

EMPIRE DA

0.1

Generated by Doxygen 1.8.8

Fri Nov 14 2014 14:37:17

Contents

1	EMPIRE Data Assimilation Documentation	1
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
2	Data Type Index	5
2.1	Data Types List	5
3	File Index	7
3.1	File List	7
4	Data Type Documentation	9
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	nproc	11
4.1.3.8	pf_mpi_comm	11
4.1.3.9	pfrank	11
4.2	histogram_data Module Reference	11
4.2.1	Detailed Description	12
4.2.2	Member Function/Subroutine Documentation	12
4.2.2.1	kill_histogram_data	12
4.2.2.2	load_histogram_data	12
4.2.3	Member Data Documentation	12
4.2.3.1	rank_hist_list	12
4.2.3.2	rank_hist_nums	12
4.2.3.3	rhl_n	12
4.2.3.4	rhn_n	12
4.3	hqht_plus_r Module Reference	13
4.3.1	Detailed Description	13
4.3.2	Member Function/Subroutine Documentation	13
4.3.2.1	hqhtr_factor	13
4.3.2.2	kill_hqhtr	13
4.3.2.3	load_hqhtr	13
4.4	pf_control Module Reference	14
4.4.1	Detailed Description	14
4.4.2	Member Function/Subroutine Documentation	15
4.4.2.1	allocate_pf	15
4.4.2.2	deallocate_pf	15
4.4.2.3	parse_pf_parameters	15
4.4.2.4	set_pf_controls	16
4.4.3	Member Data Documentation	17
4.4.3.1	pf	17
4.5	pf_control::pf_control_type Type Reference	17
4.5.1	Detailed Description	19
4.5.2	Member Data Documentation	19
4.5.2.1	count	19
4.5.2.2	couple_root	19
4.5.2.3	efac	19
4.5.2.4	gen_data	19
4.5.2.5	gen_q	19
4.5.2.6	human_readable	19
4.5.2.7	init	20
4.5.2.8	keep	20
4.5.2.9	len	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	random_beta	26
4.7.2.4	random_binomial1	26
4.7.2.5	random_binomial2	27
4.7.2.6	random_cauchy	27
4.7.2.7	random_chisq	27
4.7.2.8	random_exponential	27
4.7.2.9	random_gamma	28
4.7.2.10	random_gamma1	28
4.7.2.11	random_gamma2	29
4.7.2.12	random_inv_gauss	29
4.7.2.13	random_mvnorm	29
4.7.2.14	random_neg_binomial	29
4.7.2.15	random_normal	29
4.7.2.16	random_order	30
4.7.2.17	random_poisson	30
4.7.2.18	random_t	30
4.7.2.19	random_von_mises	30
4.7.2.20	random_weibull	31
4.7.2.21	seed_random_number	31
4.7.3	Member Data Documentation	31
4.7.3.1	dp	31
4.8	rdata Module Reference	31
4.8.1	Detailed Description	31
4.8.2	Member Function/Subroutine Documentation	32
4.8.2.1	killr	32
4.8.2.2	loadr	32
4.8.3	Member Data Documentation	32
4.8.3.1	rcol	32
4.8.3.2	rdiag	32
4.8.3.3	rn	32
4.8.3.4	rne	33
4.8.3.5	rrow	33
4.8.3.6	rval	33
4.9	sizes Module Reference	33
4.9.1	Detailed Description	33
4.9.2	Member Data Documentation	33
4.9.2.1	obs_dim	33
4.9.2.2	state_dim	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	reconfigure_model	41
5.1.1.9	rhalf	41
5.1.1.10	solve_hqht_plus_r	42
5.1.1.11	solve_r	42
5.1.1.12	solve_rhalf	44
5.2	src/controllers/pf_control.f90 File Reference	45
5.3	src/controllers/pf_couple.f90 File Reference	45
5.3.1	Function/Subroutine Documentation	45
5.3.1.1	empire	45
5.4	src/controllers/pf_parameters.dat File Reference	46
5.5	src/controllers/sizes.f90 File Reference	46
5.6	src/data/Qdata.f90 File Reference	47
5.7	src/data/Rdata.f90 File Reference	47
5.8	src/DOC_README.txt File Reference	47
5.9	src/filters/deterministic_model.f90 File Reference	47
5.9.1	Function/Subroutine Documentation	47
5.9.1.1	deterministic_model	47
5.10	src/filters/eakf_analysis.f90 File Reference	47
5.10.1	Function/Subroutine Documentation	48
5.10.1.1	eakf_analysis	48
5.11	src/filters/enkf_specific.f90 File Reference	48
5.11.1	Function/Subroutine Documentation	48
5.11.1.1	get_local_observation_data	48
5.11.1.2	h_local	49
5.11.1.3	localise_enkf	49
5.11.1.4	solve_rhalf_local	50
5.12	src/filters/equivalent_weights_step.f90 File Reference	50
5.12.1	Function/Subroutine Documentation	50
5.12.1.1	equal_weight_filter	50
5.13	src/filters/etkf_analysis.f90 File Reference	51
5.13.1	Function/Subroutine Documentation	51

5.13.1.1	etkf_analysis	52
5.14	src/filters/etkf_analysis.f90 File Reference	52
5.14.1	Function/Subroutine Documentation	52
5.14.1.1	etkf_analysis	52
5.15	src/filters/proposal_filter.f90 File Reference	53
5.15.1	Function/Subroutine Documentation	53
5.15.1.1	proposal_filter	53
5.16	src/filters/sir_filter.f90 File Reference	54
5.16.1	Function/Subroutine Documentation	54
5.16.1.1	sir_filter	54
5.17	src/filters/stochastic_model.f90 File Reference	55
5.17.1	Function/Subroutine Documentation	55
5.17.1.1	check_scaling	55
5.17.1.2	stochastic_model	56
5.18	src/operations/gen_rand.f90 File Reference	56
5.18.1	Function/Subroutine Documentation	56
5.18.1.1	mixturerandomnumbers1d	57
5.18.1.2	mixturerandomnumbers2d	58
5.18.1.3	normalrandomnumbers1d	59
5.18.1.4	normalrandomnumbers2d	60
5.18.1.5	random_seed_mpi	61
5.18.1.6	uniformrandomnumbers1d	61
5.19	src/operations/operator_wrappers.f90 File Reference	62
5.19.1	Function/Subroutine Documentation	62
5.19.1.1	bprime	62
5.19.1.2	innerhqht_plus_r_1	63
5.19.1.3	innerr_1	64
5.19.1.4	k	64
5.20	src/operations/perturb_particle.f90 File Reference	65
5.20.1	Function/Subroutine Documentation	65
5.20.1.1	perturb_particle	65
5.20.1.2	update_state	66
5.21	src/operations/resample.f90 File Reference	67
5.21.1	Function/Subroutine Documentation	67
5.21.1.1	resample	67
5.22	src/tests/alltests.f90 File Reference	68
5.22.1	Function/Subroutine Documentation	68
5.22.1.1	alltests	68
5.23	src/tests/test_h.f90 File Reference	68
5.23.1	Function/Subroutine Documentation	69

5.23.1.1 test_h	69
5.24 src/tests/test_hqhtr.f90 File Reference	69
5.24.1 Function/Subroutine Documentation	69
5.24.1.1 test_hqhtr	69
5.25 src/tests/test_q.f90 File Reference	70
5.25.1 Function/Subroutine Documentation	70
5.25.1.1 test_q	70
5.26 src/tests/test_r.f90 File Reference	71
5.26.1 Function/Subroutine Documentation	71
5.26.1.1 test_r	71
5.27 src/tests/tests.f90 File Reference	72
5.27.1 Function/Subroutine Documentation	72
5.27.1.1 h_tests	72
5.27.1.2 hqhtr_tests	73
5.27.1.3 q_tests	74
5.27.1.4 r_tests	75
5.28 src/utls/comms.f90 File Reference	76
5.29 src/utls/data_io.f90 File Reference	76
5.29.1 Function/Subroutine Documentation	77
5.29.1.1 get_observation_data	77
5.29.1.2 get_state	77
5.29.1.3 output_from_pf	78
5.29.1.4 save_observation_data	78
5.29.1.5 save_state	78
5.29.1.6 save_truth	79
5.30 src/utls/diagnostics.f90 File Reference	79
5.30.1 Function/Subroutine Documentation	79
5.30.1.1 diagnostics	79
5.30.1.2 trajectories	80
5.31 src/utls/genQ.f90 File Reference	80
5.31.1 Function/Subroutine Documentation	80
5.31.1.1 genq	80
5.32 src/utls/histogram.f90 File Reference	81
5.33 src/utls/quicksort.f90 File Reference	81
5.33.1 Function/Subroutine Documentation	81
5.33.1.1 insertionsort_d	81
5.33.1.2 quicksort_d	82
5.34 src/utls/random_d.f90 File Reference	83

Chapter 1

EMPIRE Data Assimilation Documentation

Author

Philip A. Browne 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 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.

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:

comms	Module containing EMPIRE coupling data	9
histogram_data	Module to control what variables are used to generate rank histograms	11
hqht_plus_r	13
pf_control	Module pf_control holds all the information to control the the main program	14
pf_control::pf_control_type	17
qdata	Module as a place to store user specified data for Q	23
random	A module for random number generation from the following distributions:	25
rdata	Module to hold user supplied data for R observation error covariance matrix	31
sizes	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:

model_specific.f90	35
src/controllers/pf_control.f90	45
src/controllers/pf_couple.f90	45
src/controllers/pf_parameters.dat	46
src/controllers/sizes.f90	46
src/data/Qdata.f90	47
src/data/Rdata.f90	47
src/filters/deterministic_model.f90	47
src/filters/eakf_analysis.f90	47
src/filters/enkf_specific.f90	48
src/filters/ekf_analysis.f90	50
src/filters/etkf_analysis.f90	51
src/filters/letkf_analysis.f90	52
src/filters/proposal_filter.f90	53
src/filters/sir_filter.f90	54
src/filters/stochastic_model.f90	55
src/operations/gen_rand.f90	56
src/operations/operator_wrappers.f90	62
src/operations/perturb_particle.f90	65
src/operations/resample.f90	67
src/tests/alltests.f90	68
src/tests/test_h.f90	68
src/tests/test_hqhtr.f90	69
src/tests/test_q.f90	70
src/tests/test_r.f90	71
src/tests/tests.f90	72
src/utls/comms.f90	76
src/utls/data_io.f90	76
src/utls/diagnostics.f90	79
src/utls/genQ.f90	80
src/utls/histogram.f90	81
src/utls/quicksort.f90	81
src/utls/random_d.f90	83

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

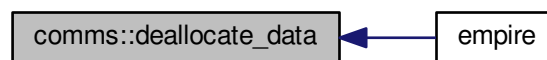
4.1.2.1 subroutine comms::allocate_data ()

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:

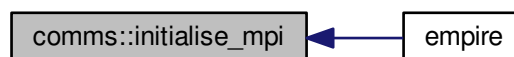


4.1.2.3 subroutine comms::initialise_mpi ()

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

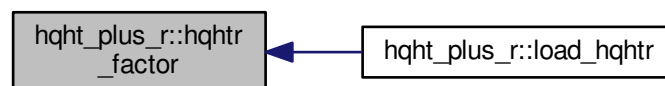
Definition at line 59 of file Rdata.f90.

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:



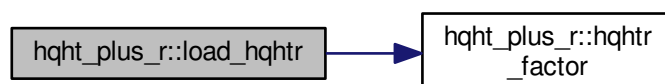
4.3.2.2 subroutine hqht_plus_r::kill_hqhtr ()

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:



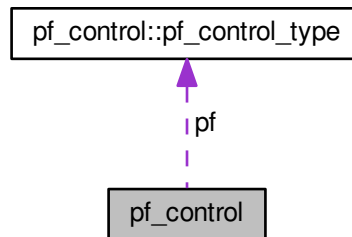
The documentation for this module was generated from the following file:

- [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](#).

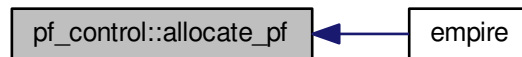
4.4.2 Member Function/Subroutine Documentation

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:

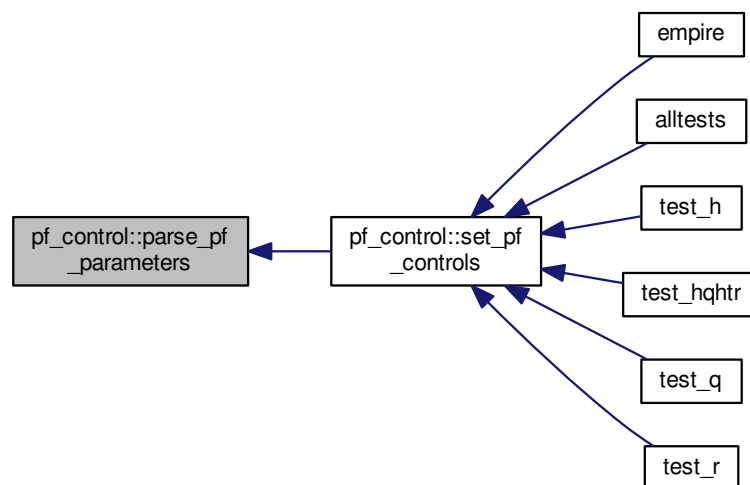
- [init](#)

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` ()

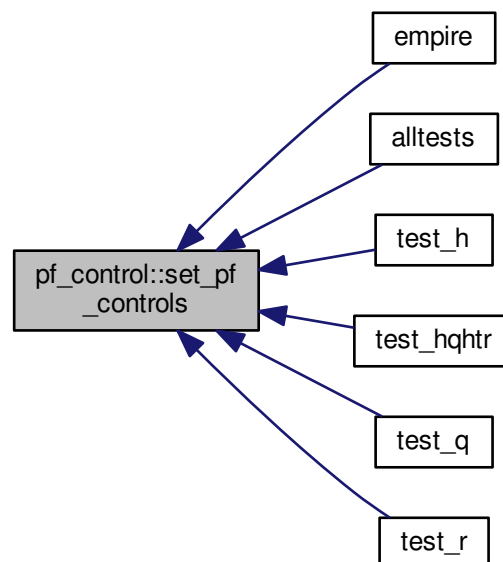
subroutine to ensure [pf_control](#) data is ok

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::nudgefac`

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

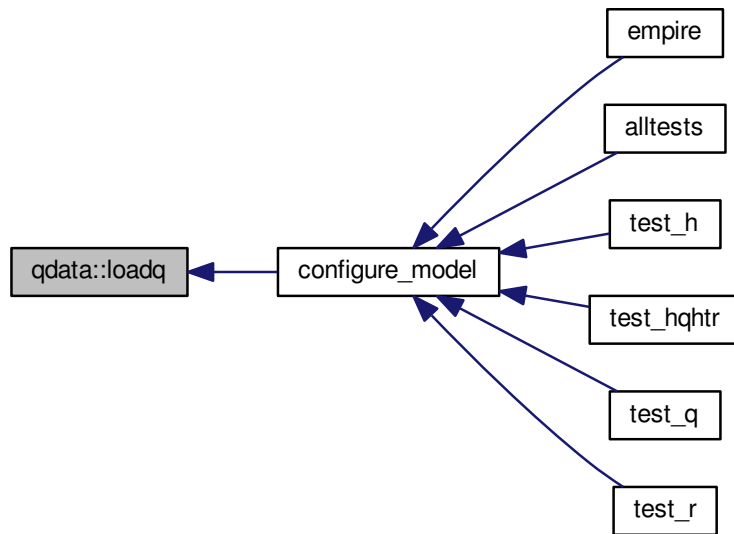
Definition at line 44 of file Qdata.f90.

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

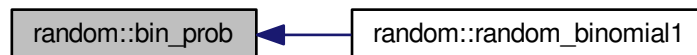
4.7.2.1 `real(kind=kind(1.0d+0)) function random::bin_prob (integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) p, integer, intent(in) r)`

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

Here is the call graph for this function:



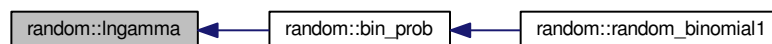
Here is the caller graph for this function:



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:



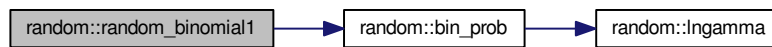
4.7.2.3 `real(kind=kind(1.0d+0)) function random::random_beta (real(kind=kind(1.0d+0)), intent(in) aa, real(kind=kind(1.0d+0)), intent(in) bb, logical, intent(in) first)`

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`.

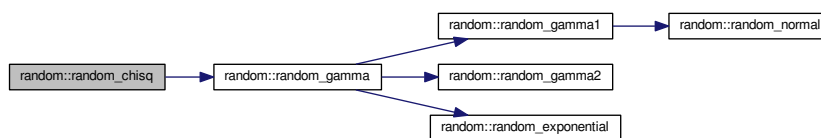
4.7.2.6 real(kind=kind(1.0d+0)) function `random::random_cauchy` ()

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`.

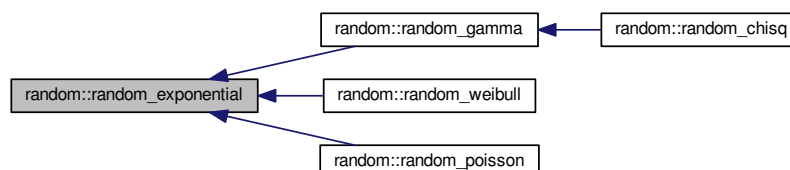
Here is the call graph for this function:



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

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

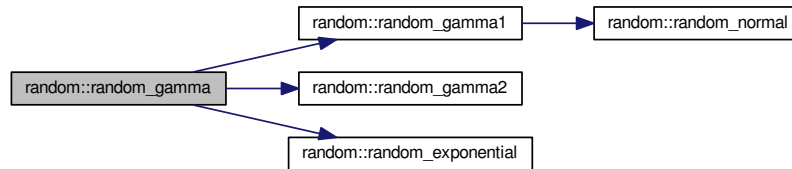
Here is the caller graph for this function:



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:



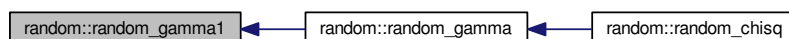
4.7.2.10 `real(kind=kind(1.0d+0)) function random::random_gamma1 (real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first)`

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

Here is the call graph for this function:



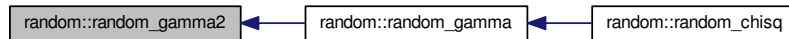
Here is the caller graph for this function:



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 ()`

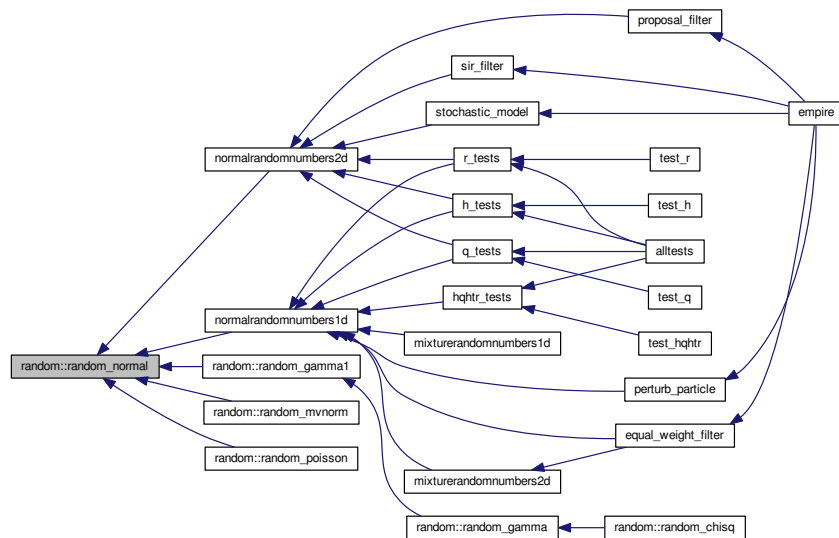
function to get random normal with zero mean and stdev 1

Returns

`fn_val`

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

Here is the caller graph for this function:



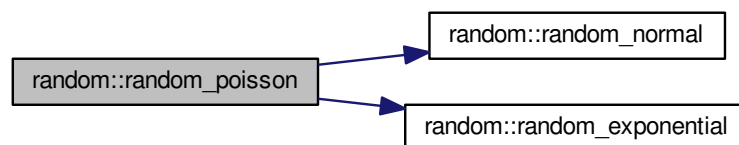
4.7.2.16 subroutine `random::random_order` (integer, dimension(n), intent(out) *order*, integer, intent(in) *n*)

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

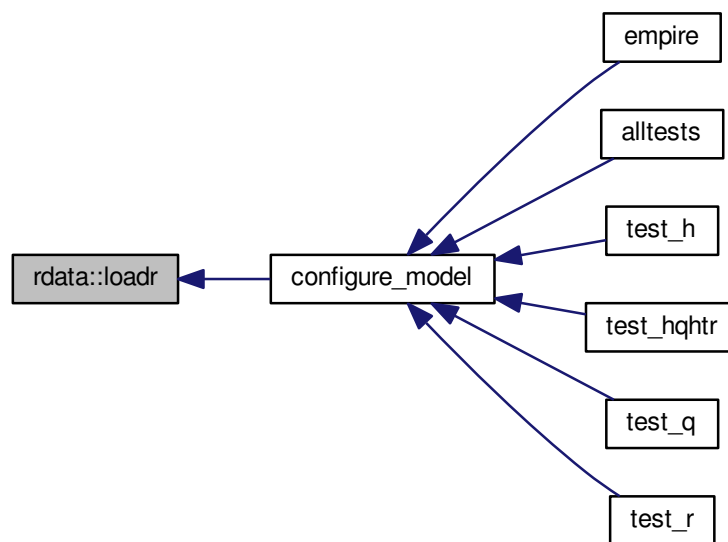
Definition at line 49 of file `Rdata.f90`.

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 [reconfigure_model](#)
subroutine to reset variables that may change when the observation network changes
- 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

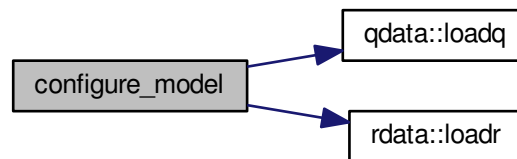
5.1.1 Function/Subroutine Documentation

5.1.1.1 subroutine [configure_model](#) ()

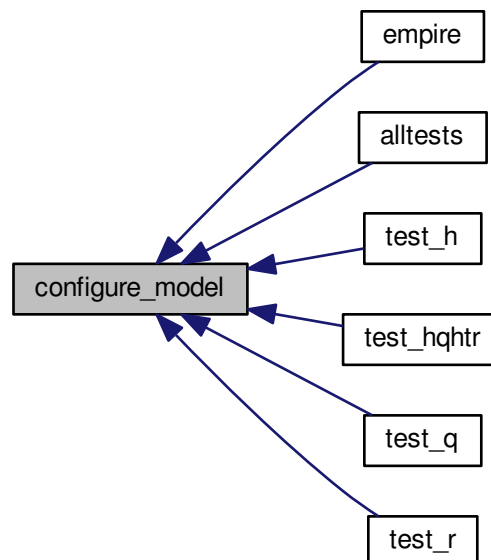
subroutine called initially to set up details and data for model specific functions

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

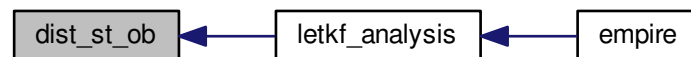
Parameters

<code>in</code>	<code>xp</code>	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 292 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.

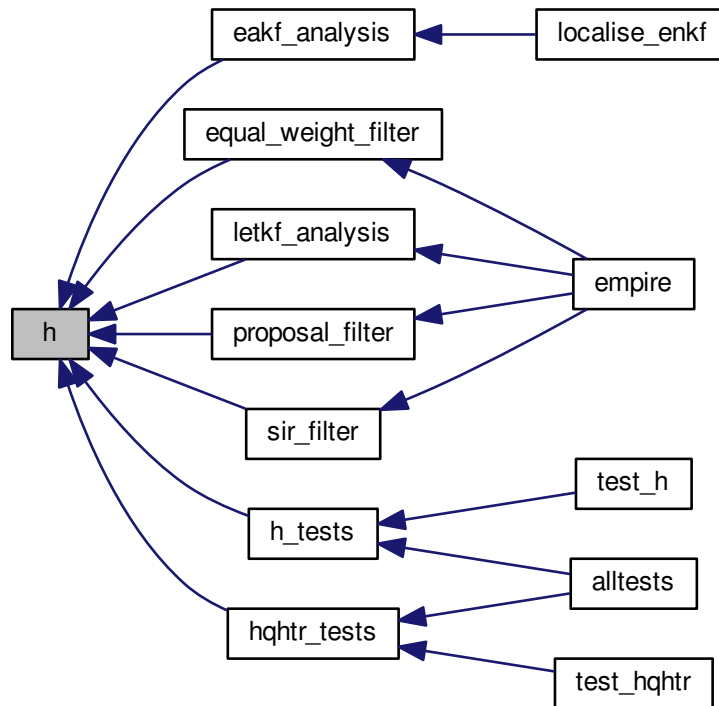
Given *x* compute Hx

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 246 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.

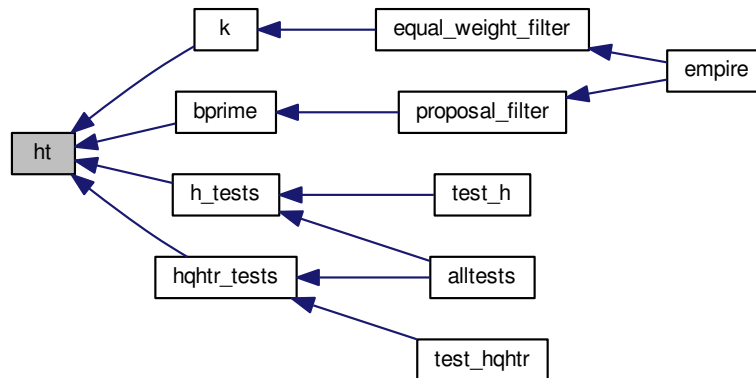
Given y compute $x = H^T(y)$

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 269 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

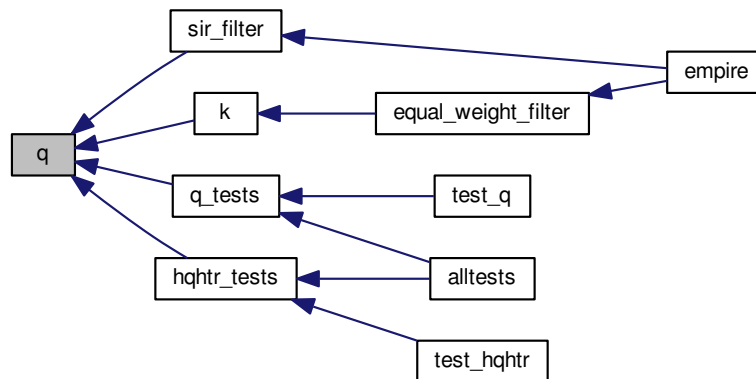
in	<i>nrhs</i>	the number of right hand sides
in	<i>x</i>	the input vector
out	<i>qx</i>	the resulting vector where $Qx = Qx$

Definition at line 156 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.

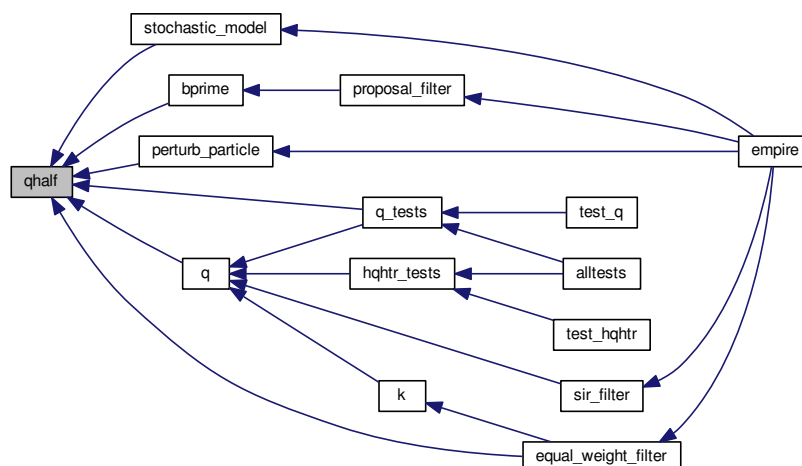
Given x compute $Q^{\frac{1}{2}}x$

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 181 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.

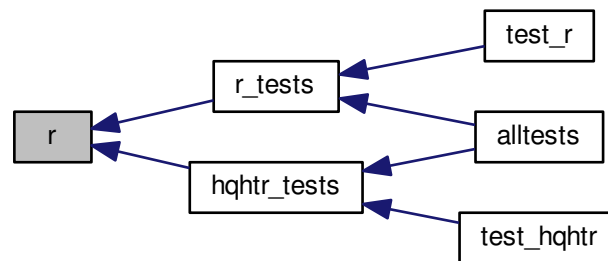
Given *y* compute *Ry*

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 201 of file `model_specific.f90`.

Here is the caller graph for this function:



5.1.1.8 subroutine `reconfigure_model` ()

subroutine to reset variables that may change when the observation network changes

Definition at line 70 of file `model_specific.f90`.

5.1.1.9 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 $R^{\frac{1}{2}}y$

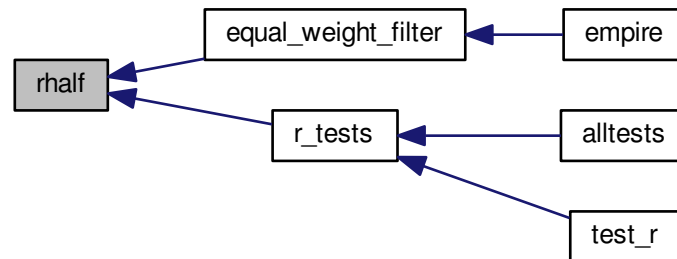
Parameters

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	t	the timestep
----	-----	--------------

Definition at line 223 of file model_specific.f90.

Here is the caller graph for this function:



5.1.1.10 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.

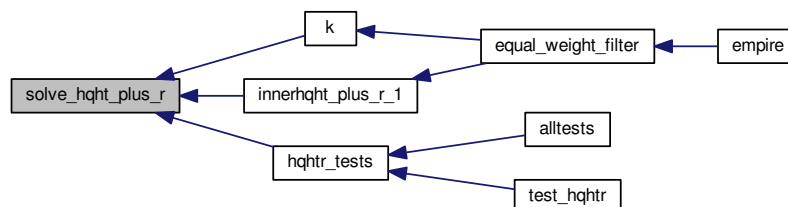
Given y find v such that $(HQH^T + R)v = y$

Parameters

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

Definition at line 136 of file model_specific.f90.

Here is the caller graph for this function:



5.1.1.11 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.

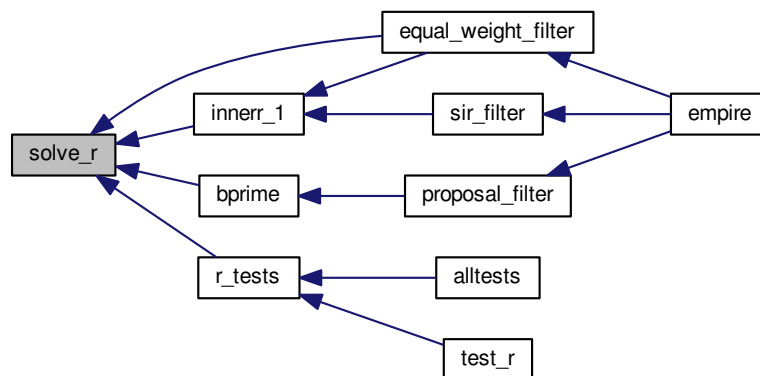
Given y find v such that $Rv = y$

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 94 of file model_specific.f90.

Here is the caller graph for this function:



5.1.1.12 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.

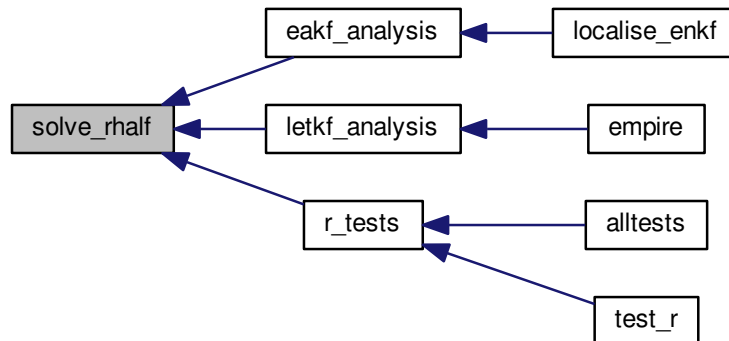
Given *y* find *v* such that $R^{\frac{1}{2}}v = y$

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 114 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

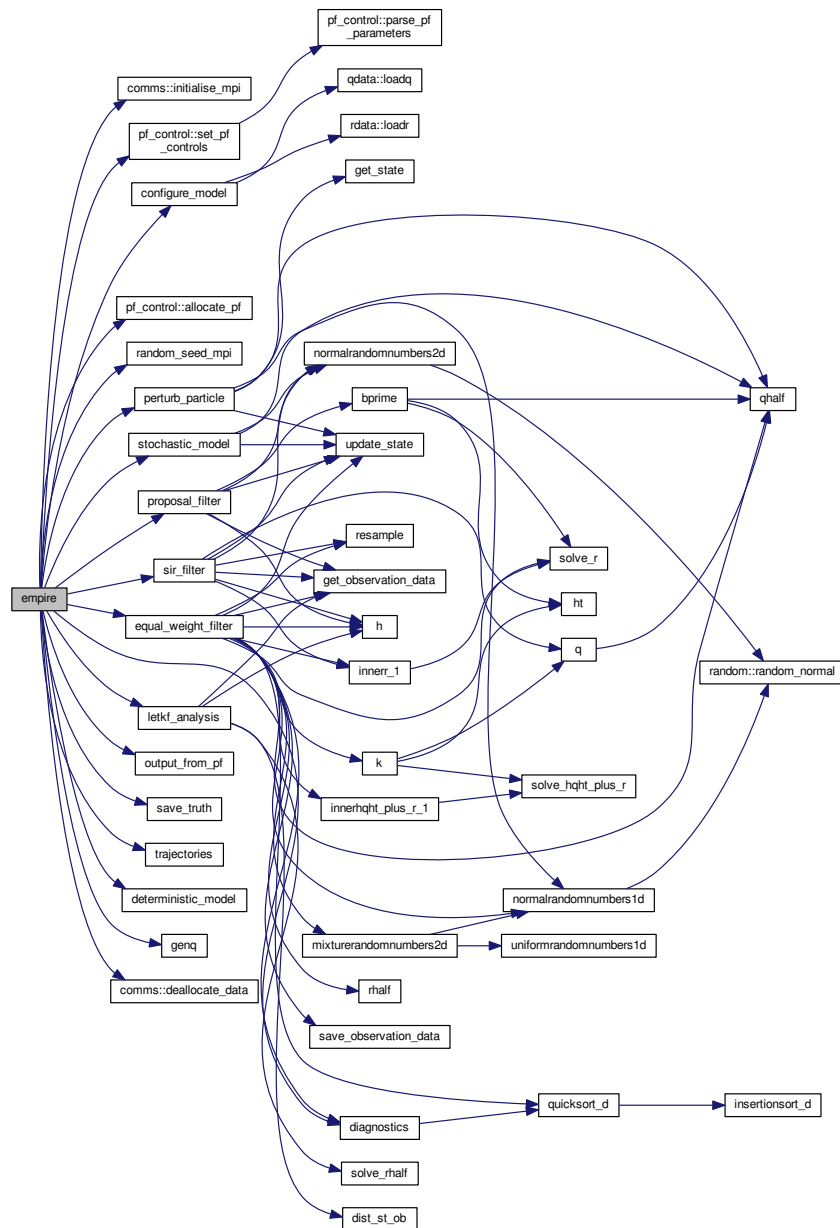
5.3.1 Function/Subroutine Documentation

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

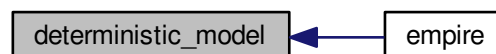
5.9.1 Function/Subroutine Documentation

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

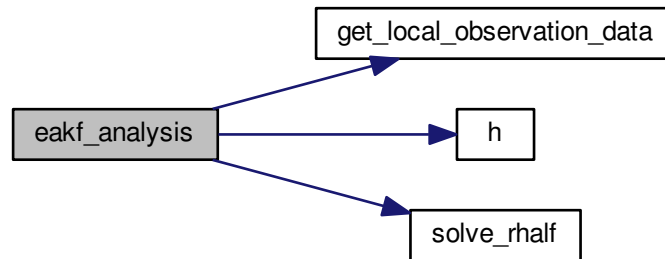
- subroutine [eakf_analysis](#) (num_hor, num_ver, this_hor, this_ver, boundary, x, N, stateDim, obsDim, rho)

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

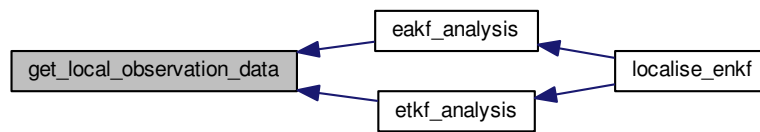
- subroutine `h_local` (`num_hor`, `num_ver`, `this_hor`, `this_ver`, `boundary`, `nrhs`, `stateDim`, `x`, `obsDim`, `y`)
- subroutine `solve_rhalf_local` (`num_hor`, `num_ver`, `this_hor`, `this_ver`, `boundary`, `nrhs`, `obsDim`, `y`, `v`)
- subroutine `get_local_observation_data` (`num_hor`, `num_ver`, `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`.

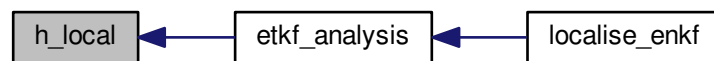
Here is the caller graph for this function:



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`.

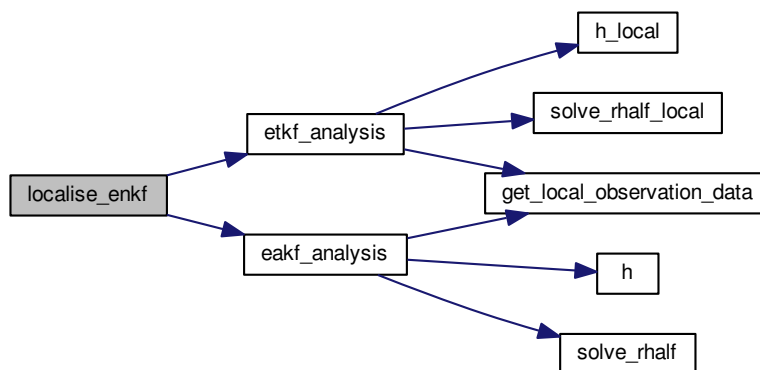
Here is the caller graph for this function:



5.11.1.3 subroutine `localise_enkf` (integer, intent(in) *enkf_analysis*)

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

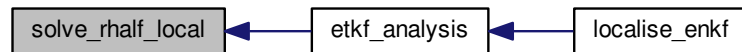
Here is the call graph for this function:



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

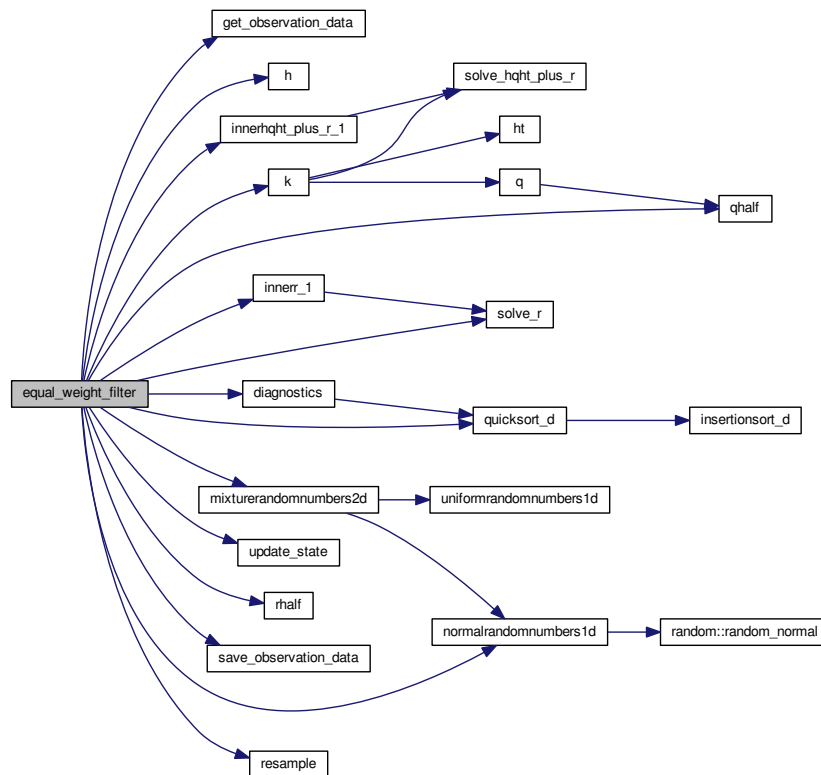
5.12.1 Function/Subroutine Documentation

5.12.1.1 subroutine `equal_weight_filter` ()

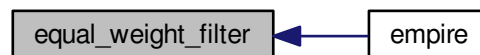
subroutine to do the equivalent weights step

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

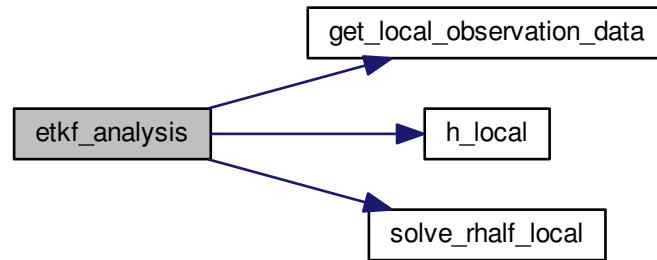
5.13.1 Function/Subroutine Documentation

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

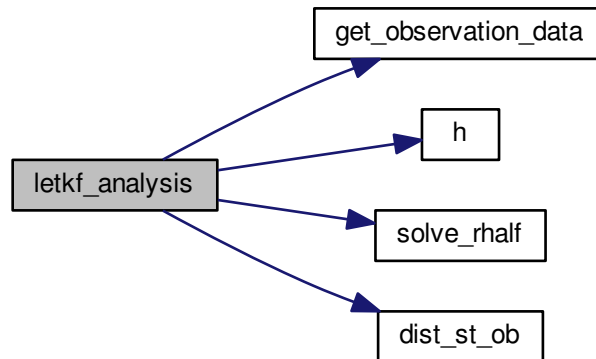
5.14.1.1 subroutine `letkf_analysis` ()

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.

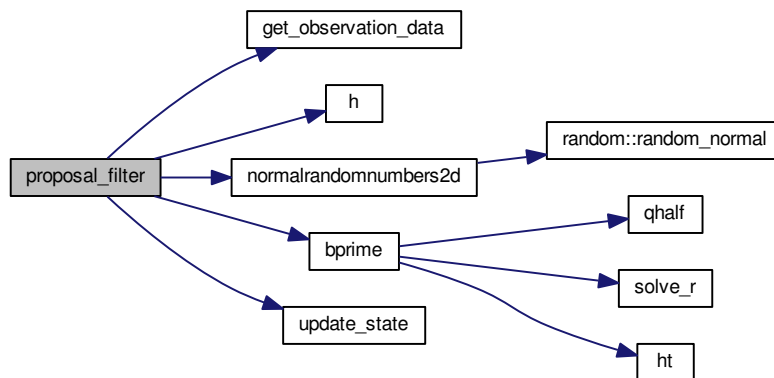
5.15.1 Function/Subroutine Documentation

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)

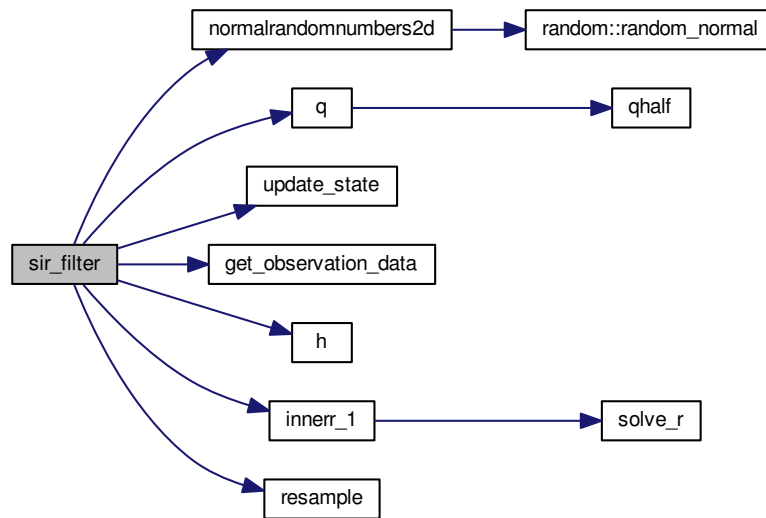
5.16.1 Function/Subroutine Documentation

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)

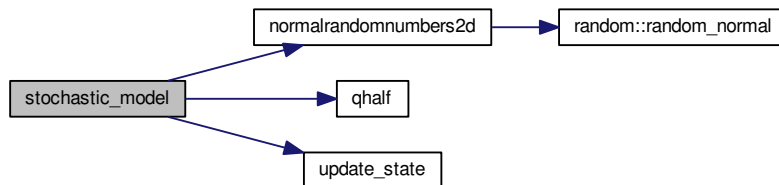
Definition at line 80 of file stochastic_model.f90.

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*)

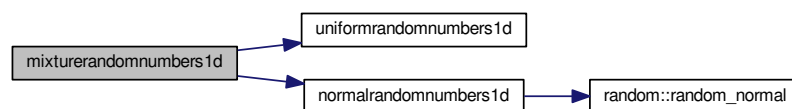
generate one dimensional vector drawn from mixture density

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*)

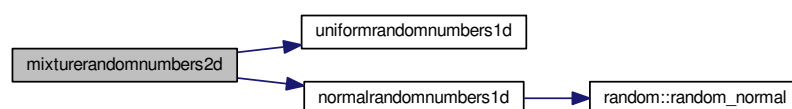
generate two dimensional vector, each drawn from mixture density

Parameters

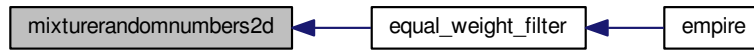
in	<i>mean</i>	Mean of normal distribution
in	<i>stdev</i>	Standard deviation of normal distribution
in	<i>ufac</i>	half-width of uniform distribution that is centered on the mean
in	<i>epsi</i>	Proportion controlling mixture draw. if random_number > epsi then draw from uniform, else normal
in	<i>n</i>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

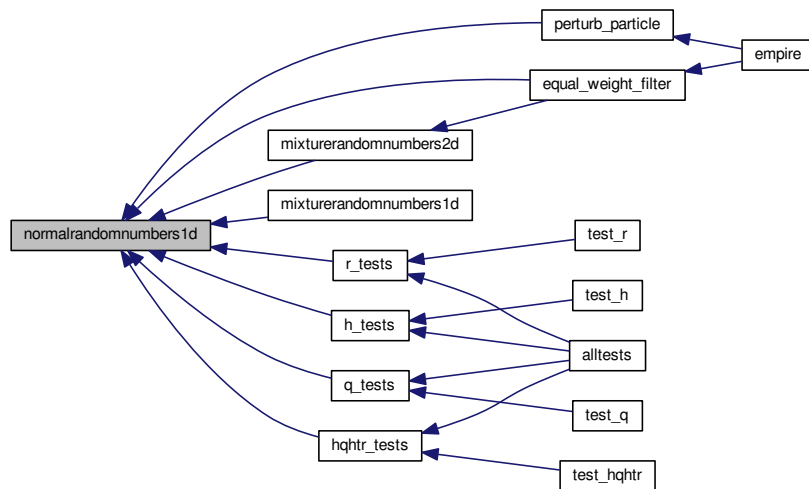
in	<i>n</i>	n size of output vector
in	<i>mean</i>	mean mean of normal distribution
in	<i>stdev</i>	stdev Standard Deviation of normal distribution
out	<i>phi</i>	phi n dimensional normal random numbers

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

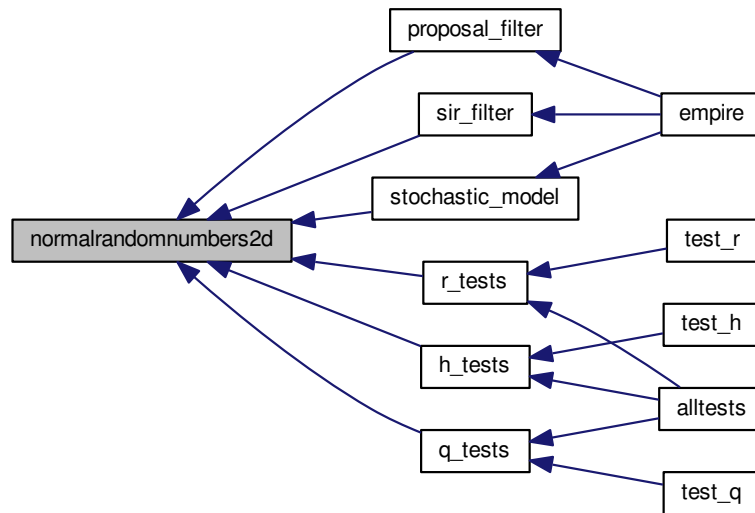
in	<i>n</i>	n first dimension of output vector
in	<i>k</i>	k second dimension of output vector
in	<i>mean</i>	mean mean of normal distribution
in	<i>stdev</i>	stdev Standard Deviation of normal distribution
out	<i>phi</i>	phi n,k dimensional normal random numbers

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*)

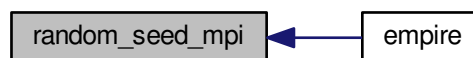
Subroutine to set the random seed across MPI threads.

Parameters

in	<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*)

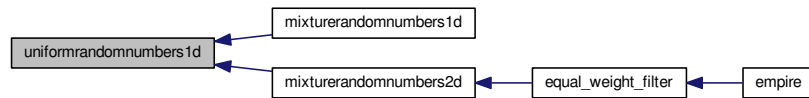
generate one dimension of uniform random numbers

Parameters

in	<i>n</i>	n size of output vector
in	<i>minv</i>	minv minimum value of uniform distribution
in	<i>maxv</i>	maxv maximum value of uniform distribution
out	<i>phi</i>	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 [innerr_1](#) (y, w)
subroutine to compute the inner product with R^{-1}
- subroutine [innerhqht_plus_r_1](#) (y, w)
subroutine to compute the inner product with $(HQH^T + R)^{-1}$
- subroutine [bprime](#) (y, x, QHtR_1y, normaln, betan)
subroutine to calculate nudging term and correlated random errors efficiently

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)

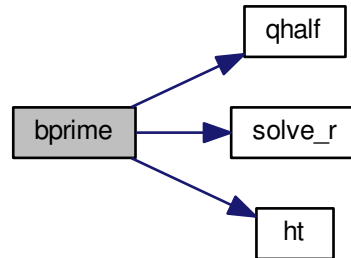
subroutine to calculate nudging term and correlated random errors efficiently

Parameters

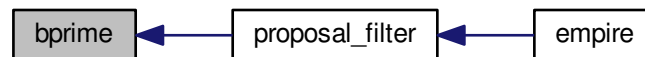
in	<i>y</i>	(obs_dim,pf%count) vectors of innovations $y - H(x^{n-1})$
out	<i>x</i>	(state_dim,pf%count) vectors of $pH^T R^{-1}[y - H(x^{n-1})]$
out	<i>QHtR_1y</i>	(state_dim,pf%count) vectors of $pQH^T R^{-1}[y - H(x^{n-1})]$
in	<i>normaln</i>	(state_dim,pf%count) uncorrelated random vectors such that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

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

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

Parameters

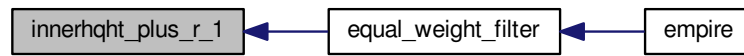
<code>in</code>	<code>y</code>	vector in observation space
<code>out</code>	<code>w</code>	scalar with value $y^T R^{-1} y$

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

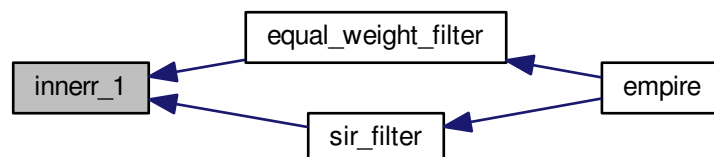
<code>in</code>	<code>y</code>	multiple vectors in observation space (pf%count of them)
<code>out</code>	<code>w</code>	multiple scalars (pf%count) where $w(i)$ has the value $y(:,i)^T R^{-1} y(:,i)$

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`)

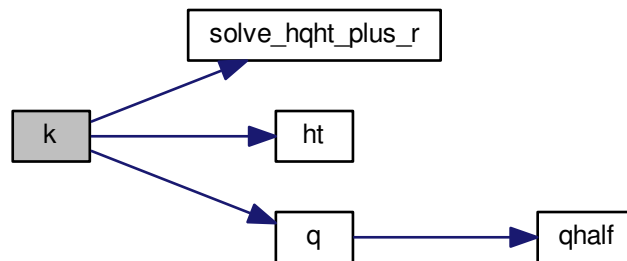
Subroutine to apply K to a vector y in observation space where $K := QH^T(HQH^T + R)^{-1}$.

Parameters

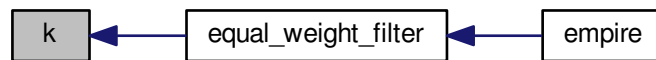
in	y	vector in observation space
out	x	vector in state space

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, fps, kgain, betan)
Subroutine to update the state.

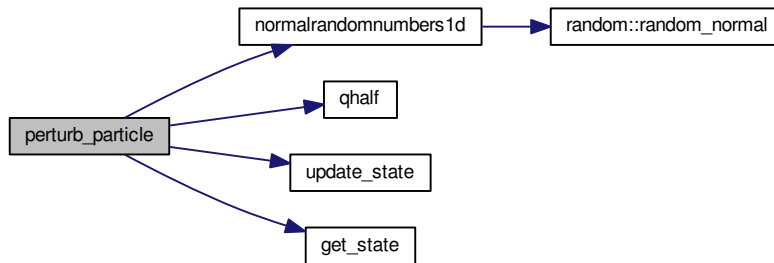
5.20.1 Function/Subroutine Documentation

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_i`, `real(kind=rk)`, `dimension(state_dim)`, `intent(in) kgain`, `real(kind=rk)`, `dimension(state_dim)`, `intent(inout) betan`)

Subroutine to update the state.

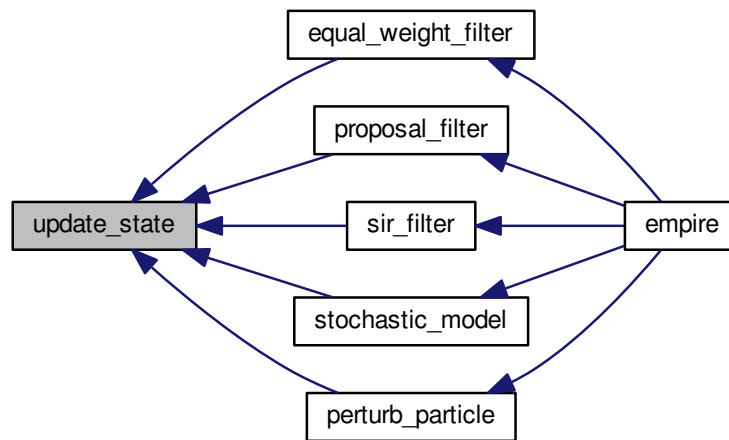
This can be changed for the specific model if it needs to be

Parameters

in	<i>fps_i</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.

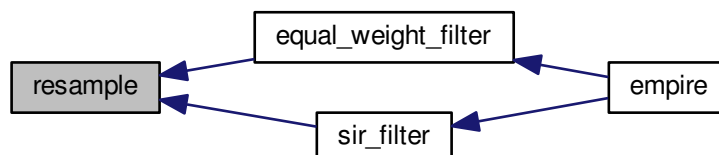
5.21.1 Function/Subroutine Documentation

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

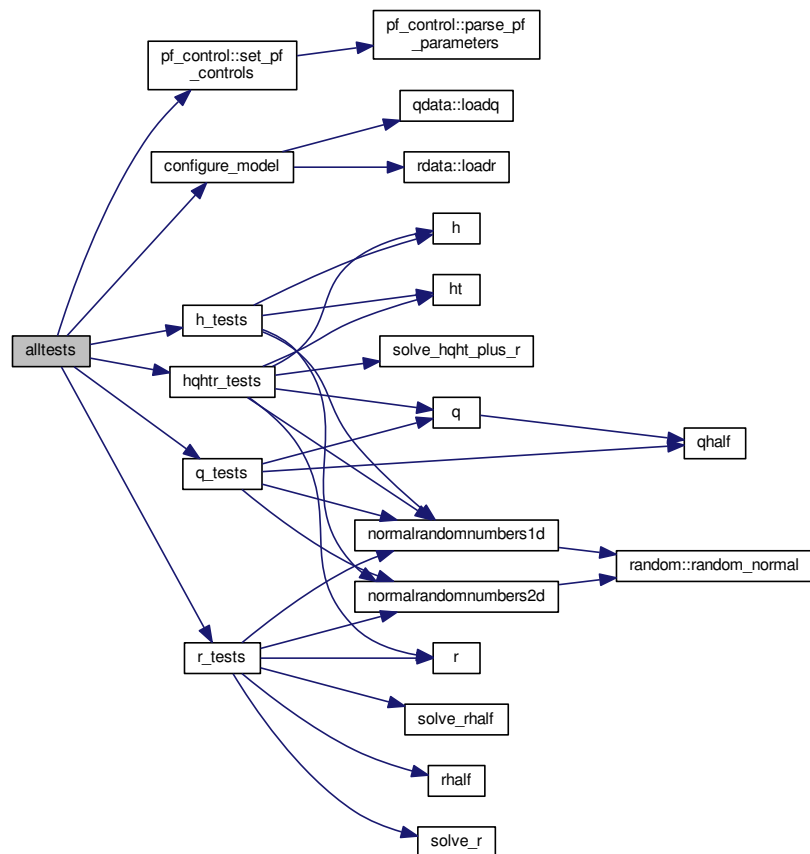
5.22.1 Function/Subroutine Documentation

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

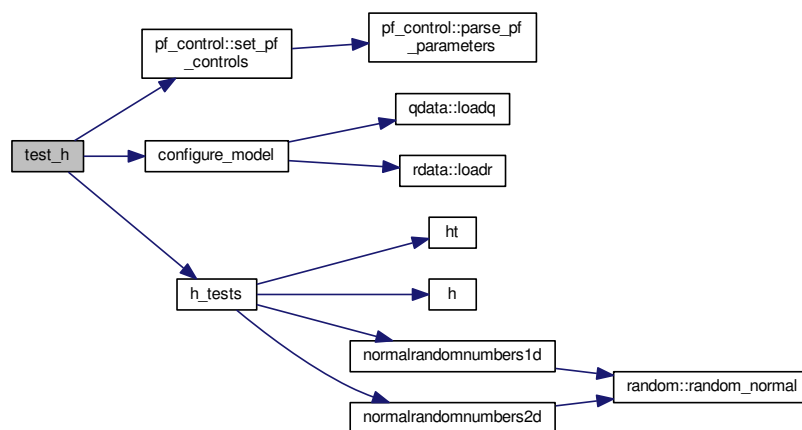
5.23.1 Function/Subroutine Documentation

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

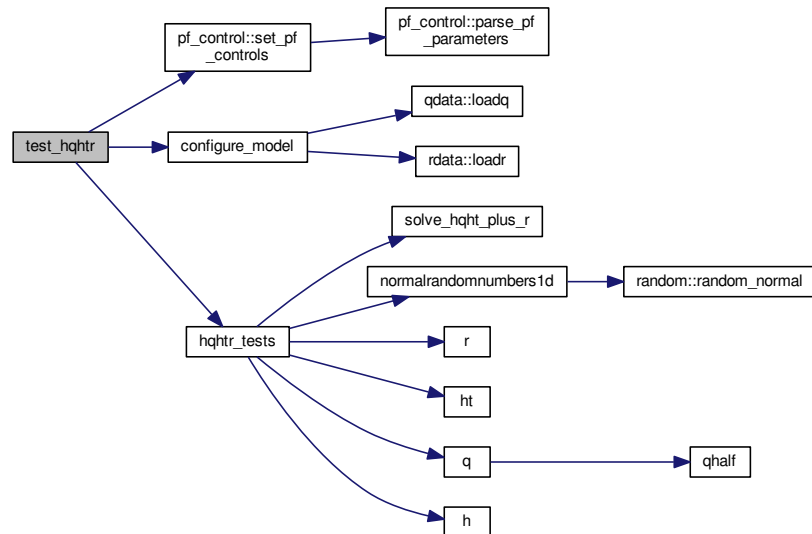
5.24.1.1 program test_hqhtr ()

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

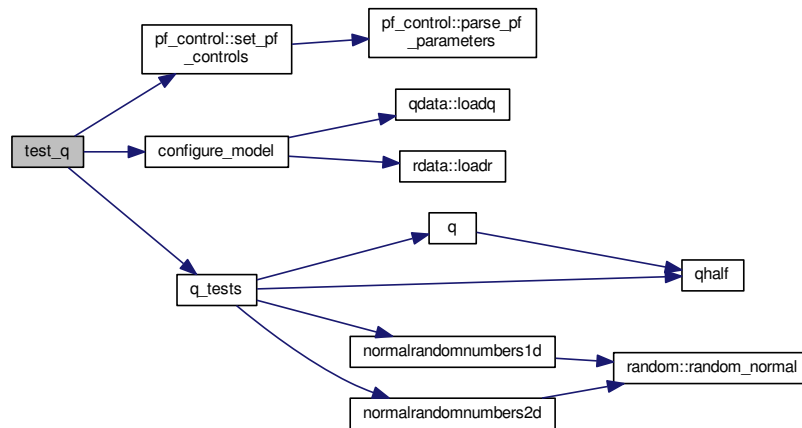
5.25.1 Function/Subroutine Documentation

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/

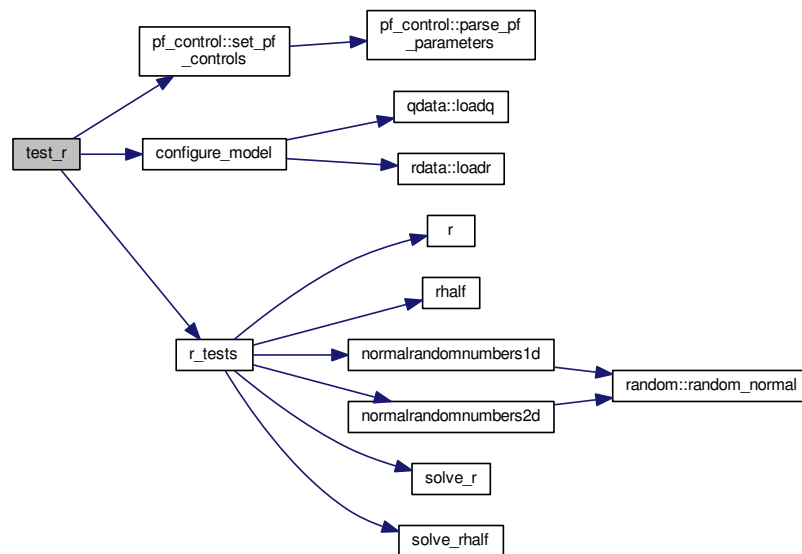
5.26.1 Function/Subroutine Documentation

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](#) ()

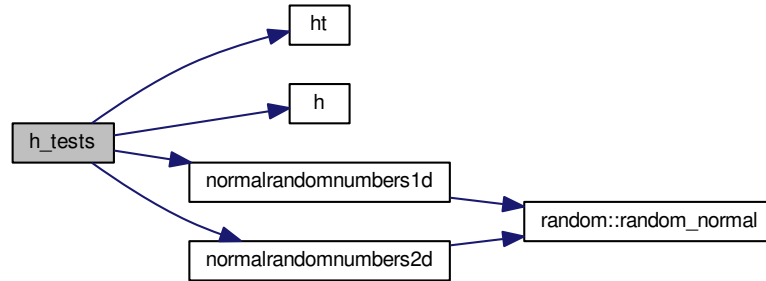
5.27.1 Function/Subroutine Documentation

5.27.1.1 subroutine `h_tests` ()

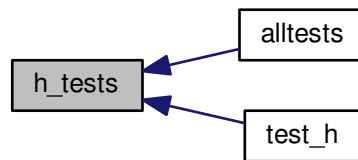
These are some tests to check that the observation operator is implemented correctly

Definition at line 27 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



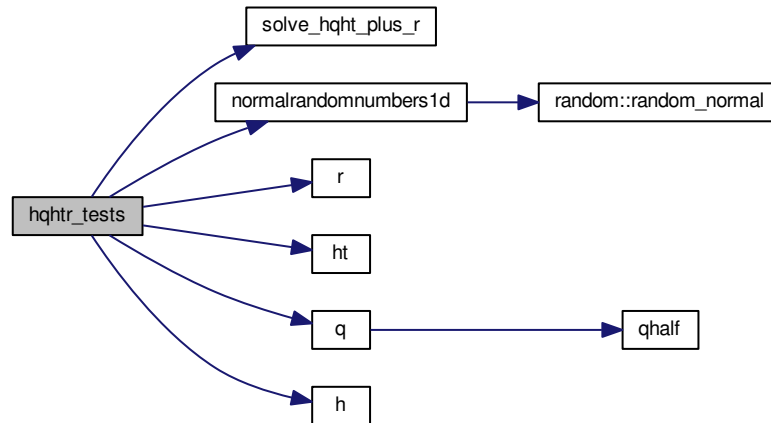
5.27.1.2 subroutine hqhtr_tests ()

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

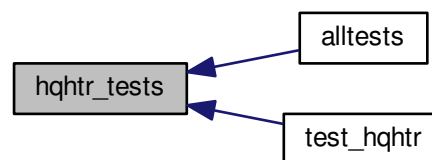
This should check the operation $(HQH^T + R)^{-1}$ is working

Definition at line 876 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

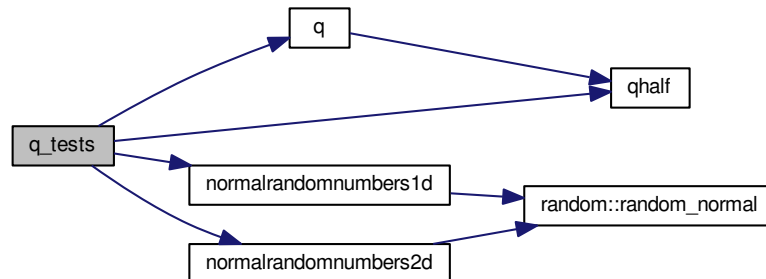


5.27.1.3 subroutine q_tests ()

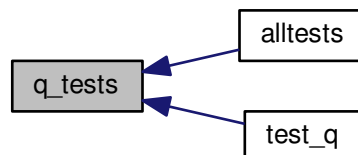
These are some tests to check that the model error covariance matrix is implemented correctly

Definition at line 672 of file tests.f90.

Here is the call graph for this function:



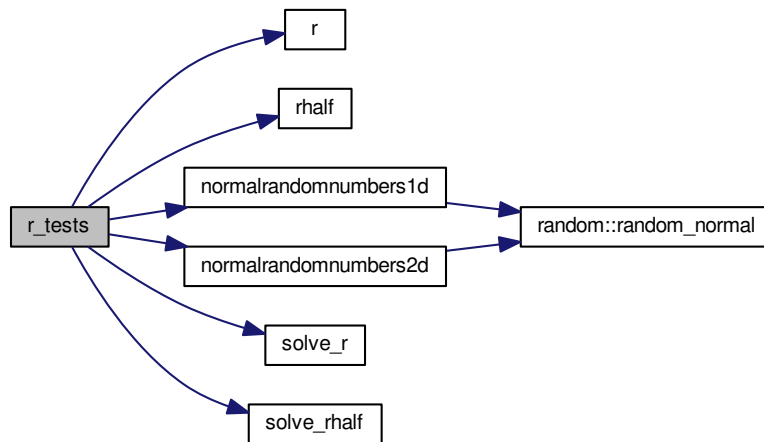
Here is the caller graph for this function:



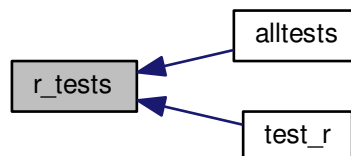
5.27.1.4 subroutine `r_tests` ()

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/utls/comms.f90 File Reference

Data Types

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

5.29 src/utls/data_io.f90 File Reference

Functions/Subroutines

- subroutine [get_observation_data](#) (y)
*Subroutine to read observation from a file
Uses `pftimestep` to determine which observation to read.*
- subroutine [save_observation_data](#) (y)

Subroutine to save observation to a file
 Uses pftimestep to determine which observation to save.

- subroutine `save_truth` (x)

Subroutine to save truth to a file

- subroutine `output_from_pf`

subroutine to ouput data from the filter

- 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)

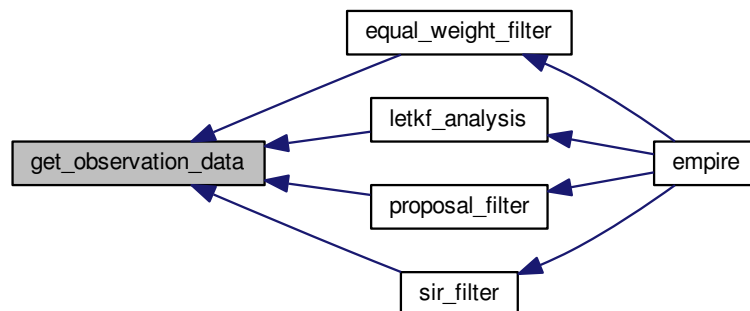
Subroutine to read observation from a file
 Uses pftimestep to determine which observation to read.

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)

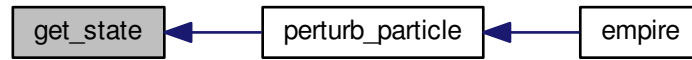
subroutine to write the state vector to a named file as an unformatted fortran file

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

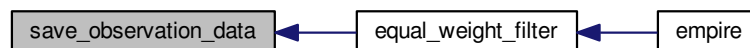
Uses pftimestep to determine which observation to save.

Parameters

in	y	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

in	state	the state vector
in	filename	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

in	x	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

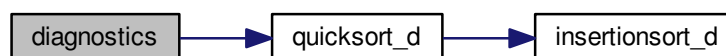
5.30.1 Function/Subroutine Documentation

5.30.1.1 subroutine diagnostics ()

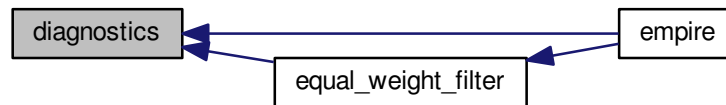
Subroutine to give output diagnostics such as rank histograms and trajectories.

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*)

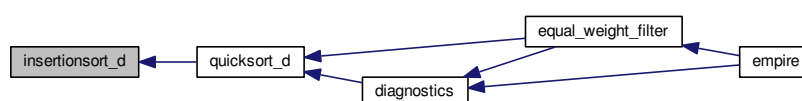
subroutine to sort using the insertionsort algorithm

Parameters

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

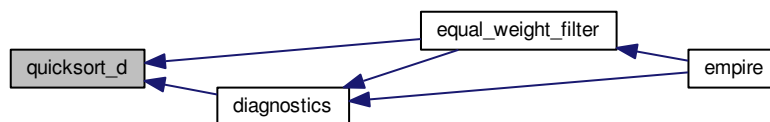
in, out	<i>a</i>	array of doubles to be sorted
in	<i>na</i>	dimension of array a

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/utils/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, [68](#)
- alltests.f90
 - alltests, [68](#)
- bin_prob
 - random, [26](#)
- bprime
 - operator_wrappers.f90, [62](#)
- check_scaling
 - stochastic_model.f90, [55](#)
- 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, [77](#)
 - get_state, [77](#)
 - output_from_pf, [78](#)
 - save_observation_data, [78](#)
 - save_state, [78](#)
 - save_truth, [79](#)
- deallocate_data
 - comms, [10](#)
- deallocate_pf
 - pf_control, [15](#)
- deterministic_model
 - deterministic_model.f90, [47](#)
- deterministic_model.f90
 - deterministic_model, [47](#)
- diagnostics
 - diagnostics.f90, [79](#)
- diagnostics.f90
 - diagnostics, [79](#)
 - trajectories, [80](#)
- dist_st_ob
 - model_specific.f90, [36](#)
- dp
 - random, [31](#)
- eakf_analysis
 - eakf_analysis.f90, [48](#)
- eakf_analysis.f90
 - eakf_analysis, [48](#)
- efac
 - pf_control::pf_control_type, [19](#)
- empire
 - pf_couple.f90, [45](#)
- enkf_specific.f90
 - get_local_observation_data, [48](#)
 - h_local, [49](#)
 - localise_enkf, [49](#)
 - solve_rhalf_local, [49](#)
- equal_weight_filter
 - equivalent_weights_step.f90, [50](#)
- equivalent_weights_step.f90
 - equal_weight_filter, [50](#)
- etkf_analysis
 - etkf_analysis.f90, [51](#)
- etkf_analysis.f90
 - etkf_analysis, [51](#)
- 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, [56](#)
 - mixture_randomnumbers2d, [58](#)
 - normal_randomnumbers1d, [59](#)
 - normal_randomnumbers2d, [60](#)
 - random_seed_mpi, [61](#)
 - uniform_randomnumbers1d, [61](#)

- genQ.f90
 - genq, 80
- genq
 - genQ.f90, 80
- get_local_observation_data
 - enkf_specific.f90, 48
- get_observation_data
 - data_io.f90, 77
- get_state
 - data_io.f90, 77
- h
 - model_specific.f90, 37
- h_local
 - enkf_specific.f90, 49
- h_tests
 - tests.f90, 72
- 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, 73
- 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, 63
- innerr_1
 - operator_wrappers.f90, 64
- insertionsort_d
 - quicksort.f90, 81
- k
 - operator_wrappers.f90, 64
- 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, 52
- letkf_analysis.f90
 - letkf_analysis, 52
- lngamma
 - 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, 49
- mean
 - pf_control::pf_control_type, 20
- mixture_randomnumbers1d
 - gen_rand.f90, 56
- mixture_randomnumbers2d
 - gen_rand.f90, 58
- model_specific.f90, 35
 - configure_model, 35
 - dist_st_ob, 36
 - h, 37
 - ht, 38
 - q, 39
 - qhalf, 40
 - r, 40
 - reconfigure_model, 41
 - rhalf, 41
 - solve_hqht_plus_r, 42
 - solve_r, 42
 - solve_rhalf, 44
- 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, 59
- normal_randomnumbers2d
 - gen_rand.f90, 60
- npfs
 - comms, 11
- nproc
 - comms, 11
- nudgefac
 - pf_control::pf_control_type, 20

- obs_dim
 - sizes, 33
- operator_wrappers.f90
 - bprime, 62
 - innerhght_plus_r_1, 63
 - innerr_1, 64
 - k, 64
- output_from_pf
 - data_io.f90, 78
- parse_pf_parameters
 - pf_control, 15
- particles
 - pf_control::pf_control_type, 20
- perturb_particle
 - perturb_particle.f90, 65
- perturb_particle.f90
 - perturb_particle, 65
 - update_state, 66
- 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, 45
- pf_mpi_comm
 - comms, 11
- pfrank
 - comms, 11
- proposal_filter
 - proposal_filter.f90, 53
- proposal_filter.f90
 - proposal_filter, 53
- psi
 - pf_control::pf_control_type, 20
- q
 - model_specific.f90, 39
- q_tests
 - tests.f90, 74
- 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, 81
 - quicksort_d, 81
- quicksort_d
 - quicksort.f90, 81
- qval
 - qdata, 24
- r
 - model_specific.f90, 40
- r_tests
 - tests.f90, 75
- 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, [61](#)
- 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](#)
- reconfigure_model
 - model_specific.f90, [41](#)
- resample
 - resample.f90, [67](#)
- resample.f90
 - resample, [67](#)
- 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, [78](#)
- save_state
 - data_io.f90, [78](#)
- save_truth
 - data_io.f90, [79](#)
- seed_random_number
 - random, [31](#)
- set_pf_controls
 - pf_control, [16](#)
- sir_filter
 - sir_filter.f90, [54](#)
- sir_filter.f90
 - sir_filter, [54](#)
- 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, [44](#)
- solve_rhalf_local
 - enkf_specific.f90, [49](#)
- src/DOC_README.txt, [47](#)
- src/controllers/pf_control.f90, [45](#)
- src/controllers/pf_couple.f90, [45](#)
- src/controllers/pf_parameters.dat, [46](#)
- src/controllers/sizes.f90, [46](#)
- src/data/Qdata.f90, [47](#)
- src/data/Rdata.f90, [47](#)
- src/filters/deterministic_model.f90, [47](#)
- src/filters/eakf_analysis.f90, [47](#)
- src/filters/enkf_specific.f90, [48](#)
- src/filters/equivalent_weights_step.f90, [50](#)
- src/filters/etkf_analysis.f90, [51](#)
- src/filters/letkf_analysis.f90, [52](#)
- src/filters/proposal_filter.f90, [53](#)
- src/filters/sir_filter.f90, [54](#)
- src/filters/stochastic_model.f90, [55](#)
- src/operations/gen_rand.f90, [56](#)
- src/operations/operator_wrappers.f90, [62](#)
- src/operations/perturb_particle.f90, [65](#)
- src/operations/resample.f90, [67](#)
- src/tests/alltests.f90, [68](#)
- src/tests/test_h.f90, [68](#)
- src/tests/test_hqhtr.f90, [69](#)
- src/tests/test_q.f90, [70](#)
- src/tests/test_r.f90, [71](#)
- src/tests/tests.f90, [72](#)
- src/utis/comms.f90, [76](#)
- src/utis/data_io.f90, [76](#)
- src/utis/diagnostics.f90, [79](#)
- src/utis/genQ.f90, [80](#)
- src/utis/histogram.f90, [81](#)
- src/utis/quicksort.f90, [81](#)
- src/utis/random_d.f90, [83](#)
- state_dim
 - sizes, [33](#)
- stochastic_model
 - stochastic_model.f90, [55](#)
- stochastic_model.f90
 - check_scaling, [55](#)
 - stochastic_model, [55](#)
- talagrand
 - pf_control::pf_control_type, [21](#)
- test_h
 - test_h.f90, [69](#)
- test_h.f90
 - test_h, [69](#)
- test_hqhtr
 - test_hqhtr.f90, [69](#)
- test_hqhtr.f90
 - test_hqhtr, [69](#)
- test_q
 - test_q.f90, [70](#)
- test_q.f90
 - test_q, [70](#)
- test_r
 - test_r.f90, [71](#)
- test_r.f90
 - test_r, [71](#)
- tests.f90
 - h_tests, [72](#)
 - hqhtr_tests, [73](#)
 - q_tests, [74](#)
 - r_tests, [75](#)
- 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, [80](#)
- type
 - pf_control::pf_control_type, [21](#)
- ufac
 - pf_control::pf_control_type, [22](#)
- uniformrandomnumbers1d
 - gen_rand.f90, [61](#)
- update_state
 - perturb_particle.f90, [66](#)
- 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](#)