.12 `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 )`

Definition at line 610 of file `random_d.f90`.

4.7.2.13 `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 )`

Definition at line 509 of file `random_d.f90`.

Here is the call graph for this function:



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

Definition at line 1314 of file `random_d.f90`.

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

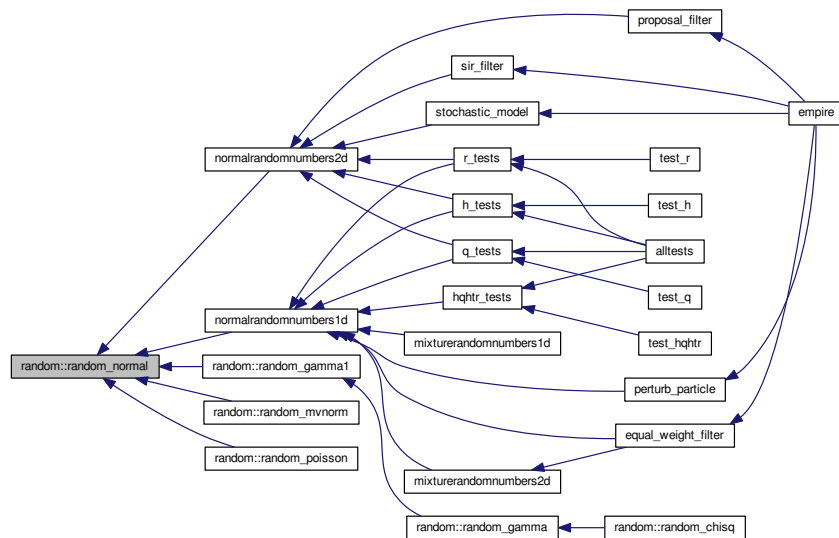
function to get random normal with zero mean and stdev 1

Returns

`fn_val`

Definition at line 108 of file `random_d.f90`.

Here is the caller graph for this function:



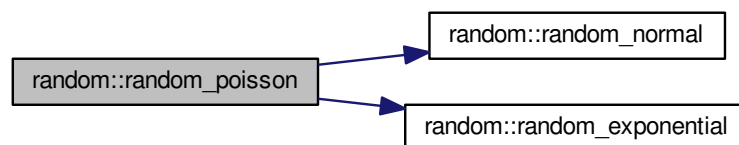
4.7.2.16 subroutine `random::random_order` ( integer, dimension(n), intent(out) *order*, integer, intent(in) *n* )

Definition at line 1539 of file `random_d.f90`.

4.7.2.17 integer function `random::random_poisson` ( real(kind=kind(1.0d+0)), intent(in) *mu*, logical, intent(in) *first* )

Definition at line 681 of file `random_d.f90`.

Here is the call graph for this function:



4.7.2.18 real(kind=kind(1.0d+0)) function `random::random_t` ( integer, intent(in) *m* )

Definition at line 448 of file `random_d.f90`.

4.7.2.19 real(kind=kind(1.0d+0)) function `random::random_von_mises` ( real(kind=kind(1.0d+0)), intent(in) *k*, logical, intent(in) *first* )

Definition at line 1389 of file `random_d.f90`.



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

Definition at line 351 of file random\_d.f90.

Here is the call graph for this function:



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

Definition at line 1573 of file random\_d.f90.

### 4.7.3 Member Data Documentation

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

Definition at line 101 of file random\_d.f90.

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

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

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

### 4.8.1 Detailed Description

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

Definition at line 29 of file Rdata.f90.

## 4.8.2 Member Function/Subroutine Documentation

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

Subroutine to deallocate R data

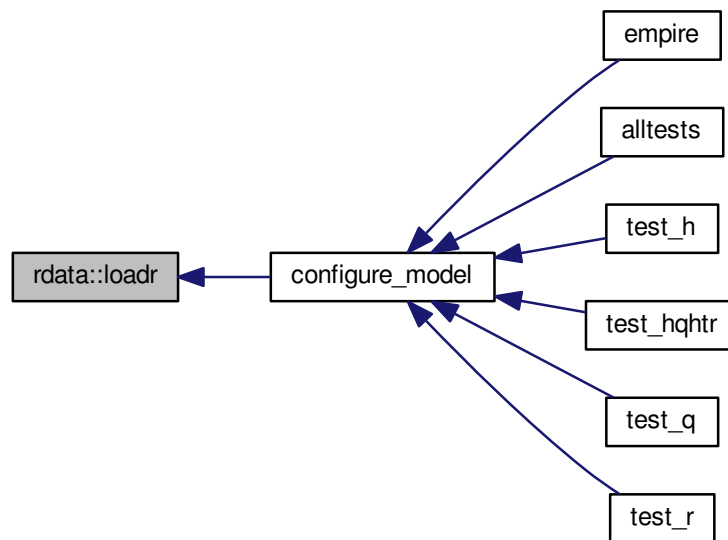
Definition at line 49 of file `Rdata.f90`.

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

Subroutine to load data for R.

Definition at line 36 of file `Rdata.f90`.

Here is the caller graph for this function:



## 4.8.3 Member Data Documentation

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

Definition at line 32 of file `Rdata.f90`.

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

Definition at line 33 of file `Rdata.f90`.

### 4.8.3.3 integer `rdata::rn`

Definition at line 31 of file `Rdata.f90`.

#### 4.8.3.4 integer rdata::rne

Definition at line 31 of file Rdata.f90.

#### 4.8.3.5 integer, dimension(:), allocatable rdata::rrow

Definition at line 32 of file Rdata.f90.

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

Definition at line 33 of file Rdata.f90.

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

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

## 4.9 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*

### 4.9.1 Detailed Description

Module that stores the dimension of observation and state spaces.

Definition at line 29 of file sizes.f90.

### 4.9.2 Member Data Documentation

#### 4.9.2.1 integer sizes::obs\_dim

size of the observation space

Definition at line 31 of file sizes.f90.

#### 4.9.2.2 integer sizes::state\_dim

dimension of the model

Definition at line 32 of file sizes.f90.

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

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



## Chapter 5

# File Documentation

### 5.1 model\_specific.f90 File Reference

#### Functions/Subroutines

- subroutine `configure_model`  
*subroutine called initially to set up details and data for model specific functions*
- subroutine `solve_r` (obsDim, nrhs, y, v, t)  
*subroutine to take an observation vector y and return v in observation space.*
- subroutine `solve_rhalf` (obsdim, nrhs, y, v, t)  
*subroutine to take an observation vector y and return v in observation space.*
- subroutine `solve_hqht_plus_r` (obsdim, y, v, t)  
*subroutine to take an observation vector y and return v in observation space.*
- subroutine `q` (nrhs, x, Qx)  
*subroutine to take a full state vector x and return Qx in state space.*
- subroutine `qhalf` (nrhs, x, Qx)  
*subroutine to take a full state vector x and return  $Q^{1/2}x$  in state space.*
- subroutine `r` (obsDim, nrhs, y, Ry, t)  
*subroutine to take an observation vector x and return Rx in observation space.*
- subroutine `rhalf` (obsDim, nrhs, y, Ry, t)  
*subroutine to take an observation vector x and return Rx in observation space.*
- subroutine `h` (obsDim, nrhs, x, hx, t)  
*subroutine to take a full state vector x and return H(x) in observation space.*
- subroutine `ht` (obsDim, nrhs, y, x, t)  
*subroutine to take an observation vector y and return  $x = H^T(y)$  in full state space.*
- subroutine `dist_st_ob` (xp, yp, dis, t)  
*subroutine to compute the distance between the variable in the state vector and the variable in the observations*

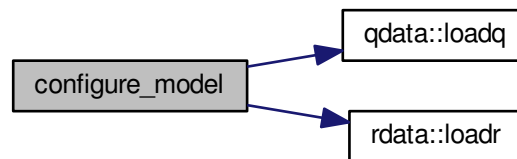
#### 5.1.1 Function/Subroutine Documentation

##### 5.1.1.1 subroutine `configure_model` ( )

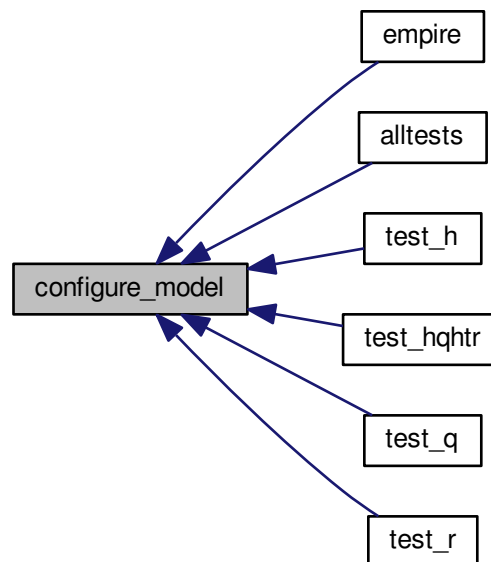
subroutine called initially to set up details and data for model specific functions

Definition at line 30 of file model\_specific.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.1.2 subroutine `dist_st_ob` ( integer, intent(in) *xp*, integer, intent(in) *yp*, real(kind=kind(1.0d0)), intent(out) *dis*, integer, intent(in) *t* )

subroutine to compute the distance between the variable in the state vector and the variable in the observations

Compute  $\text{dist}(x(xp), y(yp))$

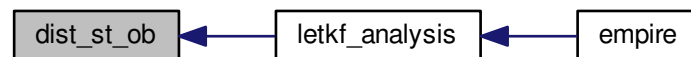
Parameters

<i>in</i>	<i>xp</i>	the index in the state vector
-----------	-----------	-------------------------------

in	<i>yp</i>	the index in the observation vector
out	<i>dis</i>	the distance between $x(xp)$ and $y(yp)$
in	<i>t</i>	the current time index for observations

Definition at line 270 of file model\_specific.f90.

Here is the caller graph for this function:



**5.1.1.3** subroutine *h* ( integer, intent(in) *obsDim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(state\_dim,nrhs), intent(in) *x*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *hx*, integer, intent(in) *t* )

subroutine to take a full state vector  $x$  and return  $H(x)$  in observation space.

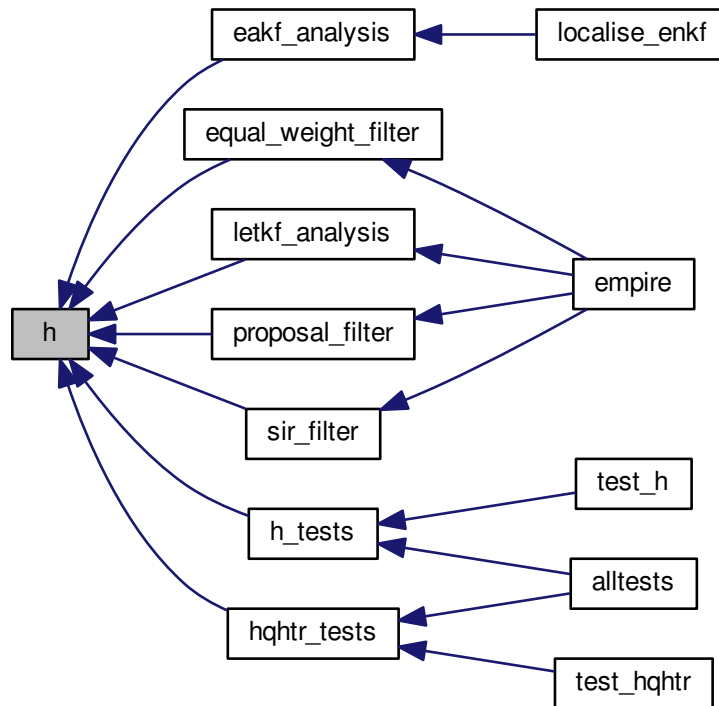
Given  $x$  compute  $Hx$

#### Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>x</i>	the input vectors in state space
out	<i>hx</i>	the resulting vector in observation space where $hx = Hx$
in	<i>t</i>	the timestep

Definition at line 224 of file model\_specific.f90.

Here is the caller graph for this function:



5.1.1.4 subroutine `ht` ( integer, intent(in) *obsDim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(*obsdim*,*nrhs*), intent(in) *y*, real(kind=rk), dimension(*state\_dim*,*nrhs*), intent(out) *x*, integer, intent(in) *t* )

subroutine to take an observation vector  $y$  and return  $x = H^T(y)$  in full state space.

Given  $y$  compute  $x = H^T(y)$

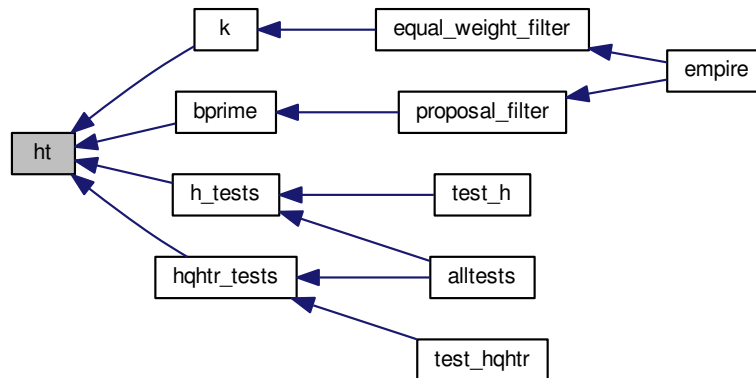
#### Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>y</i>	the input vectors in observation space
out	<i>x</i>	the resulting vector in state space where $x = H^T y$
in	<i>t</i>	the timestep

Definition at line 247 of file `model_specific.f90`.



Here is the caller graph for this function:



5.1.1.5 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* )

subroutine to take a full state vector *x* and return *Qx* in state space.

Given *x* compute  $Qx$

Parameters

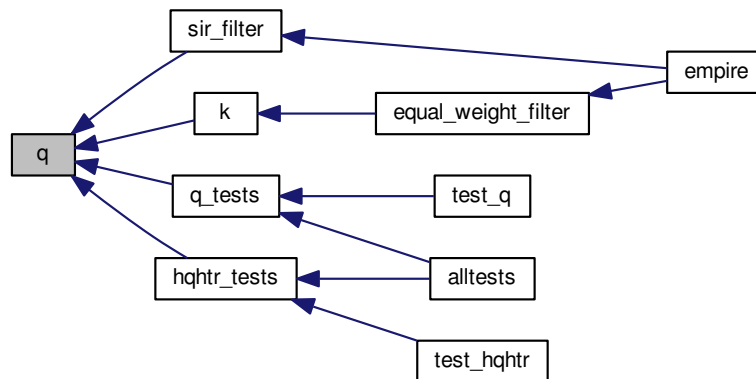
in	<i>nrhs</i>	the number of right hand sides
in	<i>x</i>	the input vector
out	<i>qx</i>	the resulting vector where $Qx = Qx$

Definition at line 134 of file model\_specific.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.1.6 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 )

subroutine to take a full state vector  $x$  and return  $Q^{1/2}x$  in state space.

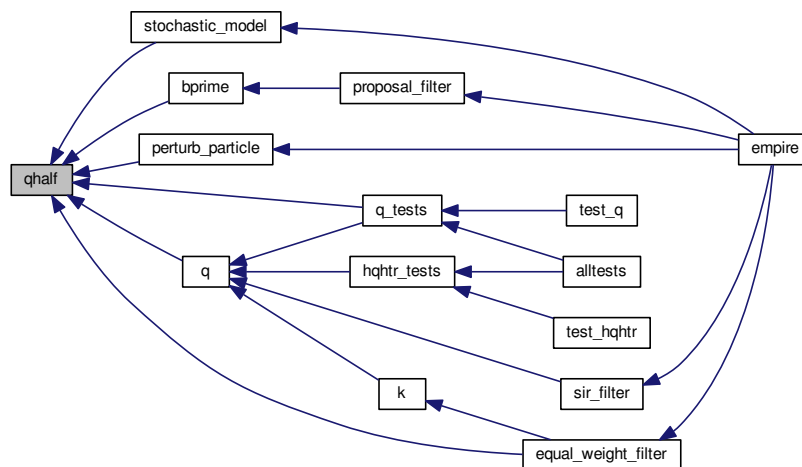
Given  $x$  compute  $Q^{\frac{1}{2}}x$

#### Parameters

in	nrhs	the number of right hand sides
in	x	the input vector
out	qx	the resulting vector where $Qx = Q^{\frac{1}{2}}x$

Definition at line 159 of file model\_specific.f90.

Here is the caller graph for this function:



5.1.1.7 subroutine *r* ( integer, intent(in) *obsDim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(obsdim,nrhs), intent(in) *y*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *Ry*, integer, intent(in) *t* )

subroutine to take an observation vector *x* and return *Rx* in observation space.

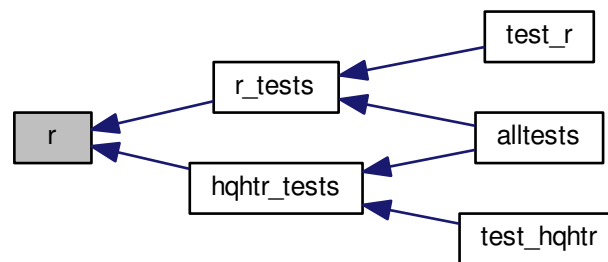
Given *y* compute *Ry*

Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>y</i>	the input vector
out	<i>ry</i>	the resulting vectors where $Ry = Ry$
in	<i>t</i>	the timestep

Definition at line 179 of file model\_specific.f90.

Here is the caller graph for this function:



5.1.1.8 subroutine *rhalf* ( integer, intent(in) *obsDim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(obsdim,nrhs), intent(in) *y*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *Ry*, integer, intent(in) *t* )

subroutine to take an observation vector *x* and return *Rx* in observation space.

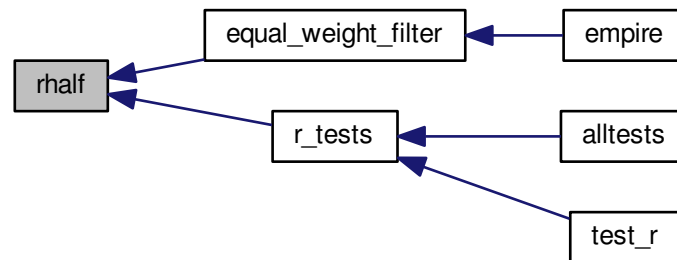
Given *y* compute  $R^{\frac{1}{2}}y$

Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>y</i>	the input vector
out	<i>ry</i>	the resulting vector where $Ry = R^{\frac{1}{2}}y$
in	<i>t</i>	the timestep

Definition at line 201 of file model\_specific.f90.

Here is the caller graph for this function:



**5.1.1.9** subroutine `solve_hqht_plus_r` ( integer, intent(in) *obsdim*, real(kind=rk), dimension(obsdim), intent(in) *y*, real(kind=rk), dimension(obsdim), intent(out) *v*, integer, intent(in) *t* )

subroutine to take an observation vector *y* and return *v* in observation space.

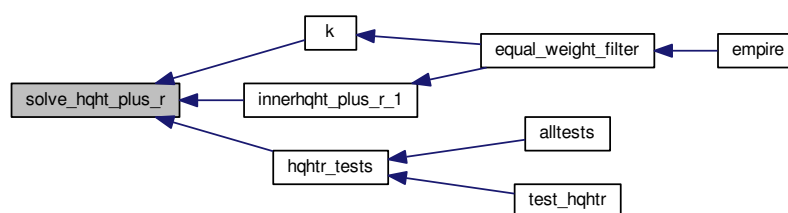
Given *y* find *v* such that  $(HQH^T + R)v = y$

**Parameters**

in	<i>obsdim</i>	the dimension of the observations
in	<i>y</i>	the input vector
out	<i>v</i>	the result where $v = (HQH^T + R)^{-1}y$
in	<i>t</i>	the timestep

Definition at line 114 of file `model_specific.f90`.

Here is the caller graph for this function:



**5.1.1.10** subroutine `solve_r` ( integer, intent(in) *obsDim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(obsdim,nrhs), intent(in) *y*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *v*, integer, intent(in) *t* )

subroutine to take an observation vector *y* and return *v* in observation space.

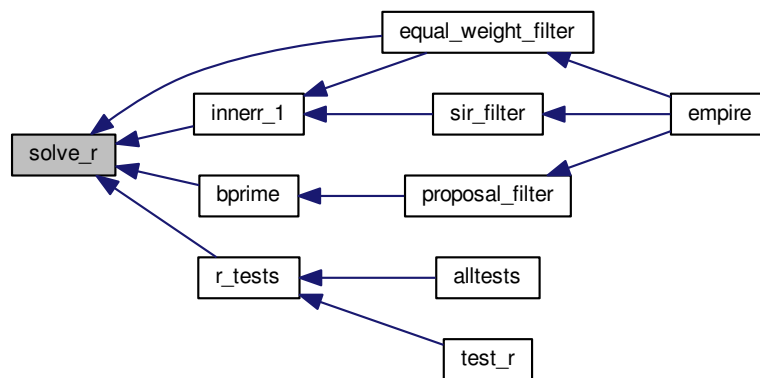
Given *y* find *v* such that  $Rv = y$

## Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>y</i>	input vector
out	<i>v</i>	result vector where $v = R^{-1}y$
in	<i>t</i>	the timestep

Definition at line 72 of file model\_specific.f90.

Here is the caller graph for this function:



5.1.1.11 subroutine `solve_half` ( integer, intent(in) *obsdim*, integer, intent(in) *nrhs*, real(kind=rk), dimension(*obsdim*,*nrhs*), intent(in) *y*, real(kind=rk), dimension(*obsdim*,*nrhs*), intent(out) *v*, integer, intent(in) *t* )

subroutine to take an observation vector *y* and return *v* in observation space.

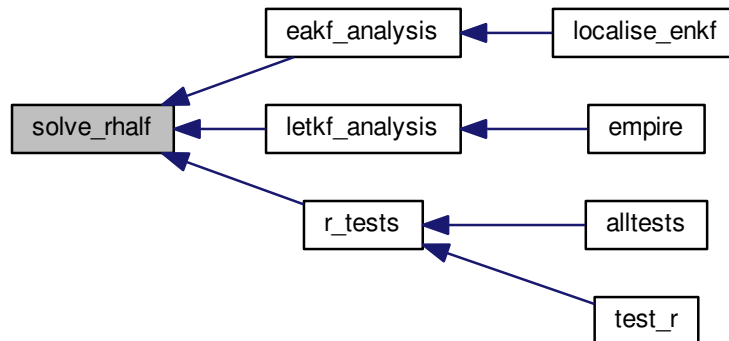
Given *y* find *v* such that  $R^{\frac{1}{2}}v = y$

## Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides
in	<i>y</i>	input vector
out	<i>v</i>	result vector where $v = R^{-\frac{1}{2}}y$
in	<i>t</i>	the timestep

Definition at line 92 of file model\_specific.f90.

Here is the caller graph for this function:



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

### Data Types

- module [pf\\_control](#)

*module [pf\\_control](#) holds all the information to control the the main program*

- type [pf\\_control::pf\\_control\\_type](#)

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

### Functions/Subroutines

- program [empire](#)

*the main program*

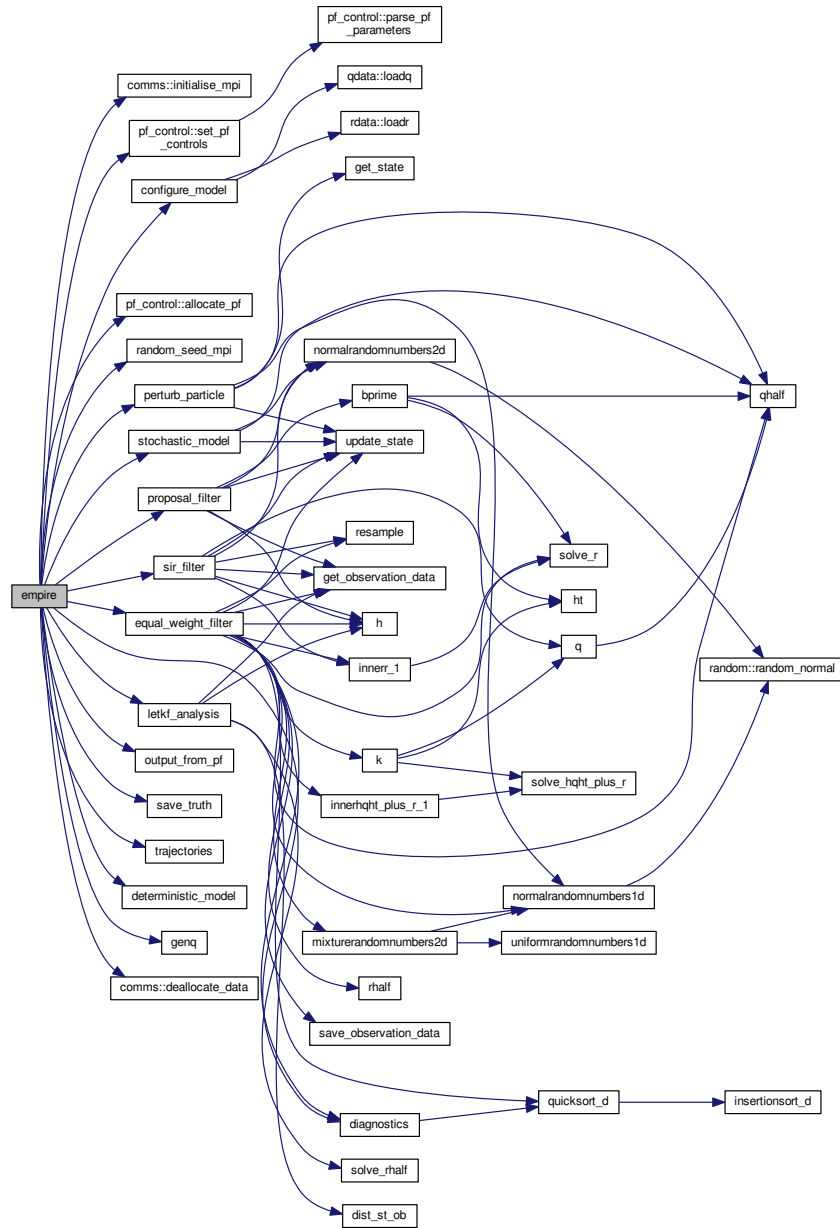
### 5.3.1 Function/Subroutine Documentation

#### 5.3.1.1 program [empire](#) ( )

the main program

Definition at line 37 of file `pf_couple.f90`.

Here is the call graph for this function:



## 5.4 src/controllers/pf\_parameters.dat File Reference

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

### Data Types

- module [sizes](#)

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

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

### Data Types

- module [qdata](#)  
*Module as a place to store user specified data for Q.*

## 5.7 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](#)

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

## 5.9 src/filters/deterministic\_model.f90 File Reference

### Functions/Subroutines

- subroutine [deterministic\\_model](#)  
*subroutine to simply move the model forward in time one timestep PAB 21-05-2013*

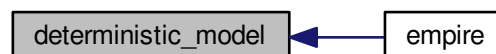
### 5.9.1 Function/Subroutine Documentation

#### 5.9.1.1 subroutine `deterministic_model` ( )

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

Definition at line 32 of file `deterministic_model.f90`.

Here is the caller graph for this function:



## 5.10 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)

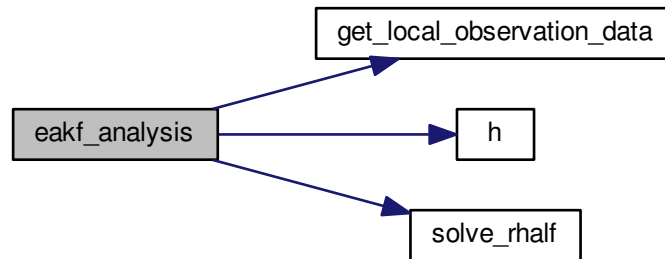


### 5.10.1 Function/Subroutine Documentation

5.10.1.1 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* )

Definition at line 27 of file `eakf_analysis.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

### Functions/Subroutines

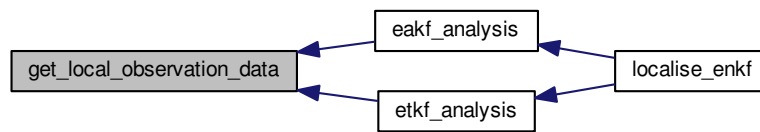
- subroutine `h_local` (*num\_hor*, *num\_ver*, *this\_hor*, *this\_ver*, *boundary*, *nrhs*, *stateDim*, *x*, *obsDim*, *y*)
- subroutine `solve_rhalf_local` (*num\_hor*, *num\_ver*, *this\_hor*, *this\_ver*, *boundary*, *nrhs*, *obsDim*, *y*, *v*)
- subroutine `get_local_observation_data` (*num\_hor*, *num\_ver*, *this\_hor*, *this\_ver*, *boundary*, *obsDim*, *y*)
- subroutine `localise_enkf` (*enkf\_analysis*)

### 5.11.1 Function/Subroutine Documentation

5.11.1.1 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* )

Definition at line 83 of file `enkf_specific.f90`.

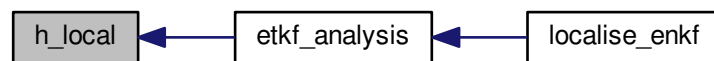
Here is the caller graph for this function:



5.11.1.2 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* )

Definition at line 27 of file `enkf_specific.f90`.

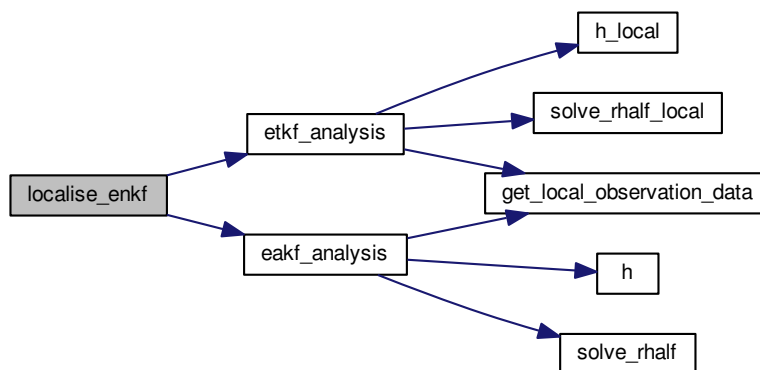
Here is the caller graph for this function:



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

Definition at line 142 of file `enkf_specific.f90`.

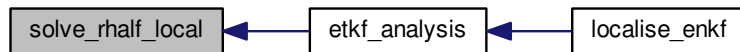
Here is the call graph for this function:



5.11.1.4 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* )

Definition at line 69 of file `enkf_specific.f90`.

Here is the caller graph for this function:



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

### Functions/Subroutines

- subroutine [equal\\_weight\\_filter](#)

*subroutine to do the equivalent weights step*

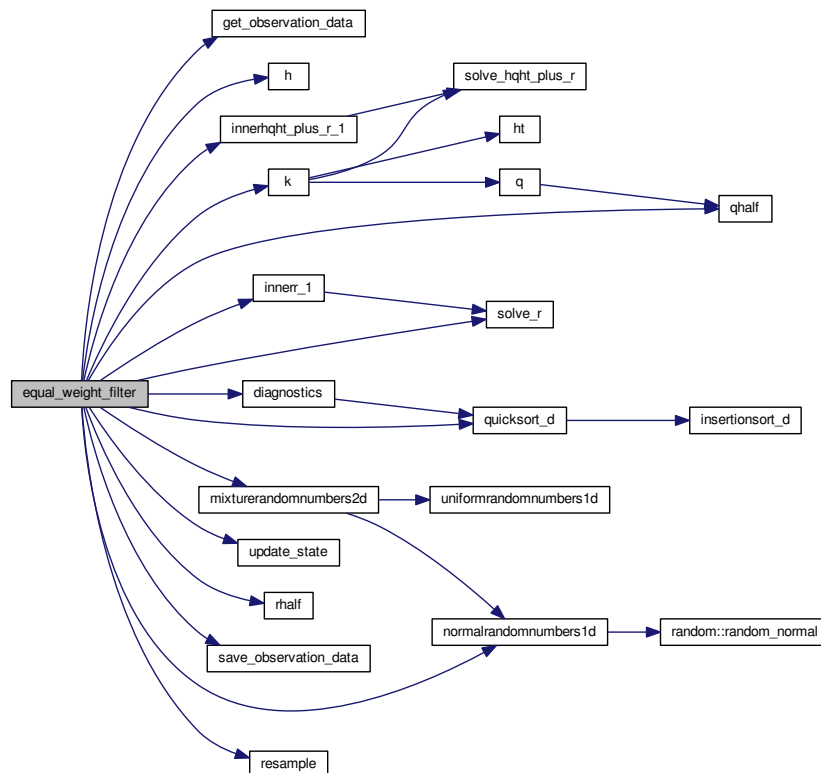
### 5.12.1 Function/Subroutine Documentation

5.12.1.1 subroutine `equal_weight_filter` ( )

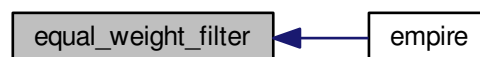
subroutine to do the equivalent weights step

Definition at line 29 of file `equivalent_weights_step.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.13 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)  
*subroutine to perform the ensemble transform Kalman filter*

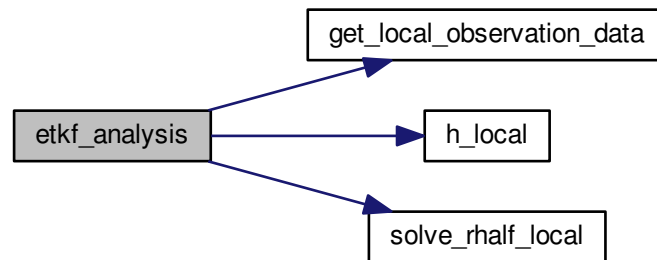
#### 5.13.1 Function/Subroutine Documentation

5.13.1.1 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* )

subroutine to perform the ensemble transform Kalman filter

Definition at line 34 of file `etkf_analysis.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.14 src/filters/letkf\_analysis.f90 File Reference

### Functions/Subroutines

- subroutine [letkf\\_analysis](#)

*subroutine to perform the ensemble transform Kalman filter as part of L-ETKF*

### 5.14.1 Function/Subroutine Documentation

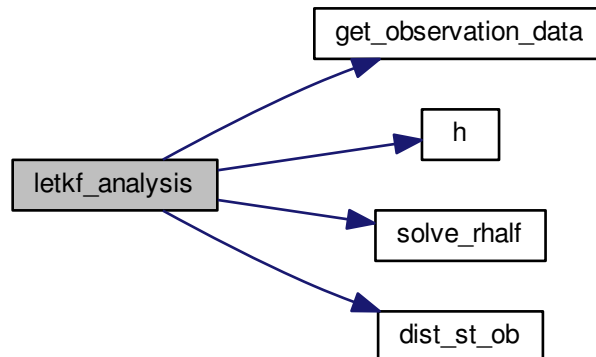
5.14.1.1 subroutine `letkf_analysis` ( )

subroutine to perform the ensemble transform Kalman filter as part of L-ETKF

The observation

Definition at line 35 of file `letkf_analysis.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

### Functions/Subroutines

- subroutine [proposal\\_filter](#)

*Subroutine to perform nudging in the proposal step of EWPF.*

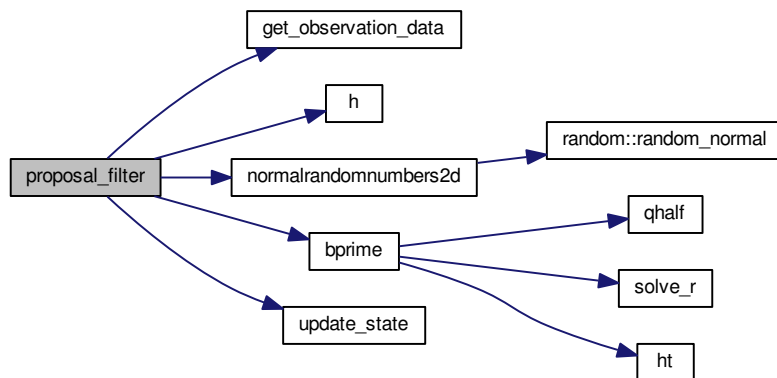
### 5.15.1 Function/Subroutine Documentation

#### 5.15.1.1 subroutine `proposal_filter` ( )

Subroutine to perform nudging in the proposal step of EWPF.

Definition at line 33 of file `proposal_filter.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

### Functions/Subroutines

- subroutine [sir\\_filter](#)

*Subroutine to perform SIR filter (Sequential Importance Resampling)*

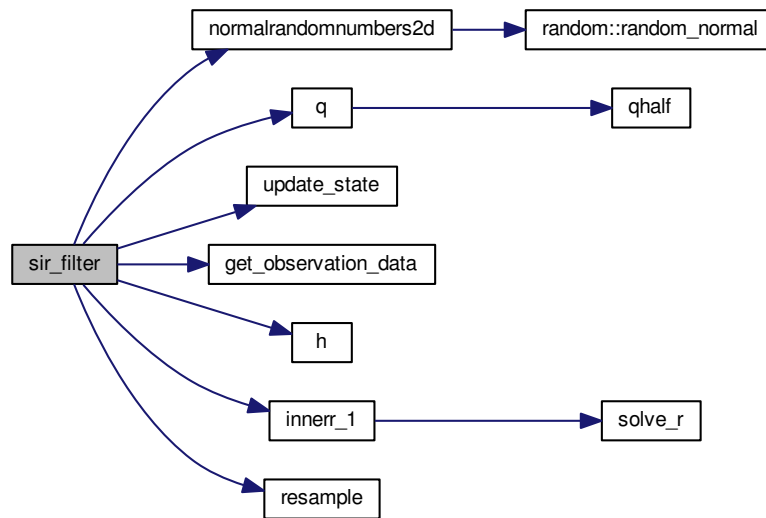
### 5.16.1 Function/Subroutine Documentation

#### 5.16.1.1 subroutine `sir_filter` ( )

Subroutine to perform SIR filter (Sequential Importance Resampling)

Definition at line 28 of file `sir_filter.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.17 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)

### 5.17.1 Function/Subroutine Documentation

5.17.1.1 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 )

Definition at line 80 of file stochastic\_model.f90.

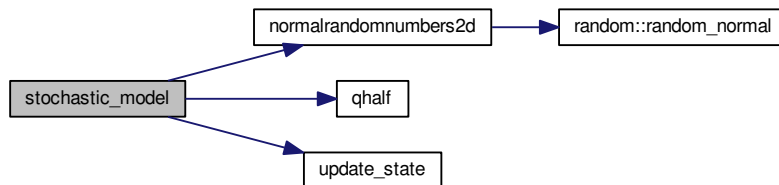


## 5.17.1.2 subroutine stochastic\_model ( )

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

Definition at line 32 of file stochastic\_model.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.18 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.*

## 5.18.1 Function/Subroutine Documentation

5.18.1.1 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* )

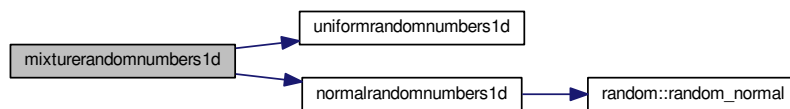
generate one dimensional vector drawn from mixture density

## 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

Definition at line 90 of file gen\_rand.f90.

Here is the call graph for this function:



**5.18.1.2** 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* )

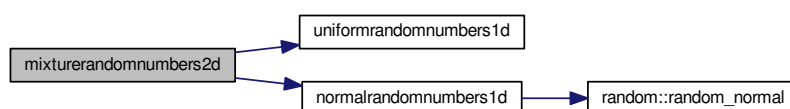
generate two dimensional vector, each drawn from mixture density

## Parameters

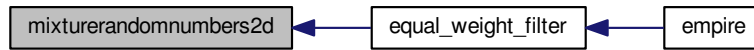
in	<i>mean</i>	Mean of normal distribution
in	<i>stdev</i>	Standard deviation of normal distribution
in	<i>ufac</i>	half-width of uniform distribution that is centered on the mean
in	<i>epsi</i>	Proportion controlling mixture draw. if random_number > epsi then draw from uniform, else normal
in	<i>n</i>	first dimension of output vector
in	<i>k</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

Definition at line 125 of file gen\_rand.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.3 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

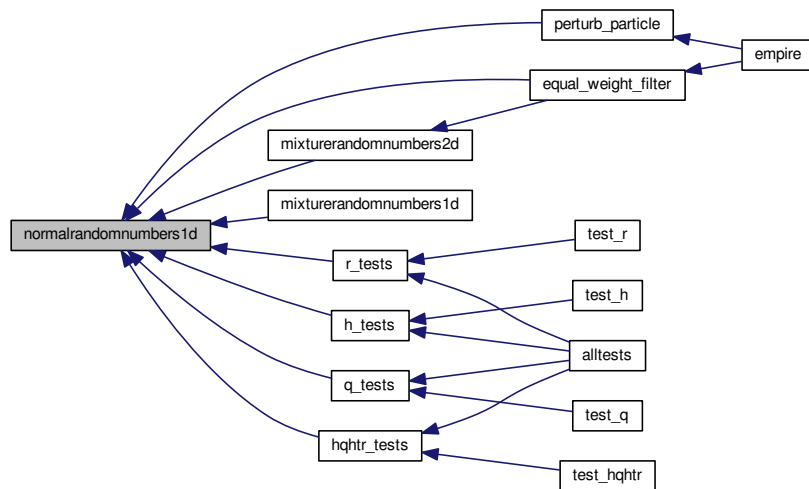
in	<i>n</i>	n size of output vector
in	<i>mean</i>	mean mean of normal distribution
in	<i>stdev</i>	stdev Standard Deviation of normal distribution
out	<i>phi</i>	phi n dimensional normal random numbers

Definition at line 43 of file `gen_rand.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



**5.18.1.4** 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

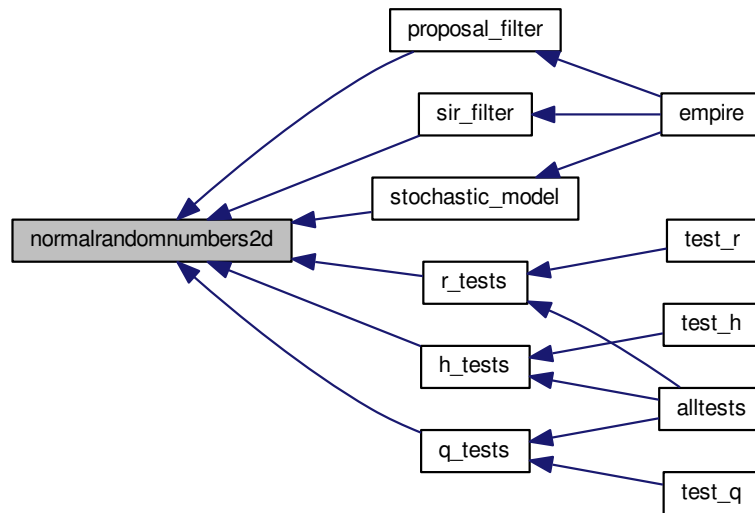
in	<i>n</i>	n first dimension of output vector
in	<i>k</i>	k second dimension of output vector
in	<i>mean</i>	mean mean of normal distribution
in	<i>stdev</i>	stdev Standard Deviation of normal distribution
out	<i>phi</i>	phi n,k dimensional normal random numbers

Definition at line 60 of file `gen_rand.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.18.1.5 subroutine `random_seed_mpi` ( integer, intent(in) *pfid* )

Subroutine to set the random seed across MPI threads.

##### Parameters

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

Definition at line 151 of file `gen_rand.f90`.

Here is the caller graph for this function:



#### 5.18.1.6 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* )

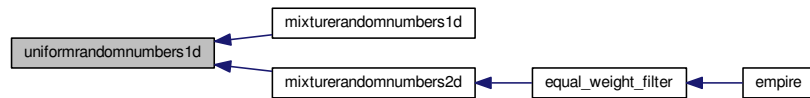
generate one dimension of uniform random numbers

## Parameters

in	$n$	n size of output vector
in	$minv$	minv minimum value of uniform distribution
in	$maxv$	maxv maximum value of uniform distribution
out	$\phi$	$\phi$ n dimensional uniform random numbers

Definition at line 28 of file gen\_rand.f90.

Here is the caller graph for this function:



## 5.19 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  $K := QH^T(HQH^T + R)^{-1}$ .
- subroutine **inner\_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

## 5.19.1 Function/Subroutine Documentation

**5.19.1.1** 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)  $betan$  )

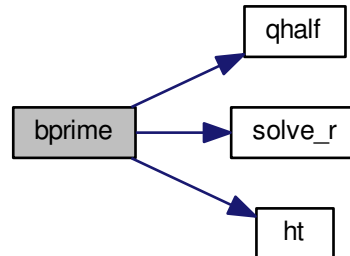
subroutine to calculate nudging term and correlated random errors efficiently

## Parameters

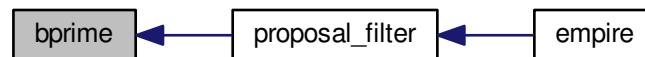
in	$y$	(obs_dim,pf%count) vectors of innovations $y - H(x^{n-1})$
out	$x$	(state_dim,pf%count) vectors of $pH^T R^{-1}[y - H(x^{n-1})]$
out	$QHtR\_1y$	(state_dim,pf%count) vectors of $pQH^T R^{-1}[y - H(x^{n-1})]$
in	$normaln$	(state_dim,pf%count) uncorrelated random vectors such that $normaln(:,i) \sim \mathcal{N}(0, I)$
out	$betan$	(state_dim,pf%count) correlated random vectors such that $betan(:,i) \sim \mathcal{N}(0, Q)$

Definition at line 155 of file operator\_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



5.19.1.2 subroutine `innerhght_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

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

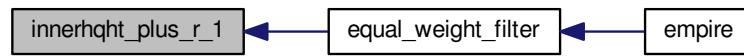
Definition at line 91 of file `operator_wrappers.f90`.

Here is the call graph for this function:





Here is the caller graph for this function:



**5.19.1.3** 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

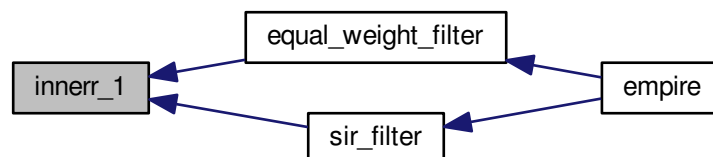
<code>in</code>	<code>y</code>	multiple vectors in observation space (pf%count of them)
<code>out</code>	<code>w</code>	multiple scalars (pf%count) where $w(i)$ has the value $y(:,i)^T R^{-1} y(:,i)$

Definition at line 65 of file `operator_wrappers.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

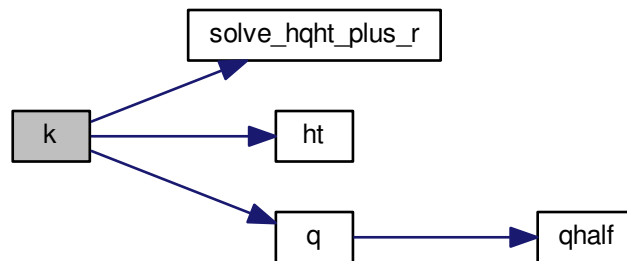
Subroutine to apply  $K$  to a vector  $y$  in observation space where  $K := QH^T(HQH^T + R)^{-1}$ .

## Parameters

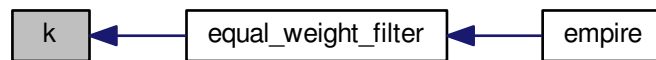
in	y	vector in observation space
out	x	vector in state space

Definition at line 32 of file operator\_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.20 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, fpsi, kgain, betan)  
Subroutine to update the state.

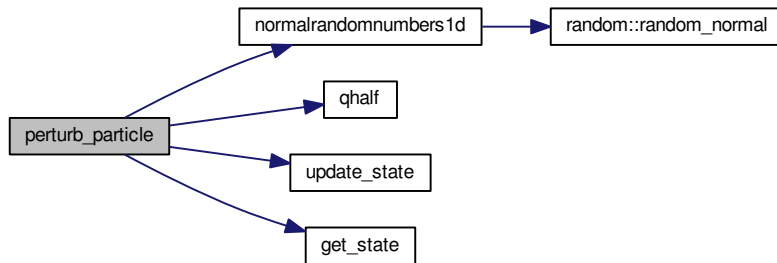
### 5.20.1 Function/Subroutine Documentation

#### 5.20.1.1 subroutine perturb\_particle ( real(kind=rk), dimension(state\_dim), intent(inout) x )

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

Definition at line 30 of file perturb\_particle.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



**5.20.1.2** 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.

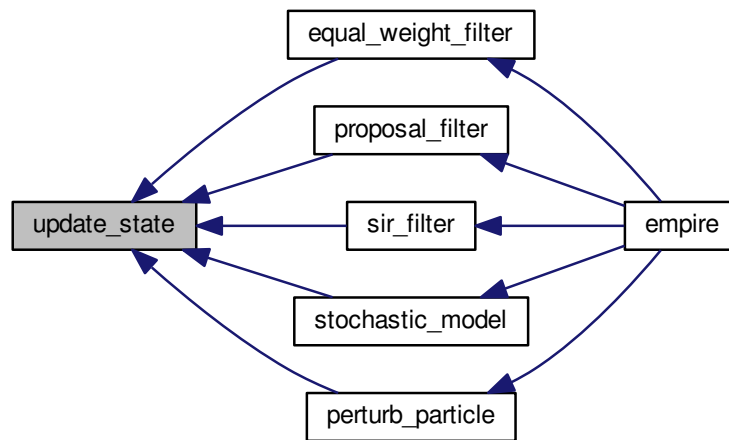
This can be changed for the specific model if it needs to be

#### 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

Definition at line 95 of file `perturb_particle.f90`.

Here is the caller graph for this function:



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

### Functions/Subroutines

- subroutine [resample](#)  
*Subroutine to perform Universal Importance Resampling.*

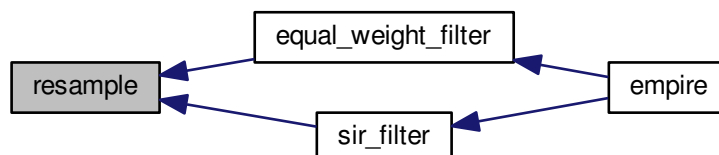
#### 5.21.1 Function/Subroutine Documentation

##### 5.21.1.1 subroutine resample ( )

Subroutine to perform Universal Importance Resampling.

Definition at line 28 of file `resample.f90`.

Here is the caller graph for this function:



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

### Functions/Subroutines

- program [alltests](#)  
*program to run all tests of user specific functions*

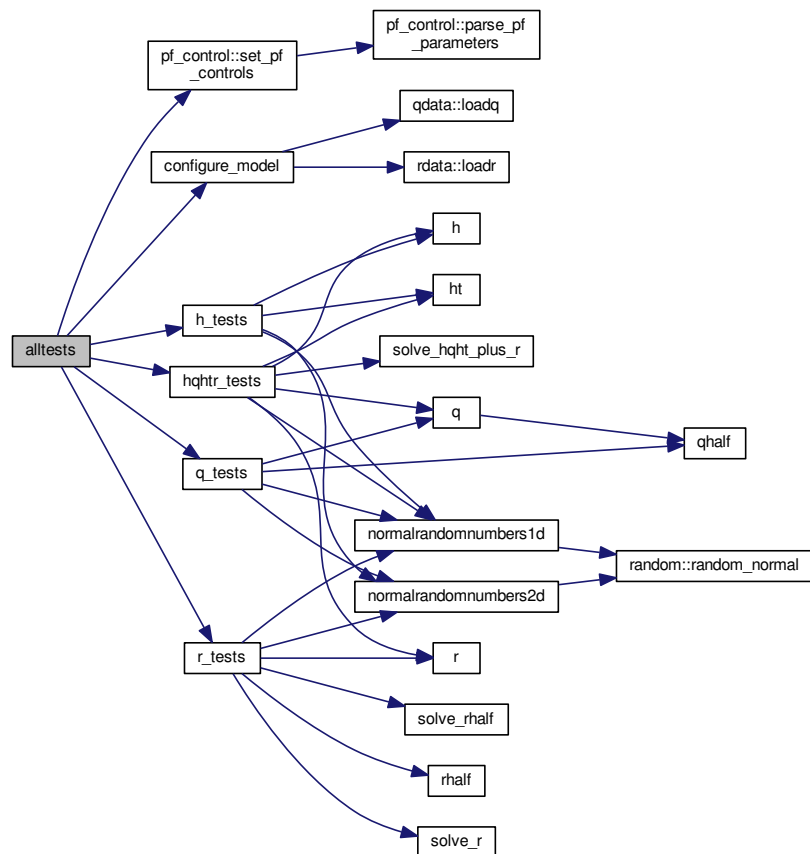
#### 5.22.1 Function/Subroutine Documentation

##### 5.22.1.1 program alltests ( )

program to run all tests of user specific functions

Definition at line 31 of file alltests.f90.

Here is the call graph for this function:



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

### Functions/Subroutines

- program [test\\_h](#)  
*program to run tests of user supplied observation operator*

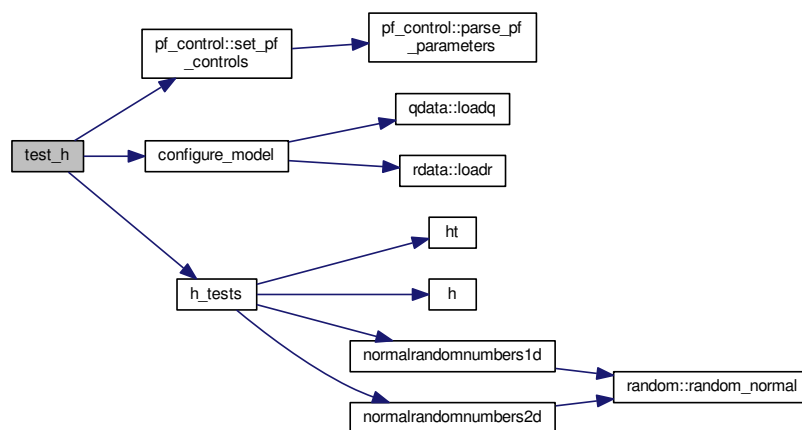
### 5.23.1 Function/Subroutine Documentation

#### 5.23.1.1 program test\_h ( )

program to run tests of user supplied observation operator

Definition at line 31 of file test\_h.f90.

Here is the call graph for this function:



## 5.24 src/tests/test\_hqhtr.f90 File Reference

### Functions/Subroutines

- program [test\\_hqhtr](#)

*program to run tests of user supplied linear solve*

### 5.24.1 Function/Subroutine Documentation

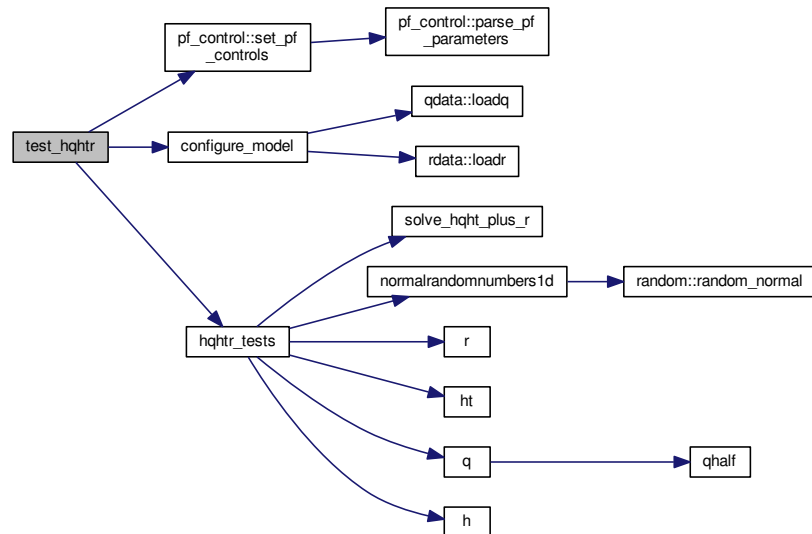
#### 5.24.1.1 program test\_hqhtr ( )

program to run tests of user supplied linear solve

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

Definition at line 33 of file test\_hqhtr.f90.

Here is the call graph for this function:



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

### Functions/Subroutines

- program [test\\_q](#)

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

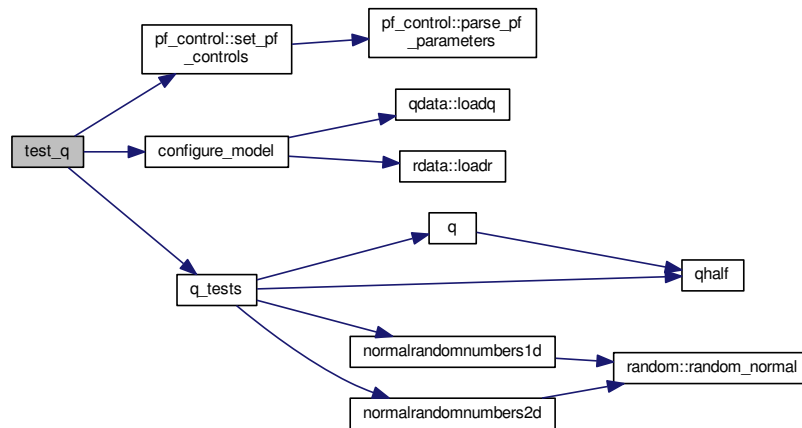
### 5.25.1 Function/Subroutine Documentation

#### 5.25.1.1 program test\_q ( )

program to run tests of user supplied model error covariance matrix

Definition at line 31 of file test\_q.f90.

Here is the call graph for this function:



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

### Functions/Subroutines

- program [test\\_r](#)

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

### 5.26.1 Function/Subroutine Documentation

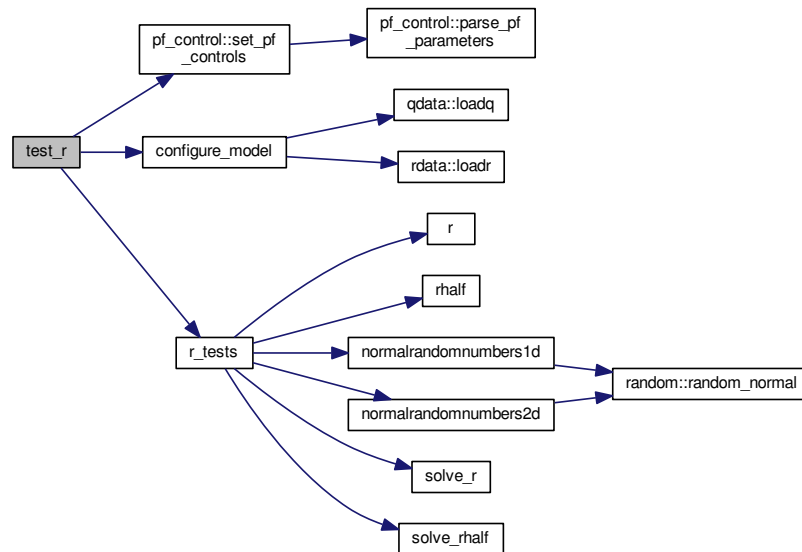
#### 5.26.1.1 program `test_r` ( )

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

Definition at line 31 of file `test_r.f90`.



Here is the call graph for this function:



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

### Functions/Subroutines

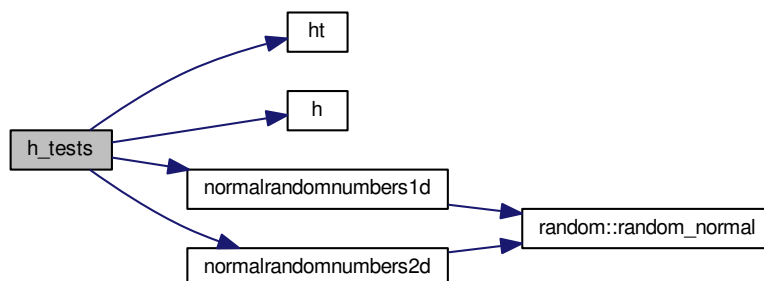
- subroutine [h\\_tests](#) ()
- subroutine [r\\_tests](#) ()
- subroutine [q\\_tests](#) ()
- subroutine [hqhtr\\_tests](#) ()

### 5.27.1 Function/Subroutine Documentation

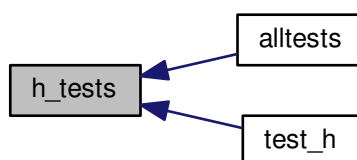
#### 5.27.1.1 subroutine `h_tests` ( )

These are some tests to check that the observation operator is implemented correctly  
Definition at line 27 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



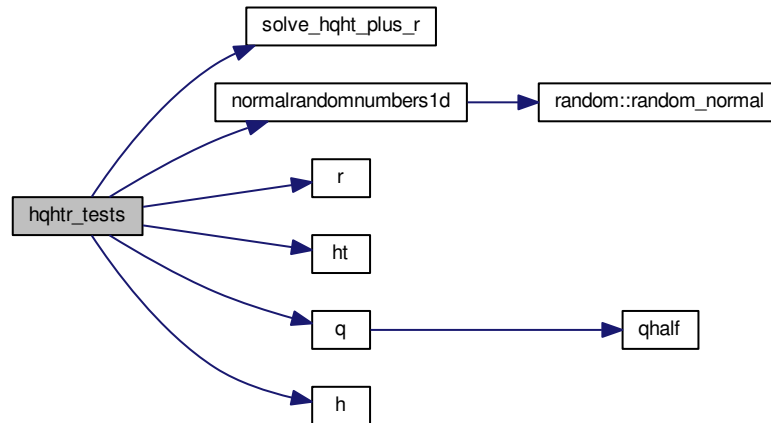
#### 5.27.1.2 subroutine hqhtr\_tests ( )

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

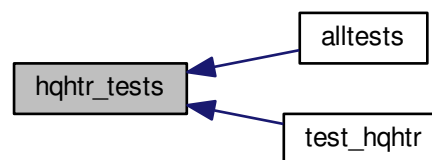
This should check the operation  $(HQH^T + R)^{-1}$  is working

Definition at line 876 of file tests.f90.

Here is the call graph for this function:



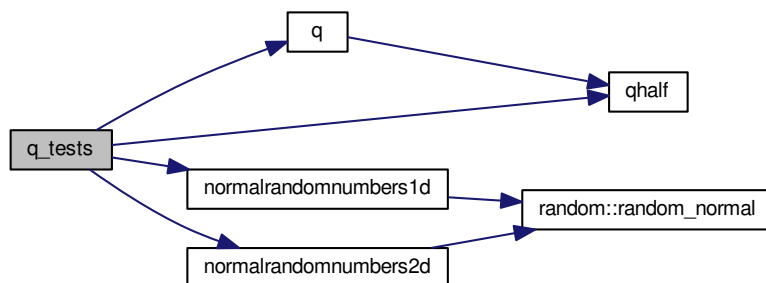
Here is the caller graph for this function:



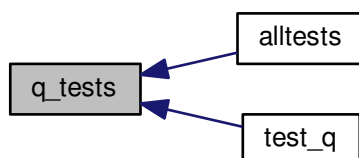
### 5.27.1.3 subroutine q\_tests ( )

These are some tests to check that the model error covariance matrix is implemented correctly  
 Definition at line 672 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

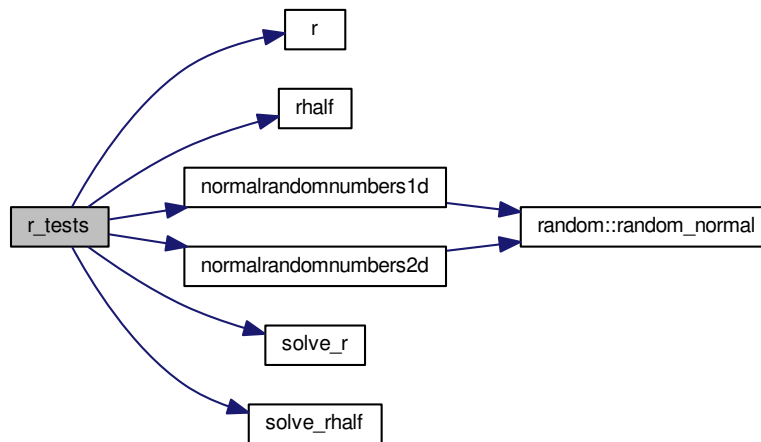


#### 5.27.1.4 subroutine `r_tests` ( )

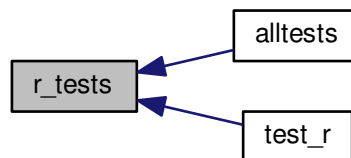
These are some tests to check that the observation error covariance matrix is implemented correctly

Definition at line 254 of file `tests.f90`.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.28 src/utils/comms.f90 File Reference

### Data Types

- module [comms](#)  
*Module containing EMPIRE coupling data.*

## 5.29 src/utils/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*

- subroutine `save_state` (state, filename)

*subroutine to save the state vector to a named file as an unformatted fortran file*

- subroutine `get_state` (state, filename)

*subroutine to write the state vector to a named file as an unformatted fortran file*

## 5.29.1 Function/Subroutine Documentation

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

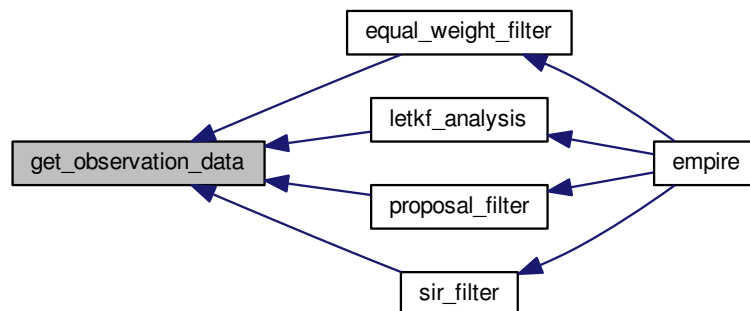
Subroutine to read observation from a file  
Uses pftimestep to determine which observation to read.

#### Parameters

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

Definition at line 32 of file `data_io.f90`.

Here is the caller graph for this function:



### 5.29.1.2 subroutine `get_state` ( real(kind=rk), dimension(state\_dim), intent(out) state, character(14), intent(in) filename )

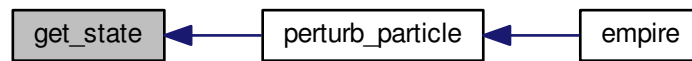
subroutine to write the state vector to a named file as an unformatted fortran file

#### Parameters

out	state	the state vector
in	filename	the name of the file to write the state vector in

Definition at line 283 of file `data_io.f90`.

Here is the caller graph for this function:



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

subroutine to output data from the filter

Definition at line 124 of file data\_io.f90.

Here is the caller graph for this function:



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

Subroutine to save observation to a file

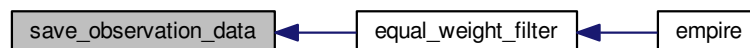
Uses `pftimestep` to determine which observation to save.

##### Parameters

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

Definition at line 60 of file data\_io.f90.

Here is the caller graph for this function:



#### 5.29.1.5 subroutine save\_state ( real(kind=rk), dimension(state\_dim), intent(in) state, character(14), intent(in) filename )

subroutine to save the state vector to a named file as an unformatted fortran file

## Parameters

<i>in</i>	<i>state</i>	the state vector
<i>in</i>	<i>filename</i>	the name of the file to save the state vector in

Definition at line 257 of file data\_io.f90.

#### 5.29.1.6 subroutine save\_truth ( real(kind=rk), dimension(state\_dim), intent(in) x )

Subroutine to save truth to a file

.

## Parameters

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

Definition at line 98 of file data\_io.f90.

Here is the caller graph for this function:



## 5.30 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

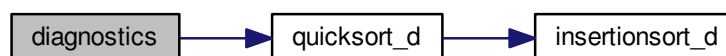
#### 5.30.1 Function/Subroutine Documentation

##### 5.30.1.1 subroutine diagnostics ( )

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

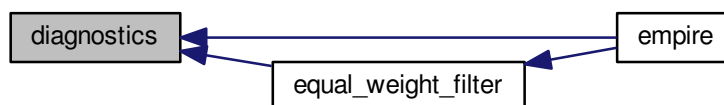
Definition at line 31 of file diagnostics.f90.

Here is the call graph for this function:





Here is the caller graph for this function:



#### 5.30.1.2 subroutine trajectories ( )

subroutine to output trajectories

Definition at line 203 of file diagnostics.f90.

Here is the caller graph for this function:



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

### Functions/Subroutines

- subroutine [genq](#)

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

#### 5.31.1 Function/Subroutine Documentation

##### 5.31.1.1 subroutine genq ( )

Subroutine to estimate Q from a long model run.

Definition at line 28 of file genQ.f90.

Here is the caller graph for this function:



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

### Data Types

- module [histogram\\_data](#)

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

## 5.33 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*

### 5.33.1 Function/Subroutine Documentation

5.33.1.1 subroutine `insertionsort_d` ( `real(kind=kind(1.0d0))`, `dimension(na)`, `intent(inout) A`, `integer, intent(in) nA` )

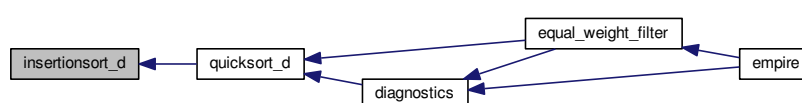
subroutine to sort using the insertionsort algorithm

#### Parameters

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

Definition at line 86 of file `quicksort.f90`.

Here is the caller graph for this function:



5.33.1.2 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**

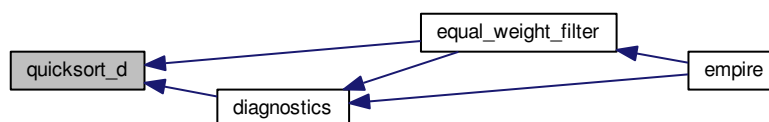
<code>in, out</code>	<code>a</code>	array of doubles to be sorted
<code>in</code>	<code>na</code>	dimension of array a

Definition at line 9 of file quicksort.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.34 src/utls/random\_d.f90 File Reference

**Data Types**

- module [random](#)

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

# Index

- allocate\_data
  - comms, 10
- allocate\_pf
  - pf\_control, 15
- alltests
  - alltests.f90, 67
- alltests.f90
  - alltests, 67
- bin\_prob
  - random, 26
- bprime
  - operator\_wrappers.f90, 61
- check\_scaling
  - stochastic\_model.f90, 54
- comms, 9
  - allocate\_data, 10
  - cpl\_mpi\_comm, 10
  - deallocate\_data, 10
  - gblcount, 10
  - gbldisp, 10
  - initialise\_mpi, 10
  - mype\_id, 11
  - myrank, 11
  - npfs, 11
  - nproc, 11
  - pf\_mpi\_comm, 11
  - pfrank, 11
- configure\_model
  - model\_specific.f90, 35
- count
  - pf\_control::pf\_control\_type, 19
- couple\_root
  - pf\_control::pf\_control\_type, 19
- cpl\_mpi\_comm
  - comms, 10
- data\_io.f90
  - get\_observation\_data, 76
  - get\_state, 76
  - output\_from\_pf, 77
  - save\_observation\_data, 77
  - save\_state, 77
  - save\_truth, 78
- deallocate\_data
  - comms, 10
- deallocate\_pf
  - pf\_control, 15
- deterministic\_model
  - deterministic\_model.f90, 46
- deterministic\_model.f90
  - deterministic\_model, 46
- diagnostics
  - diagnostics.f90, 78
- diagnostics.f90
  - diagnostics, 78
  - trajectories, 79
- dist\_st\_ob
  - model\_specific.f90, 36
- dp
  - random, 31
- eakf\_analysis
  - eakf\_analysis.f90, 47
- eakf\_analysis.f90
  - eakf\_analysis, 47
- efac
  - pf\_control::pf\_control\_type, 19
- empire
  - pf\_couple.f90, 44
- enkf\_specific.f90
  - get\_local\_observation\_data, 47
  - h\_local, 48
  - localise\_enkf, 48
  - solve\_rhalf\_local, 48
- equal\_weight\_filter
  - equivalent\_weights\_step.f90, 49
- equivalent\_weights\_step.f90
  - equal\_weight\_filter, 49
- etkf\_analysis
  - etkf\_analysis.f90, 50
- etkf\_analysis.f90
  - etkf\_analysis, 50
- gblcount
  - comms, 10
- gbldisp
  - comms, 10
- gen\_data
  - pf\_control::pf\_control\_type, 19
- gen\_q
  - pf\_control::pf\_control\_type, 19
- gen\_rand.f90
  - mixture\_randomnumbers1d, 55
  - mixture\_randomnumbers2d, 57
  - normal\_randomnumbers1d, 58
  - normal\_randomnumbers2d, 59
  - random\_seed\_mpi, 60
  - uniform\_randomnumbers1d, 60

- genQ.f90
  - genq, 79
- genq
  - genQ.f90, 79
- get\_local\_observation\_data
  - enkf\_specific.f90, 47
- get\_observation\_data
  - data\_io.f90, 76
- get\_state
  - data\_io.f90, 76
- h
  - model\_specific.f90, 37
- h\_local
  - enkf\_specific.f90, 48
- h\_tests
  - tests.f90, 71
- histogram\_data, 11
  - kill\_histogram\_data, 12
  - load\_histogram\_data, 12
  - rank\_hist\_list, 12
  - rank\_hist\_nums, 12
  - rhl\_n, 12
  - rhn\_n, 12
- hqht\_plus\_r, 13
  - hqhtr\_factor, 13
  - kill\_hqhtr, 13
  - load\_hqhtr, 13
- hqhtr\_factor
  - hqht\_plus\_r, 13
- hqhtr\_tests
  - tests.f90, 72
- ht
  - model\_specific.f90, 38
- human\_readable
  - pf\_control::pf\_control\_type, 19
- init
  - pf\_control::pf\_control\_type, 19
- initialise\_mpi
  - comms, 10
- innerhqht\_plus\_r\_1
  - operator\_wrappers.f90, 62
- innerr\_1
  - operator\_wrappers.f90, 63
- insertionsort\_d
  - quicksort.f90, 80
- k
  - operator\_wrappers.f90, 63
- keep
  - pf\_control::pf\_control\_type, 20
- kill\_histogram\_data
  - histogram\_data, 12
- kill\_hqhtr
  - hqht\_plus\_r, 13
- killq
  - qdata, 23
- killr
  - rdata, 32
- len
  - pf\_control::pf\_control\_type, 20
- letkf\_analysis
  - letkf\_analysis.f90, 51
- letkf\_analysis.f90
  - letkf\_analysis, 51
- Ingamma
  - random, 26
- load\_histogram\_data
  - histogram\_data, 12
- load\_hqhtr
  - hqht\_plus\_r, 13
- loadq
  - qdata, 23
- loadr
  - rdata, 32
- localise\_enkf
  - enkf\_specific.f90, 48
- mean
  - pf\_control::pf\_control\_type, 20
- mixture\_randomnumbers1d
  - gen\_rand.f90, 55
- mixture\_randomnumbers2d
  - gen\_rand.f90, 57
- model\_specific.f90, 35
  - configure\_model, 35
  - dist\_st\_ob, 36
  - h, 37
  - ht, 38
  - q, 39
  - qhalf, 40
  - r, 40
  - rhalf, 41
  - solve\_hqht\_plus\_r, 42
  - solve\_r, 42
  - solve\_rhalf, 43
- mype\_id
  - comms, 11
- myrank
  - comms, 11
- nens
  - pf\_control::pf\_control\_type, 20
- nfac
  - pf\_control::pf\_control\_type, 20
- normal\_randomnumbers1d
  - gen\_rand.f90, 58
- normal\_randomnumbers2d
  - gen\_rand.f90, 59
- npfs
  - comms, 11
- nproc
  - comms, 11
- nudgefac
  - pf\_control::pf\_control\_type, 20

- obs\_dim
  - sizes, 33
- operator\_wrappers.f90
  - bprime, 61
  - innerhght\_plus\_r\_1, 62
  - innerr\_1, 63
  - k, 63
- output\_from\_pf
  - data\_io.f90, 77
- parse\_pf\_parameters
  - pf\_control, 15
- particles
  - pf\_control::pf\_control\_type, 20
- perturb\_particle
  - perturb\_particle.f90, 64
- perturb\_particle.f90
  - perturb\_particle, 64
  - update\_state, 65
- pf
  - pf\_control, 17
- pf\_control, 14
  - allocate\_pf, 15
  - deallocate\_pf, 15
  - parse\_pf\_parameters, 15
  - pf, 17
  - set\_pf\_controls, 16
- pf\_control::pf\_control\_type, 17
  - count, 19
  - couple\_root, 19
  - efac, 19
  - gen\_data, 19
  - gen\_q, 19
  - human\_readable, 19
  - init, 19
  - keep, 20
  - len, 20
  - mean, 20
  - nens, 20
  - nfac, 20
  - nudgefac, 20
  - particles, 20
  - psi, 20
  - qscale, 21
  - rho, 21
  - talagrand, 21
  - time, 21
  - time\_bwn\_obs, 21
  - time\_obs, 21
  - timestep, 21
  - type, 21
  - ufac, 22
  - use\_mean, 22
  - use\_rmse, 22
  - use\_talagrand, 22
  - use\_traj, 22
  - use\_var, 22
  - use\_weak, 22
  - weight, 22
- pf\_couple.f90
  - empire, 44
- pf\_mpi\_comm
  - comms, 11
- pfrank
  - comms, 11
- proposal\_filter
  - proposal\_filter.f90, 52
- proposal\_filter.f90
  - proposal\_filter, 52
- psi
  - pf\_control::pf\_control\_type, 20
- q
  - model\_specific.f90, 39
- q\_tests
  - tests.f90, 73
- qcol
  - qdata, 24
- qdata, 23
  - killq, 23
  - loadq, 23
  - qcol, 24
  - qdiag, 24
  - qn, 24
  - qne, 24
  - qrow, 24
  - qscale, 24
  - qval, 24
- qdiag
  - qdata, 24
- qhalf
  - model\_specific.f90, 40
- qn
  - qdata, 24
- qne
  - qdata, 24
- qrow
  - qdata, 24
- qscale
  - pf\_control::pf\_control\_type, 21
  - qdata, 24
- quicksort.f90
  - insertionsort\_d, 80
  - quicksort\_d, 80
- quicksort\_d
  - quicksort.f90, 80
- qval
  - qdata, 24
- r
  - model\_specific.f90, 40
- r\_tests
  - tests.f90, 74
- random, 25
  - bin\_prob, 26
  - dp, 31
  - lngamma, 26
  - random\_beta, 26

- random\_binomial1, [26](#)
- random\_binomial2, [27](#)
- random\_cauchy, [27](#)
- random\_chisq, [27](#)
- random\_exponential, [27](#)
- random\_gamma, [27](#)
- random\_gamma1, [28](#)
- random\_gamma2, [28](#)
- random\_inv\_gauss, [29](#)
- random\_mvnorm, [29](#)
- random\_neg\_binomial, [29](#)
- random\_normal, [29](#)
- random\_order, [30](#)
- random\_poisson, [30](#)
- random\_t, [30](#)
- random\_von\_mises, [30](#)
- random\_weibull, [30](#)
- seed\_random\_number, [31](#)
- random\_beta
  - random, [26](#)
- random\_binomial1
  - random, [26](#)
- random\_binomial2
  - random, [27](#)
- random\_cauchy
  - random, [27](#)
- random\_chisq
  - random, [27](#)
- random\_exponential
  - random, [27](#)
- random\_gamma
  - random, [27](#)
- random\_gamma1
  - random, [28](#)
- random\_gamma2
  - random, [28](#)
- random\_inv\_gauss
  - random, [29](#)
- random\_mvnorm
  - random, [29](#)
- random\_neg\_binomial
  - random, [29](#)
- random\_normal
  - random, [29](#)
- random\_order
  - random, [30](#)
- random\_poisson
  - random, [30](#)
- random\_seed\_mpi
  - gen\_rand.f90, [60](#)
- random\_t
  - random, [30](#)
- random\_von\_mises
  - random, [30](#)
- random\_weibull
  - random, [30](#)
- rank\_hist\_list
  - histogram\_data, [12](#)
- rank\_hist\_nums
  - histogram\_data, [12](#)
- rcol
  - rdata, [32](#)
- rdata, [31](#)
  - killr, [32](#)
  - loadr, [32](#)
  - rcol, [32](#)
  - rdiag, [32](#)
  - rn, [32](#)
  - rne, [32](#)
  - rrow, [33](#)
  - rval, [33](#)
- rdiag
  - rdata, [32](#)
- resample
  - resample.f90, [66](#)
- resample.f90
  - resample, [66](#)
- rhalf
  - model\_specific.f90, [41](#)
- rhl\_n
  - histogram\_data, [12](#)
- rhn\_n
  - histogram\_data, [12](#)
- rho
  - pf\_control::pf\_control\_type, [21](#)
- rn
  - rdata, [32](#)
- rne
  - rdata, [32](#)
- rrow
  - rdata, [33](#)
- rval
  - rdata, [33](#)
- save\_observation\_data
  - data\_io.f90, [77](#)
- save\_state
  - data\_io.f90, [77](#)
- save\_truth
  - data\_io.f90, [78](#)
- 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, [42](#)
- solve\_r
  - model\_specific.f90, [42](#)
- solve\_rhalf
  - model\_specific.f90, [43](#)



- solve\_rhalf\_local
  - enkf\_specific.f90, 48
- src/DOC\_README.txt, 46
- src/controllers/pf\_control.f90, 44
- src/controllers/pf\_couple.f90, 44
- src/controllers/pf\_parameters.dat, 45
- src/controllers/sizes.f90, 45
- src/data/Qdata.f90, 46
- src/data/Rdata.f90, 46
- src/filters/deterministic\_model.f90, 46
- src/filters/eakf\_analysis.f90, 46
- src/filters/enkf\_specific.f90, 47
- src/filters/equivalent\_weights\_step.f90, 49
- src/filters/etkf\_analysis.f90, 50
- src/filters/letkf\_analysis.f90, 51
- src/filters/proposal\_filter.f90, 52
- src/filters/sir\_filter.f90, 53
- src/filters/stochastic\_model.f90, 54
- src/operations/gen\_rand.f90, 55
- src/operations/operator\_wrappers.f90, 61
- src/operations/perturb\_particle.f90, 64
- src/operations/resample.f90, 66
- src/tests/alltests.f90, 67
- src/tests/test\_h.f90, 67
- src/tests/test\_hqhtr.f90, 68
- src/tests/test\_q.f90, 69
- src/tests/test\_r.f90, 70
- src/tests/tests.f90, 71
- src/utills/comms.f90, 75
- src/utills/data\_io.f90, 75
- src/utills/diagnostics.f90, 78
- src/utills/genQ.f90, 79
- src/utills/histogram.f90, 80
- src/utills/quicksort.f90, 80
- src/utills/random\_d.f90, 82
- 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, 21
- test\_h
  - test\_h.f90, 68
- test\_h.f90
  - test\_h, 68
- 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, 70
- test\_r.f90
  - test\_r, 70
- tests.f90
  - h\_tests, 71
  - hqhtr\_tests, 72
  - q\_tests, 73
  - r\_tests, 74
- time
  - pf\_control::pf\_control\_type, 21
- time\_bwn\_obs
  - pf\_control::pf\_control\_type, 21
- time\_obs
  - pf\_control::pf\_control\_type, 21
- timestep
  - pf\_control::pf\_control\_type, 21
- trajectories
  - diagnostics.f90, 79
- type
  - pf\_control::pf\_control\_type, 21
- ufac
  - pf\_control::pf\_control\_type, 22
- uniformrandomnumbers1d
  - gen\_rand.f90, 60
- update\_state
  - perturb\_particle.f90, 65
- use\_mean
  - pf\_control::pf\_control\_type, 22
- use\_rmse
  - pf\_control::pf\_control\_type, 22
- use\_talagrand
  - pf\_control::pf\_control\_type, 22
- use\_traj
  - pf\_control::pf\_control\_type, 22
- use\_var
  - pf\_control::pf\_control\_type, 22
- use\_weak
  - pf\_control::pf\_control\_type, 22
- weight
  - pf\_control::pf\_control\_type, 22