

EMPIRE DA

0.1

Generated by Doxygen 1.8.6

Fri Sep 26 2014 18:02:52

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	Bug Reports and Functionality Requests	3
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	9
4.1.2.1	allocate_data	9
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	10
4.1.3.4	mype_id	10
4.1.3.5	myrank	10
4.1.3.6	npfs	10
4.1.3.7	nproc	10

4.1.3.8	pf_mpi_comm	10
4.1.3.9	pfrank	11
4.2	histogram_data Module Reference	11
4.2.1	Detailed Description	11
4.2.2	Member Function/Subroutine Documentation	11
4.2.2.1	kill_histogram_data	11
4.2.2.2	load_histogram_data	11
4.2.3	Member Data Documentation	11
4.2.3.1	rank_hist_list	11
4.2.3.2	rank_hist_nums	11
4.2.3.3	rhl_n	11
4.2.3.4	rhn_n	11
4.3	hqht_plus_r Module Reference	12
4.3.1	Member Function/Subroutine Documentation	12
4.3.1.1	hqhtr_factor	12
4.3.1.2	kill_hqhtr	12
4.3.1.3	load_hqhtr	12
4.4	pf_control Module Reference	12
4.4.1	Detailed Description	13
4.4.2	Member Function/Subroutine Documentation	13
4.4.2.1	allocate_pf	13
4.4.2.2	deallocate_pf	14
4.4.2.3	parse_pf_parameters	14
4.4.2.4	set_pf_controls	15
4.4.3	Member Data Documentation	16
4.4.3.1	pf	16
4.5	pf_control::pf_control_type Type Reference	16
4.5.1	Member Data Documentation	18
4.5.1.1	count	18
4.5.1.2	couple_root	18
4.5.1.3	efac	18
4.5.1.4	gen_data	18
4.5.1.5	gen_q	18
4.5.1.6	human_readable	18
4.5.1.7	init	18
4.5.1.8	keep	18
4.5.1.9	len	18
4.5.1.10	mean	19
4.5.1.11	nens	19
4.5.1.12	nfac	19

4.5.1.13	nudgefac	19
4.5.1.14	particles	19
4.5.1.15	psi	19
4.5.1.16	qscale	19
4.5.1.17	rho	19
4.5.1.18	talagrand	19
4.5.1.19	time	19
4.5.1.20	time_bwn_obs	19
4.5.1.21	time_obs	19
4.5.1.22	timestep	20
4.5.1.23	type	20
4.5.1.24	ufac	20
4.5.1.25	use_mean	20
4.5.1.26	use_rmse	20
4.5.1.27	use_talagrand	20
4.5.1.28	use_traj	20
4.5.1.29	use_var	20
4.5.1.30	use_weak	20
4.5.1.31	weight	20
4.6	qdata Module Reference	20
4.6.1	Detailed Description	21
4.6.2	Member Function/Subroutine Documentation	21
4.6.2.1	killq	21
4.6.2.2	loadq	21
4.6.3	Member Data Documentation	21
4.6.3.1	qcol	21
4.6.3.2	qdiag	21
4.6.3.3	qn	21
4.6.3.4	qne	21
4.6.3.5	qrow	21
4.6.3.6	qscale	21
4.6.3.7	qval	21
4.7	random Module Reference	22
4.7.1	Detailed Description	22
4.7.2	Member Function/Subroutine Documentation	22
4.7.2.1	bin_prob	23
4.7.2.2	lngamma	23
4.7.2.3	random_beta	24
4.7.2.4	random_binomial1	24
4.7.2.5	random_binomial2	24

4.7.2.6	random_cauchy	24
4.7.2.7	random_chisq	25
4.7.2.8	random_exponential	25
4.7.2.9	random_gamma	25
4.7.2.10	random_gamma1	26
4.7.2.11	random_gamma2	26
4.7.2.12	random_inv_gauss	27
4.7.2.13	random_mvnorm	27
4.7.2.14	random_neg_binomial	28
4.7.2.15	random_normal	28
4.7.2.16	random_order	29
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	32
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	32
4.8.3.5	rrow	32
4.8.3.6	rval	32
4.9	sizes Module Reference	32
4.9.1	Detailed Description	32
4.9.2	Member Data Documentation	33
4.9.2.1	obs_dim	33
4.9.2.2	state_dim	33
5	File Documentation	35
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	41
5.1.1.7	r	42
5.1.1.8	rhalf	43
5.1.1.9	solve_hqht_plus_r	44
5.1.1.10	solve_r	45
5.1.1.11	solve_rhalf	46
5.2	src/controllers/old_pf_couple.f90 File Reference	47
5.2.1	Function/Subroutine Documentation	47
5.2.1.1	couple_pf	47
5.3	src/controllers/pf_control.f90 File Reference	47
5.4	src/controllers/pf_couple.f90 File Reference	47
5.4.1	Function/Subroutine Documentation	48
5.4.1.1	empire	48
5.5	src/controllers/pf_parameters.dat File Reference	48
5.5.1	Variable Documentation	49
5.5.1.1	gen_data	49
5.5.1.2	gen_Q	49
5.5.1.3	human_readable	49
5.5.1.4	keep	49
5.5.1.5	nfac	49
5.5.1.6	nudgefac	49
5.5.1.7	Qscale	49
5.5.1.8	time_bwn_obs	49
5.5.1.9	time_obs	49
5.5.1.10	type	49
5.5.1.11	ufac	49
5.5.1.12	use_mean	49
5.5.1.13	use_rmse	49
5.5.1.14	use_talagrand	49
5.5.1.15	use_traj	49
5.5.1.16	use_var	49
5.5.1.17	use_weak	49
5.6	src/controllers/sizes.f90 File Reference	50
5.7	src/data/Qdata.f90 File Reference	50
5.8	src/data/Rdata.f90 File Reference	50
5.9	src/DOC_README.txt File Reference	50

5.10	src/filters/eakf_analysis.f90 File Reference	50
5.10.1	Function/Subroutine Documentation	50
5.10.1.1	eakf_analysis	50
5.11	src/filters/enkf_specific.f90 File Reference	51
5.11.1	Function/Subroutine Documentation	51
5.11.1.1	get_local_observation_data	51
5.11.1.2	h_local	52
5.11.1.3	localise_enkf	52
5.11.1.4	solve_rhalf_local	53
5.12	src/filters/equivalent_weights_step.f90 File Reference	53
5.12.1	Function/Subroutine Documentation	53
5.12.1.1	equal_weight_filter	53
5.13	src/filters/etkf_analysis.f90 File Reference	54
5.13.1	Function/Subroutine Documentation	54
5.13.1.1	etkf_analysis	54
5.14	src/filters/letkf_analysis.f90 File Reference	55
5.14.1	Function/Subroutine Documentation	55
5.14.1.1	letkf_analysis	55
5.15	src/filters/proposal_filter.f90 File Reference	56
5.15.1	Function/Subroutine Documentation	56
5.15.1.1	proposal_filter	56
5.16	src/filters/sir_filter.f90 File Reference	57
5.16.1	Function/Subroutine Documentation	57
5.16.1.1	sir_filter	57
5.17	src/filters/stochastic_model.f90 File Reference	58
5.17.1	Function/Subroutine Documentation	58
5.17.1.1	check_scaling	58
5.17.1.2	stochastic_model	58
5.18	src/operations/gen_rand.f90 File Reference	59
5.18.1	Function/Subroutine Documentation	59
5.18.1.1	mixturerandomnumbers1d	59
5.18.1.2	mixturerandomnumbers2d	60
5.18.1.3	normalrandomnumbers1d	61
5.18.1.4	normalrandomnumbers2d	62
5.18.1.5	random_seed_mpi	62
5.18.1.6	uniformrandomnumbers1d	63
5.19	src/operations/operator_wrappers.f90 File Reference	63
5.19.1	Function/Subroutine Documentation	64
5.19.1.1	bprime	64
5.19.1.2	innerhqht_plus_r_1	64

5.19.1.3	innerr_1	65
5.19.1.4	k	66
5.20	src/operations/perturb_particle.f90 File Reference	67
5.20.1	Function/Subroutine Documentation	68
5.20.1.1	perturb_particle	68
5.20.1.2	update_state	68
5.21	src/operations/resample.f90 File Reference	69
5.21.1	Function/Subroutine Documentation	69
5.21.1.1	resample	69
5.22	src/tests/alltests.f90 File Reference	70
5.22.1	Function/Subroutine Documentation	70
5.22.1.1	alltests	70
5.23	src/tests/test_h.f90 File Reference	71
5.23.1	Function/Subroutine Documentation	71
5.23.1.1	test_h	71
5.24	src/tests/test_hqhtr.f90 File Reference	71
5.24.1	Function/Subroutine Documentation	71
5.24.1.1	test_hqhtr	71
5.25	src/tests/test_q.f90 File Reference	72
5.25.1	Function/Subroutine Documentation	72
5.25.1.1	test_q	72
5.26	src/tests/test_r.f90 File Reference	72
5.26.1	Function/Subroutine Documentation	73
5.26.1.1	test_r	73
5.27	src/tests/tests.f90 File Reference	73
5.27.1	Function/Subroutine Documentation	73
5.27.1.1	h_tests	73
5.27.1.2	hqhtr_tests	74
5.27.1.3	q_tests	75
5.27.1.4	r_tests	75
5.28	src/utils/comms.f90 File Reference	76
5.29	src/utils/data_io.f90 File Reference	76
5.29.1	Function/Subroutine Documentation	76
5.29.1.1	get_observation_data	76
5.29.1.2	output_from_pf	77
5.29.1.3	save_observation_data	77
5.29.1.4	save_truth	78
5.30	src/utils/diagnostics.f90 File Reference	78
5.30.1	Function/Subroutine Documentation	78
5.30.1.1	diagnostics	78

5.30.1.2	trajectories	79
5.31	src/utls/genQ.f90 File Reference	79
5.31.1	Function/Subroutine Documentation	79
5.31.1.1	genq	80
5.32	src/utls/histogram.f90 File Reference	80
5.33	src/utls/quicksort.f90 File Reference	80
5.33.1	Function/Subroutine Documentation	80
5.33.1.1	insertionsort_d	80
5.33.1.2	quicksort_d	81
5.34	src/utls/random_d.f90 File Reference	81
Index		82

Chapter 1

EMPIRE Data Assimilation Documentation

Author

Philip A. Browne p.browne@reading.ac.uk

Date

Time-stamp: <2014-09-26 18:02:48 pbrowne>

1.1 Downloading

These codes are hosted on www.bitbucket.org and can be attained with the following commands:

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

Copyright

These codes are distributed under the GNU GPL 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

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

To compile the source code, simply then type the command

```
make
```

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

- [empire](#)
- [alltests](#)
- [test_h](#)
- [test_hqhtr](#)
- [test_q](#)
- [test_r](#)

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

```
make clean
```

1.2.2 Compilation of the documentation

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

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

```
make docs
```

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

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

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

```
make doc_html
```

1.3 Customising for specific models

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

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

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

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:

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

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

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 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	12
pf_control	Module pf_control holds all the information to control the the main program	12
pf_control::pf_control_type	16
qdata	Module as a place to store user specified data for Q	20
random	A module for random number generation from the following distributions:	22
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	32

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

model_specific.f90	35
src/controllers/old_pf_couple.f90	47
src/controllers/pf_control.f90	47
src/controllers/pf_couple.f90	47
src/controllers/pf_parameters.dat	48
src/controllers/sizes.f90	50
src/data/Qdata.f90	50
src/data/Rdata.f90	50
src/filters/eakf_analysis.f90	50
src/filters/enkf_specific.f90	51
src/filters/ekf_analysis.f90	53
src/filters/etkf_analysis.f90	54
src/filters/letkf_analysis.f90	55
src/filters/proposal_filter.f90	56
src/filters/sir_filter.f90	57
src/filters/stochastic_model.f90	58
src/operations/gen_rand.f90	59
src/operations/operator_wrappers.f90	63
src/operations/perturb_particle.f90	67
src/operations/resample.f90	69
src/tests/alltests.f90	70
src/tests/test_h.f90	71
src/tests/test_hqhtr.f90	71
src/tests/test_q.f90	72
src/tests/test_r.f90	72
src/tests/tests.f90	73
src/utills/comms.f90	76
src/utills/data_io.f90	76
src/utills/diagnostics.f90	78
src/utills/genQ.f90	79
src/utills/histogram.f90	80
src/utills/quicksort.f90	80
src/utills/random_d.f90	81

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](#)
- integer [mype_id](#)
- integer [myrank](#)
- integer [nproc](#)
- integer [pf_mpi_comm](#)
- integer [pfrank](#)
- integer [npfs](#)
- integer, dimension(:), allocatable [gblcount](#)
- integer, dimension(:), allocatable [gbldisp](#)

4.1.1 Detailed Description

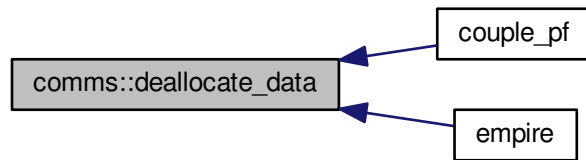
Module containing EMPIRE coupling data.

4.1.2 Member Function/Subroutine Documentation

4.1.2.1 subroutine `comms::allocate_data` ()

4.1.2.2 subroutine `comms::deallocate_data ()`

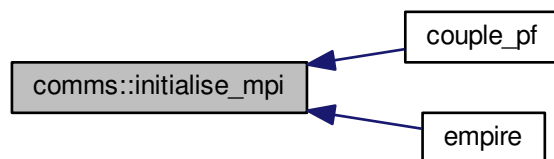
Here is the caller graph for this function:



4.1.2.3 subroutine `comms::initialise_mpi ()`

subroutine to make EMPIRE connections and saves details into [pf_control](#) module

Here is the caller graph for this function:



4.1.3 Member Data Documentation

4.1.3.1 integer `comms::cpl_mpi_comm`

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

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

4.1.3.4 integer `comms::mype_id`

4.1.3.5 integer `comms::myrank`

4.1.3.6 integer `comms::npfs`

4.1.3.7 integer `comms::nproc`

4.1.3.8 integer `comms::pf_mpi_comm`

4.1.3.9 integer comms::pfrank

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.

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

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

4.2.3 Member Data Documentation

4.2.3.1 integer, dimension(:), allocatable histogram_data::rank_hist_list

4.2.3.2 integer, dimension(:), allocatable histogram_data::rank_hist_nums

4.2.3.3 integer histogram_data::rhl_n

4.2.3.4 integer histogram_data::rhn_n

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

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

4.3 hqht_plus_r Module Reference

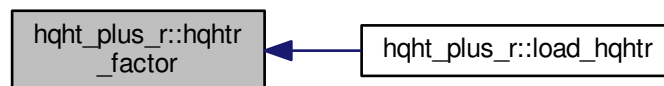
Public Member Functions

- subroutine [load_hqhtr](#)
- subroutine [hqhtr_factor](#)
- subroutine [kill_hqhtr](#)

4.3.1 Member Function/Subroutine Documentation

4.3.1.1 subroutine hqht_plus_r::hqhtr_factor ()

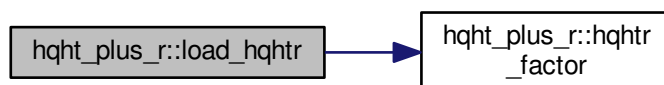
Here is the caller graph for this function:



4.3.1.2 subroutine hqht_plus_r::kill_hqhtr ()

4.3.1.3 subroutine hqht_plus_r::load_hqhtr ()

Here is the call graph for this function:



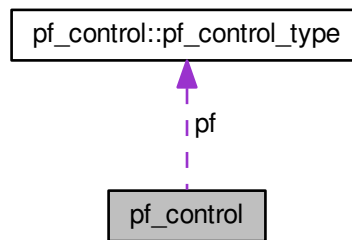
The documentation for this module was generated from the following file:

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

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\(pf_control_type\) pf](#)
the derived data type holding all controlling data

4.4.1 Detailed Description

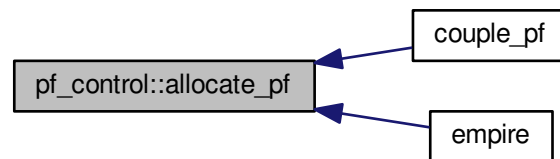
module [pf_control](#) holds all the information to control the the main program

4.4.2 Member Function/Subroutine Documentation

4.4.2.1 subroutine pf_control::allocate_pf ()

subroutine to allocate space for the filtering code

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

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:

- [nudgefacs](#)
- [nfacs](#)
- [ufacs](#)
- [Qscale](#)
- [keep](#)
- [rho](#)
- [len](#)

2 Characters:

- [type](#)

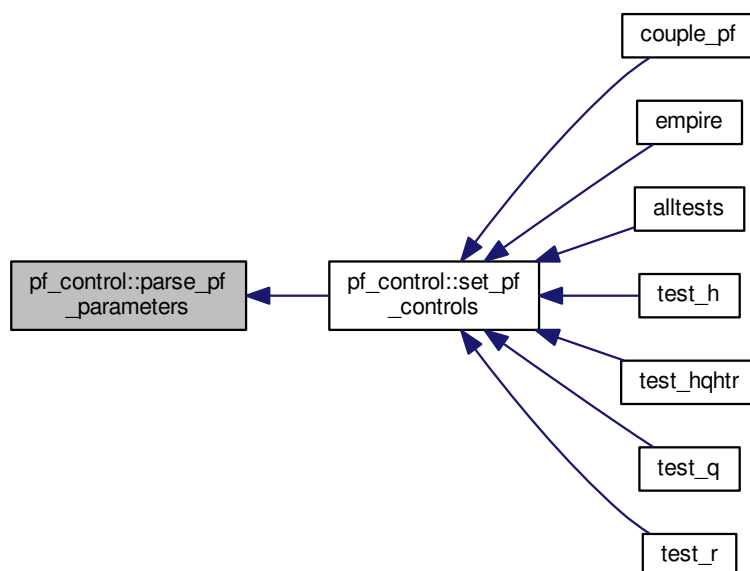
1 Character:

- [init](#)

Logicals:

- [gen_Q](#)
- [gen_data](#)
- [use_talagrand](#)
- [use_weak](#)
- [use_var](#)
- [use_traj](#)
- [use_rmse](#)
- [human_readable](#)

Here is the caller graph for this function:



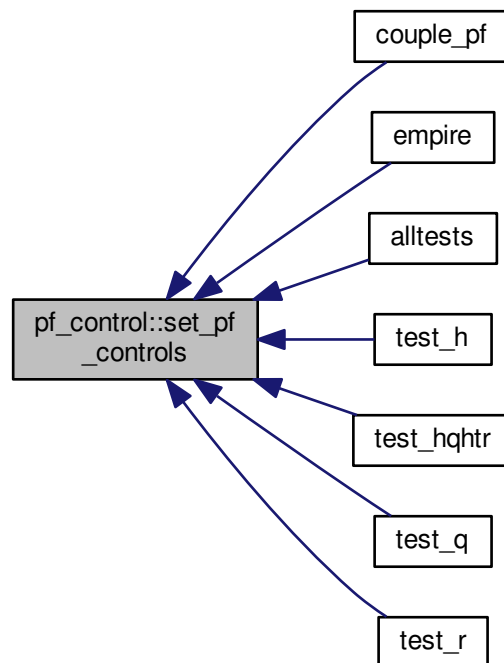
4.4.2.4 subroutine `pf_control::set_pf_controls ()`

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

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) pf_control::pf`

the derived data type holding all controlling data

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

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

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
- character(1) `init`
which method to initialise ensemble

4.5.1 Member Data Documentation

4.5.1.1 integer `pf_control::pf_control_type::count`

number of ensemble members associated with this MPI process

4.5.1.2 integer `pf_control::pf_control_type::couple_root`

empire master processor

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

4.5.1.4 logical `pf_control::pf_control_type::gen_data`

true generates synthetic obs for a twin experiment

4.5.1.5 logical `pf_control::pf_control_type::gen_q`

true attempts to build up Q from long model run

4.5.1.6 logical `pf_control::pf_control_type::human_readable`

unused

4.5.1.7 character(1) `pf_control::pf_control_type::init`

which method to initialise ensemble

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

proportion of particles to keep in EWPF EW step

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

R localisation length scale.

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

mean state vector

4.5.1.11 `integer pf_control::pf_control_type::nens`

the total number of ensemble members

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

standard deviation of normal distribution in mixture density

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

the nudging factor

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

particles associates with this MPI process

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

state vector of ensemble members on this mpi process

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

scalar to multiply Q by

4.5.1.17 `real(kind=kind(1.0d0)) pf_control::pf_control_type::rho`

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

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

storage for rank histograms

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

dunno

4.5.1.20 `integer pf_control::pf_control_type::time_bwn_obs`

the number of model timesteps between observations

4.5.1.21 `integer pf_control::pf_control_type::time_obs`

the number of observations we will assimilate

4.5.1.22 integer pf_control::pf_control_type::timestep =0

the current timestep as the model progresses

4.5.1.23 character(2) pf_control::pf_control_type::type

which filter to use

4.5.1.24 real(kind=kind(1.0d0)) pf_control::pf_control_type::ufac

half width of the uniform distribution in mixture density

4.5.1.25 logical pf_control::pf_control_type::use_mean

switch if true outputs ensemble mean

4.5.1.26 logical pf_control::pf_control_type::use_rmse

switch if true outputs Root Mean Square Errors

4.5.1.27 logical pf_control::pf_control_type::use_talagrand

switch if true outputs rank histograms

4.5.1.28 logical pf_control::pf_control_type::use_traj

switch if true outputs trajectories

4.5.1.29 logical pf_control::pf_control_type::use_var

switch if true outputs ensemble variance

4.5.1.30 logical pf_control::pf_control_type::use_weak

switch unused

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

the negative log of the weights of the particles

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

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

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

4.6.2 Member Function/Subroutine Documentation

4.6.2.1 subroutine qdata::killq ()

SUbroutine to deallocate user data for Q

4.6.2.2 subroutine qdata::loadq ()

Subroutine to load in user data for Q.

4.6.3 Member Data Documentation

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

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

4.6.3.3 integer qdata::qn

4.6.3.4 integer qdata::qne

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

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

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

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

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

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:

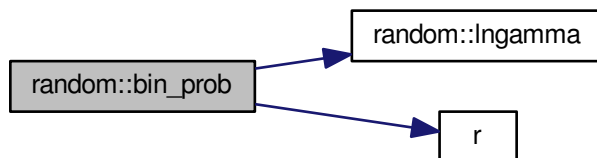
Distribution Function/subroutine name

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](#) Generalized inverse Gaussian [random_inv_gauss](#) Poisson [random_Poisson](#) Binomial [random_binomial1](#) * [random_binomial2](#) * Negative binomial [random_neg_binomial](#) von Mises [random_von_Mises](#) Cauchy [random_Cauchy](#)

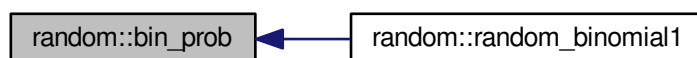
4.7.2 Member Function/Subroutine Documentation

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)`

Here is the call graph for this function:

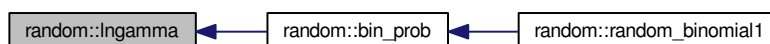


Here is the caller graph for this function:



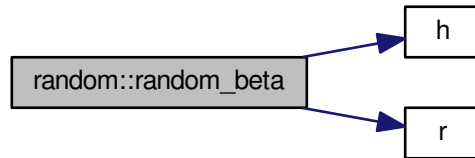
4.7.2.2 `real (dp) function random::lngamma (real (dp), intent(in) x)`

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)`

Here is the call graph for this function:



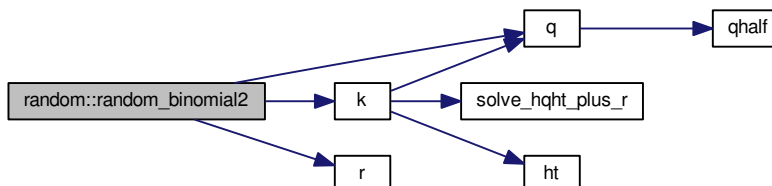
4.7.2.4 `integer function random::random_binomial1 (integer, intent(in) n, real(kind=kind(1.0d+0)), intent(in) p, logical, intent(in) first)`

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)`

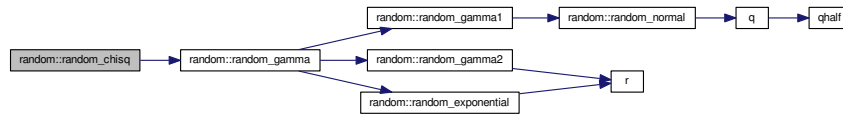
Here is the call graph for this function:



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

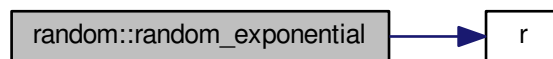
4.7.2.7 `real(kind=kind(1.0d+0)) function random::random_chisq (integer, intent(in) ndf, logical, intent(in) first)`

Here is the call graph for this function:

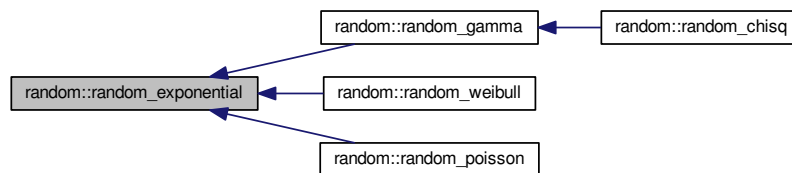


4.7.2.8 `real(kind=kind(1.0d+0)) function random::random_exponential ()`

Here is the call graph for this function:

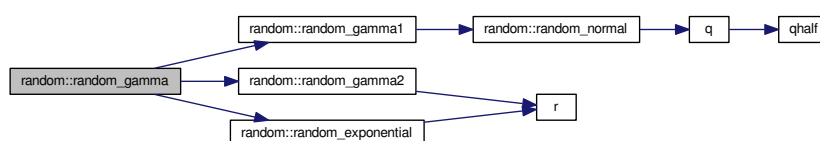


Here is the caller graph for this function:



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)`

Here is the call graph for this function:

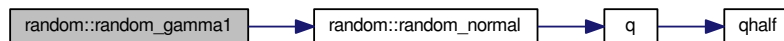


Here is the caller graph for this function:

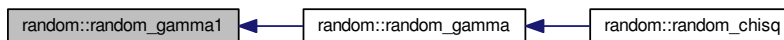


4.7.2.10 `real(kind=kind(1.0d+0)) function random::random_gamma1 (real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first)`

Here is the call graph for this function:

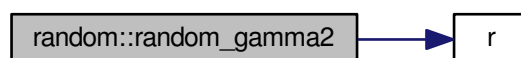


Here is the caller graph for this function:

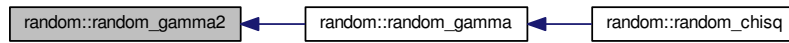


4.7.2.11 `real(kind=kind(1.0d+0)) function random::random_gamma2 (real(kind=kind(1.0d+0)), intent(in) s, logical, intent(in) first)`

Here is the call graph for this function:

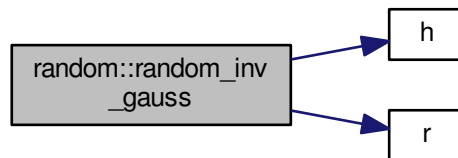


Here is the caller graph for this function:



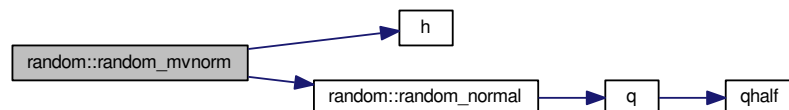
4.7.2.12 `real(kind=kind(1.0d+0)) function random::random_inv_gauss (real(kind=kind(1.0d+0)), intent(in) h,
real(kind=kind(1.0d+0)), intent(in) b, logical, intent(in) first)`

Here is the call graph for this function:



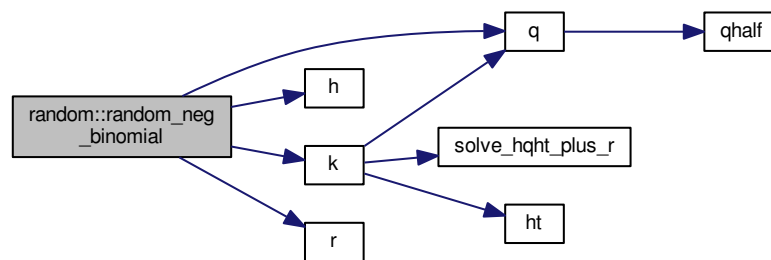
4.7.2.13 `subroutine random::random_mvnorm (integer, intent(in) n, real(kind=kind(1.0d+0)), dimension(:), intent(in) h,
real(kind=kind(1.0d+0)), dimension(:), intent(in) d, real(kind=kind(1.0d+0)), dimension(:), intent(inout) f, logical,
intent(in) first, real(kind=kind(1.0d+0)), dimension(:), intent(out)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

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`)

Here is the call graph for this function:



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

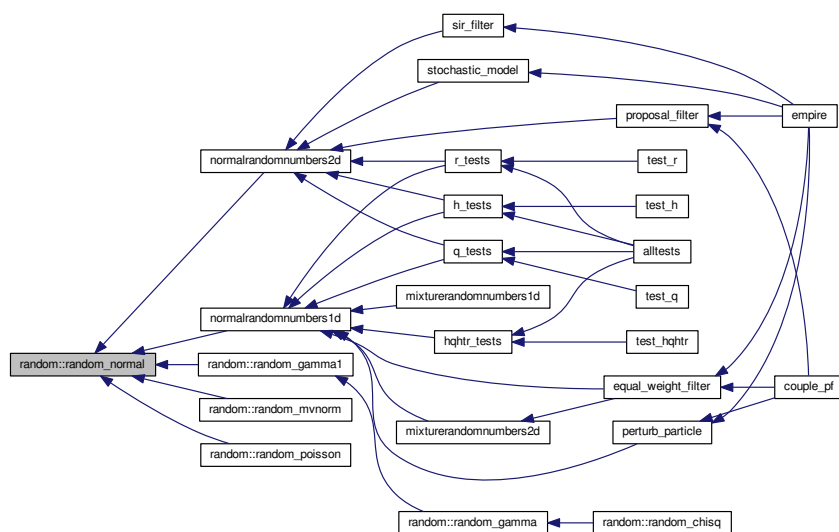
Returns

`fn_val`

Here is the call graph for this function:

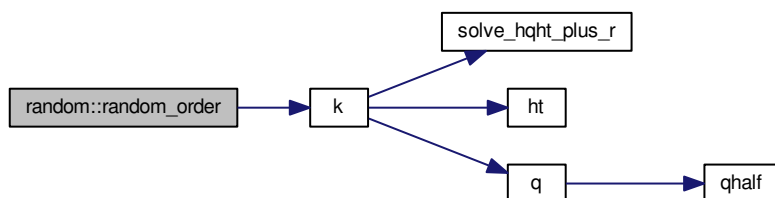


Here is the caller graph for this function:



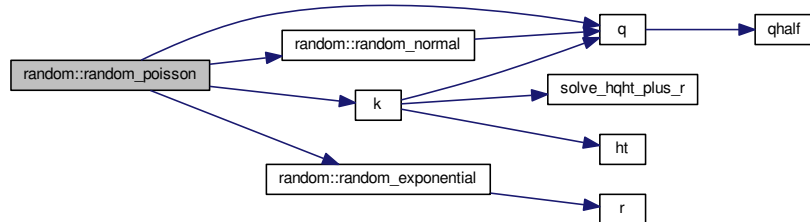
4.7.2.16 subroutine random::random_order (integer, dimension(n), intent(out) order, integer, intent(in) n)

Here is the call graph for this function:



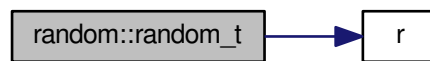
4.7.2.17 integer function random::random_poisson (real(kind=kind(1.0d+0)), intent(in) *mu*, logical, intent(in) *first*)

Here is the call graph for this function:



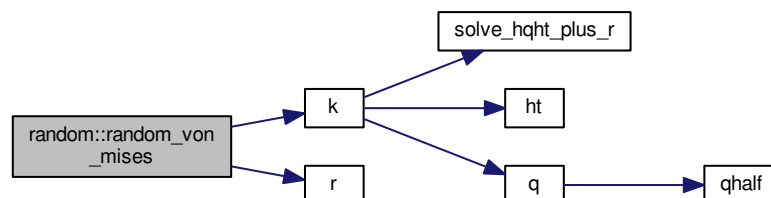
4.7.2.18 real(kind=kind(1.0d+0)) function random::random_t (integer, intent(in) *m*)

Here is the call graph for this function:



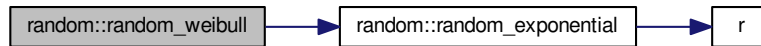
4.7.2.19 real(kind=kind(1.0d+0)) function random::random_von_mises (real(kind=kind(1.0d+0)), intent(in) *k*, logical, intent(in) *first*)

Here is the call graph for this function:



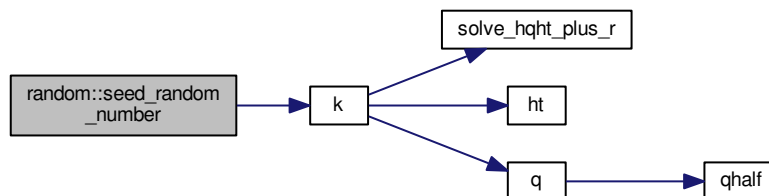
4.7.2.20 `real(kind=kind(1.0d+0)) function random::random_weibull (real(kind=kind(1.0d+0)), intent(in) a)`

Here is the call graph for this function:



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

Here is the call graph for this function:



4.7.3 Member Data Documentation

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

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

- [src/Utils/random_d.f90](#)

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.

4.8.2 Member Function/Subroutine Documentation

4.8.2.1 subroutine rdata::killr ()

SUbroutine to deallocate R data

4.8.2.2 subroutine rdata::loadr ()

Subroutine to load data for R.

4.8.3 Member Data Documentation

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

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

4.8.3.3 integer rdata::rn

4.8.3.4 integer rdata::rne

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

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

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

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

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.

4.9.2 Member Data Documentation

4.9.2.1 integer sizes::obs_dim

size of the observation space

4.9.2.2 integer sizes::state_dim

dimension of the model

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

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

Chapter 5

File Documentation

5.1 model_specific.f90 File Reference

Functions/Subroutines

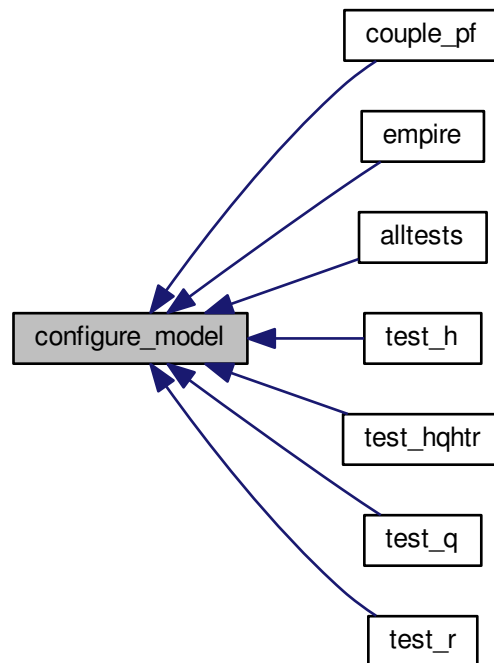
- subroutine `configure_model`
subroutine called initially to set up details and data for model specific functions
- subroutine `solve_r` (obsDim, nrhs, y, v, t)
subroutine to take an observation vector y and return v in observation space.
- subroutine `solve_rhalf` (obsdim, nrhs, y, v, t)
subroutine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

5.1.1 Function/Subroutine Documentation

5.1.1.1 subroutine `configure_model` ()

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

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*)

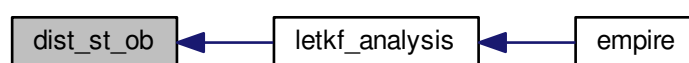
subroutine to compute the distance between the variable in the state vector and the variable in the observations

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

Parameters

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

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*)

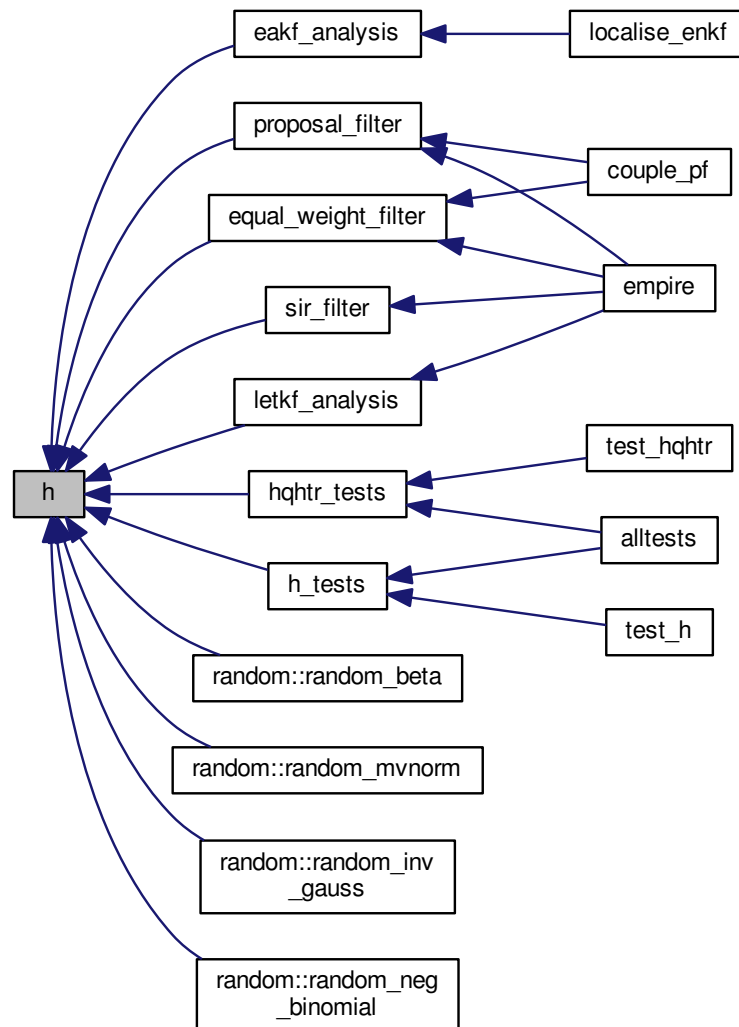
subroutine to take a full state vector x and return $H(x)$ in observation space.

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

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*)

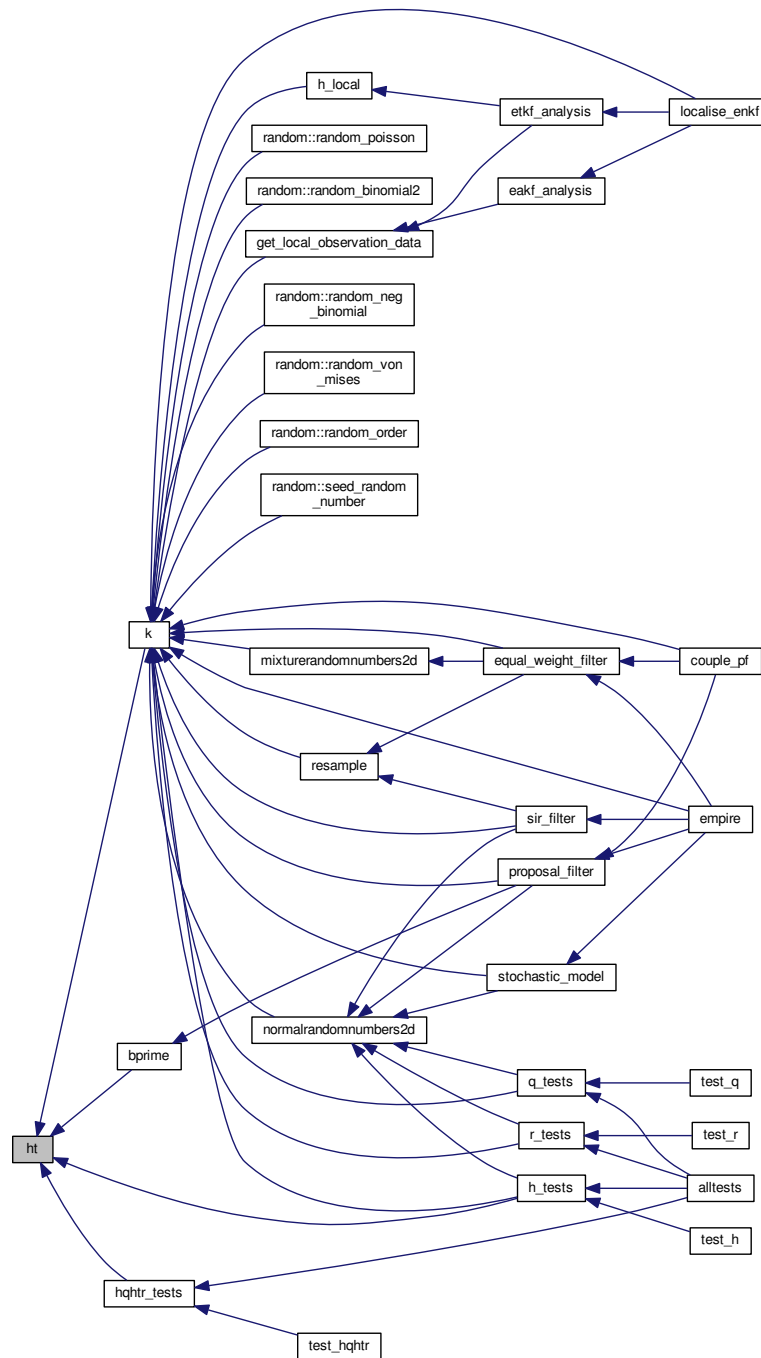
subroutine to take an observation vector y and return $x = H^T(y)$ in full state space.

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

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

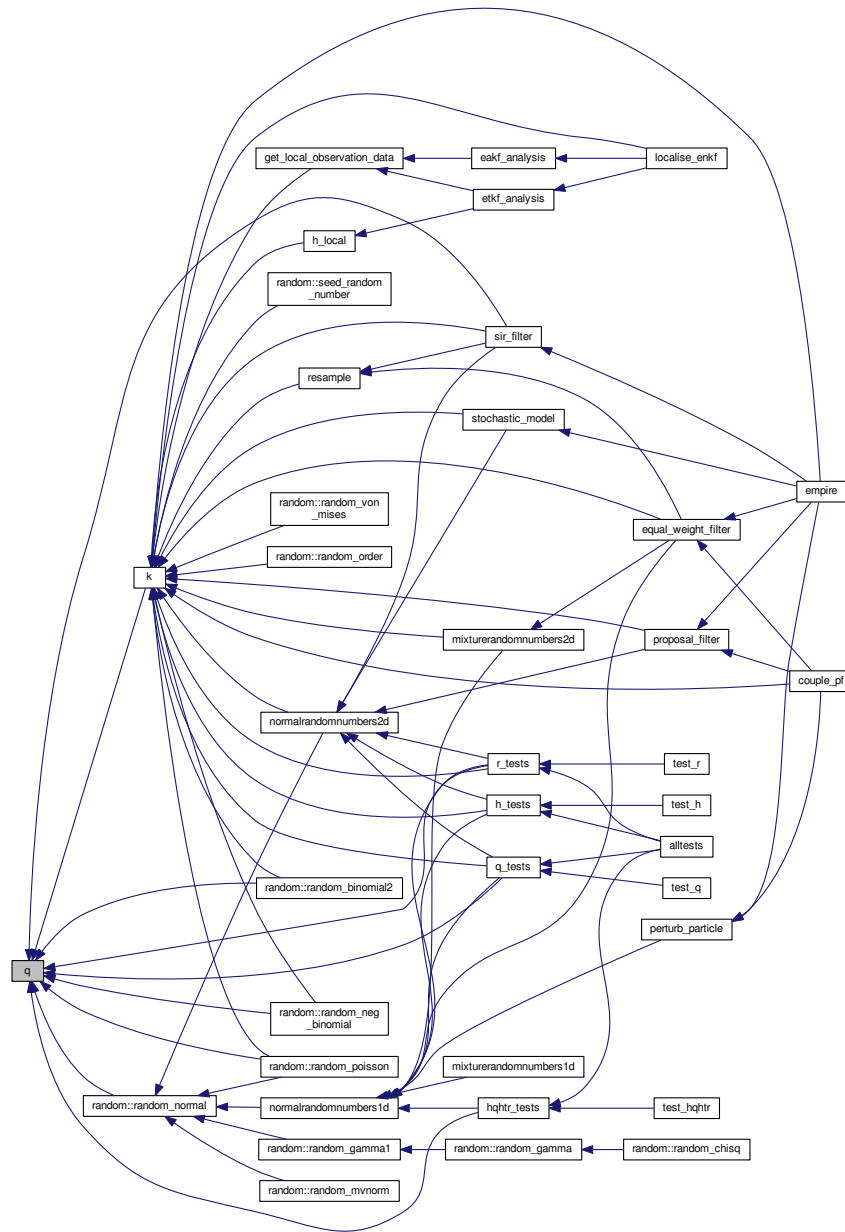
Parameters

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 $Qx = Qx$

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)

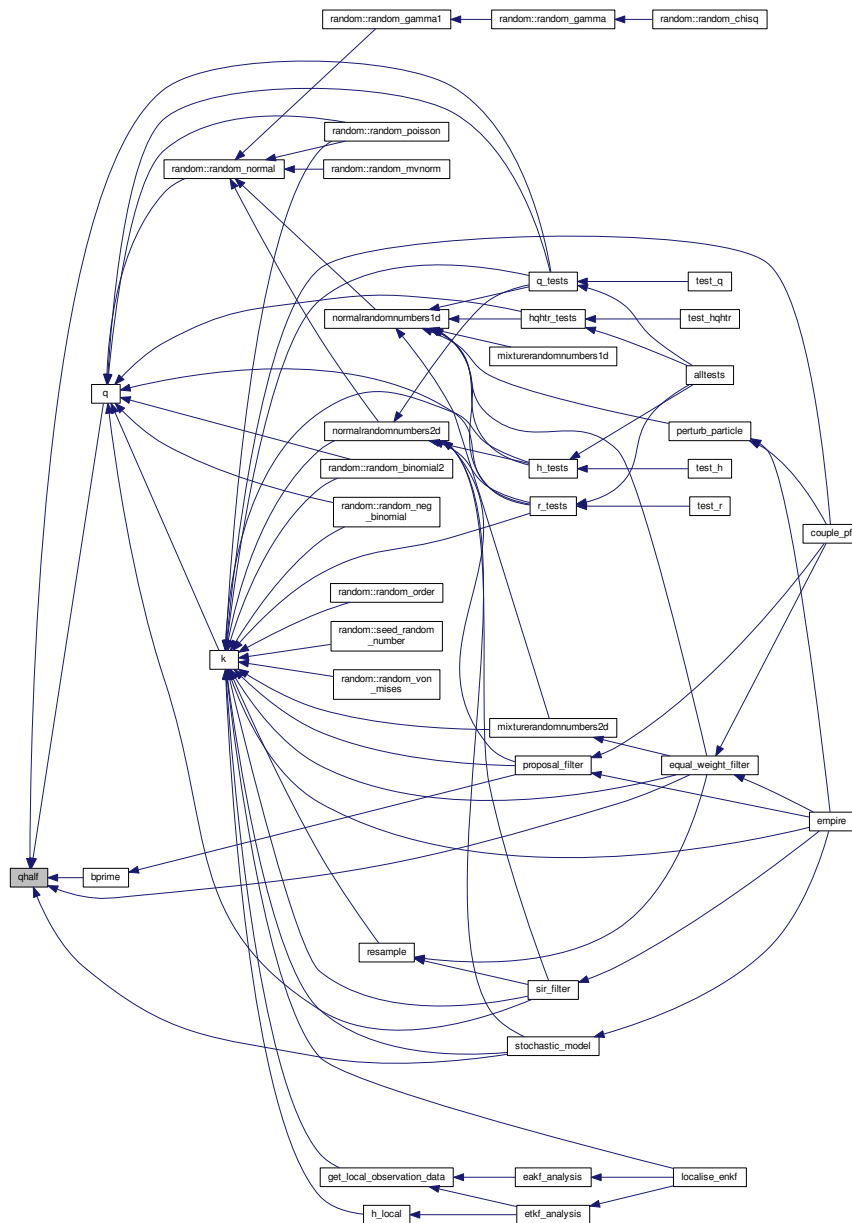
subroutine to take a full state vector x and return $Q^{1/2}x$ in state space.

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

Parameters

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

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*)

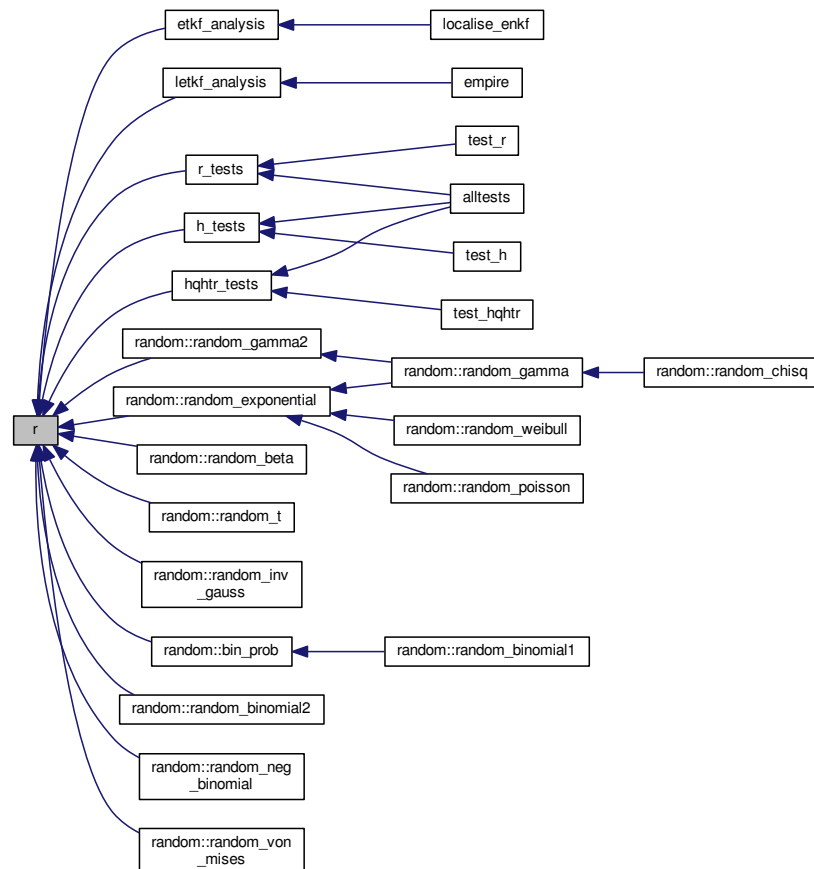
subroutine to take an observation vector *x* and return *Rx* in observation space.

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

Here is the caller graph for this function:



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

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

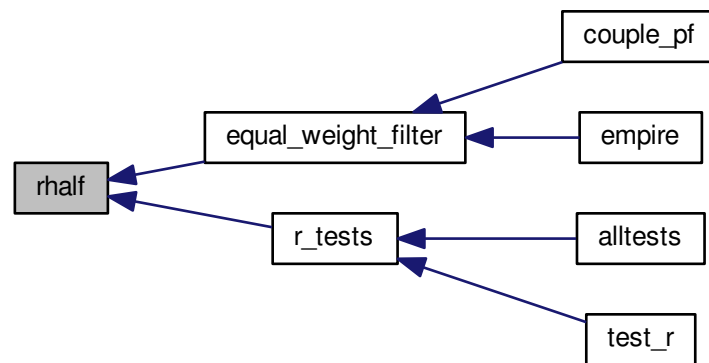
Given *y* compute $R^{\frac{1}{2}}y$

Parameters

in	<i>obsdim</i>	the dimension of the observations
in	<i>nrhs</i>	the number of right hand sides

in	y	the input vector
out	ry	the resulting vector where $Ry = R^{\frac{1}{2}}y$
in	t	the timestep

Here is the caller graph for this function:



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

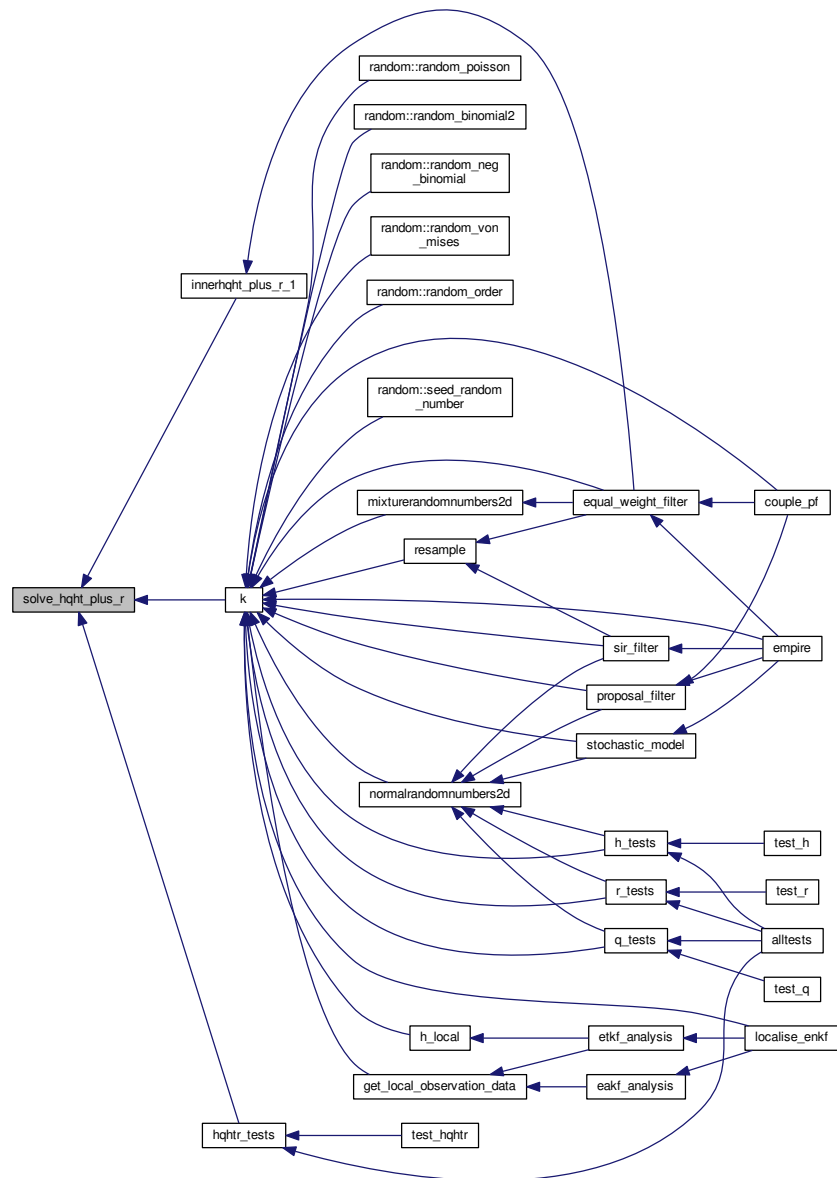
subroutine to take an observation vector y and return v in observation space.

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

Parameters

in	$obsdim$	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

Here is the caller graph for this function:



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

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

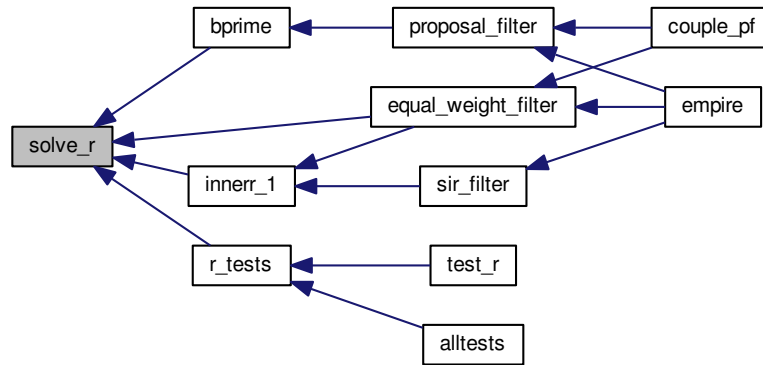
Given *y* find *v* such that $Rv = y$

Parameters

<i>in</i>	<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

Here is the caller graph for this function:



5.1.1.11 subroutine `solve_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) *v*, integer, intent(in) *t*)

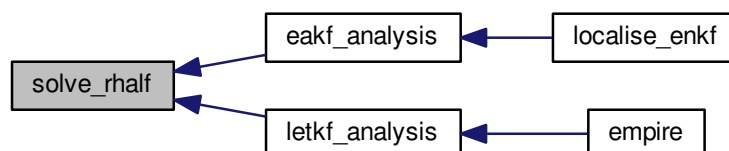
subroutine to take an observation vector *y* and return *v* in observation space.

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

Here is the caller graph for this function:



5.2 src/controlers/old_pf_couple.f90 File Reference

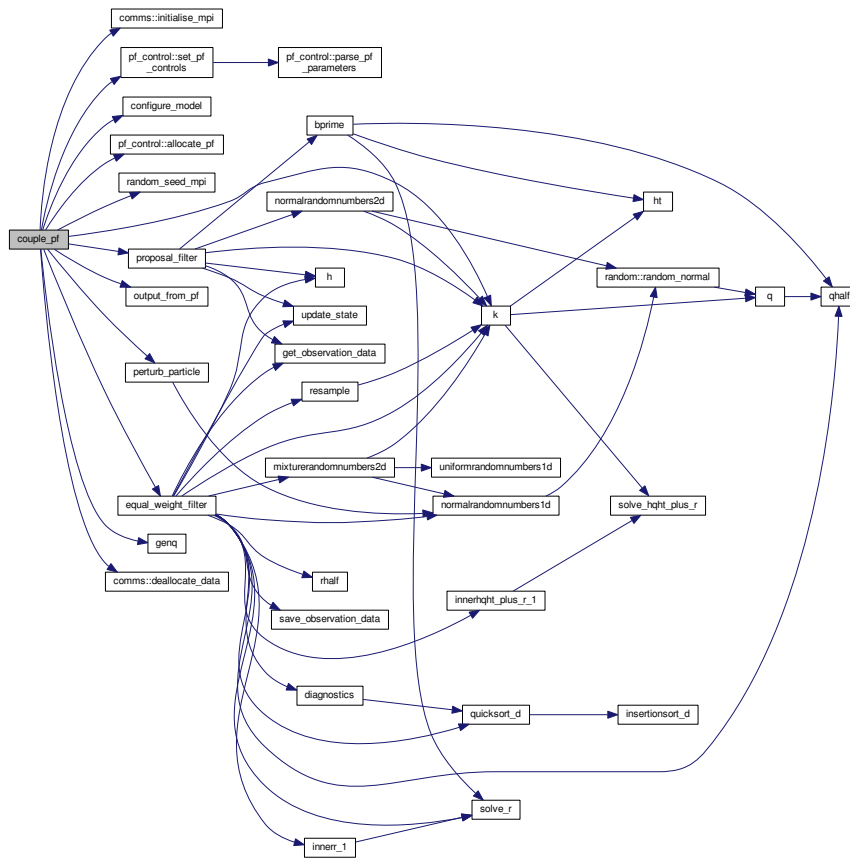
Functions/Subroutines

- program [couple_pf](#)

5.2.1 Function/Subroutine Documentation

5.2.1.1 program couple_pf ()

Here is the call graph for this function:



5.3 src/controlers/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](#)

5.4 src/controlers/pf_couple.f90 File Reference

Functions/Subroutines

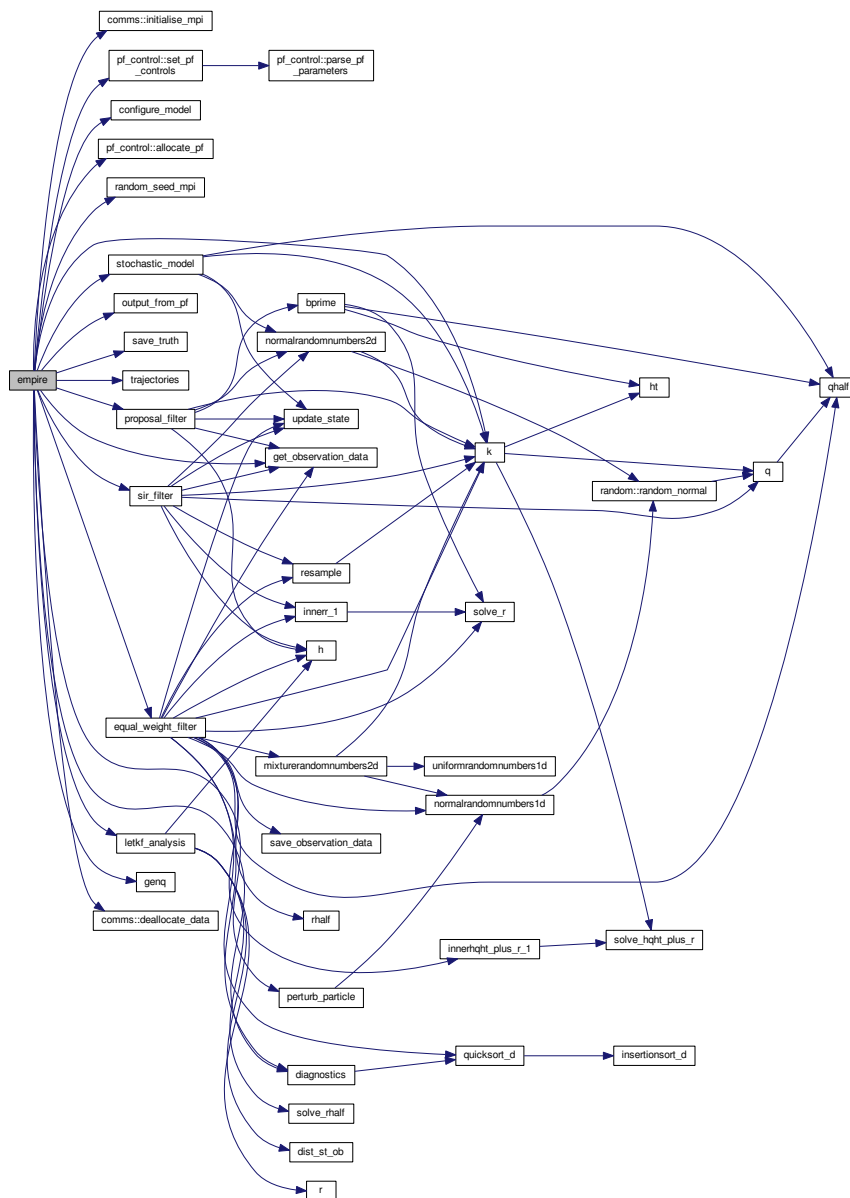
- program [empire](#)
the main program

5.4.1 Function/Subroutine Documentation

5.4.1.1 program `empire` ()

the main program

Here is the call graph for this function:



5.5 src/controllers/pf_parameters.dat File Reference

Variables

- &pf_params `time_obs` =10
- &pf_params `time_bwn_obs` =72
- &pf_params `nudgefac` =0.5D3
- &pf_params `gen_data` =.false.
- &pf_params `nfac` =1.0D-5
- &pf_params `ufac` =1.0D-5
- &pf_params `keep` =0.95D0
- &pf_params `Qscale` =1.0D3
- &pf_params `human_readable` =1.0D3
- &pf_params `use_talagrand` =.true.
- &pf_params `use_weak` =.false.
- &pf_params `use_mean` =.false.
- &pf_params `use_var` =.false.
- &pf_params `use_rmse` =.true.
- &pf_params `gen_Q` =.false.
- &pf_params `use_traj` =.true.
- &pf_params `type` ='EW'

5.5.1 Variable Documentation

5.5.1.1 & pf_params `gen_data` =.false.

5.5.1.2 & pf_params `gen_Q` =.false.

5.5.1.3 & pf_params `human_readable` =1.0D3

5.5.1.4 & pf_params `keep` =0.95D0

5.5.1.5 & pf_params `nfac` =1.0D-5

5.5.1.6 & pf_params `nudgefac` =0.5D3

5.5.1.7 & pf_params `Qscale` =1.0D3

5.5.1.8 & pf_params `time_bwn_obs` =72

5.5.1.9 & pf_params `time_obs` =10

5.5.1.10 & pf_params `type` ='EW'

5.5.1.11 & pf_params `ufac` =1.0D-5

5.5.1.12 & pf_params `use_mean` =.false.

5.5.1.13 & pf_params `use_rmse` =.true.

5.5.1.14 & pf_params `use_talagrand` =.true.

5.5.1.15 & pf_params `use_traj` =.true.

5.5.1.16 & pf_params `use_var` =.false.

5.5.1.17 & pf_params `use_weak` =.false.

5.6 src/controlers/sizes.f90 File Reference

Data Types

- module [sizes](#)
Module that stores the dimension of observation and state spaces.

5.7 src/data/Qdata.f90 File Reference

Data Types

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

5.8 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](#)

5.9 src/DOC_README.txt File Reference

5.10 src/filters/eakf_analysis.f90 File Reference

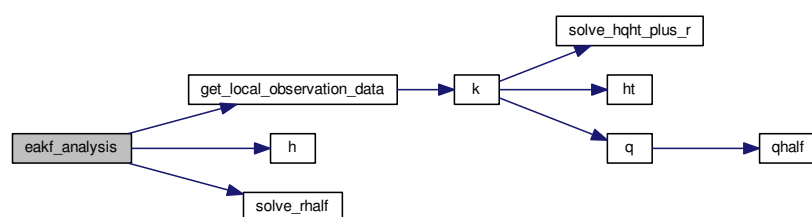
Functions/Subroutines

- 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*)

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

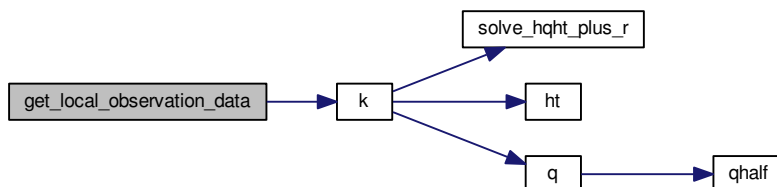
Functions/Subroutines

- 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)

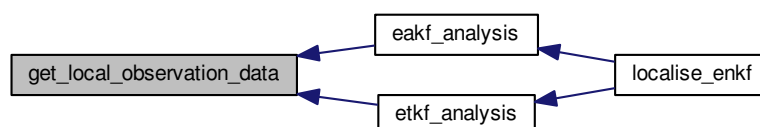
5.11.1 Function/Subroutine Documentation

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*)

Here is the call graph for this function:

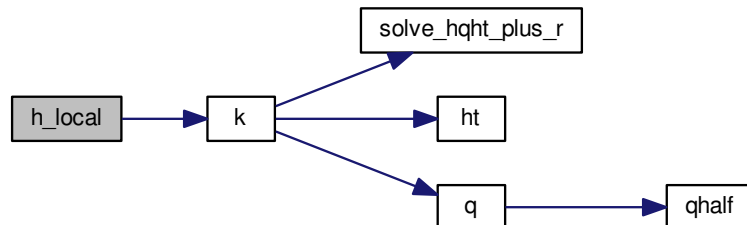


Here is the caller graph for this function:

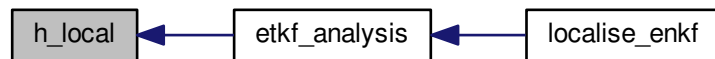


5.11.1.2 subroutine `h_local` (integer, intent(in) *num_hor*, integer, intent(in) *num_ver*, integer, intent(in) *this_hor*, integer, intent(in) *this_ver*, integer, intent(in) *boundary*, integer, intent(in) *nrhs*, integer, intent(in) *stateDim*, real(kind=rk), dimension(*statedim*,*nrhs*), intent(in) *x*, integer, intent(in) *obsDim*, real(kind=rk), dimension(*obsdim*,*nrhs*), intent(out) *y*)

Here is the call graph for this function:

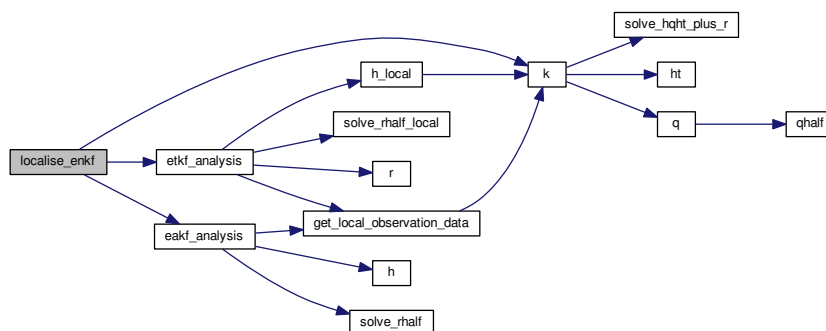


Here is the caller graph for this function:



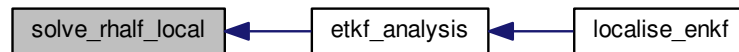
5.11.1.3 subroutine `localise_enkf` (integer, intent(in) *enkf_analysis*)

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)

Here is the caller graph for this function:



5.12 src/filters/equivalent_weights_step.f90 File Reference

Functions/Subroutines

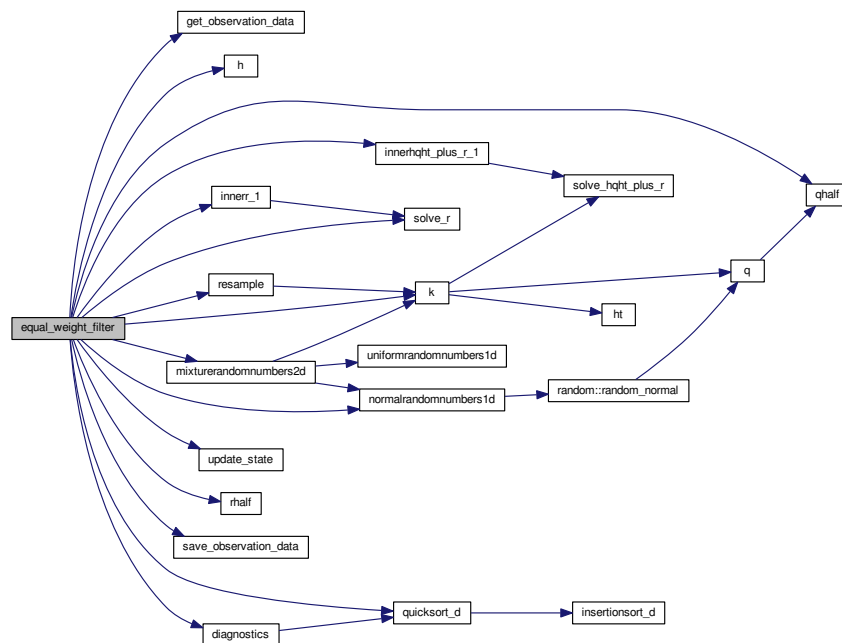
- subroutine [equal_weight_filter](#)
subroutine to do the equivalent weights step

5.12.1 Function/Subroutine Documentation

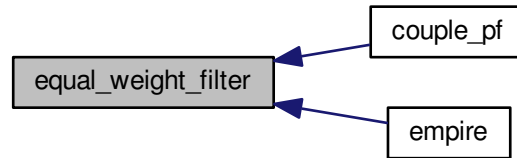
5.12.1.1 subroutine equal_weight_filter ()

subroutine to do the equivalent weights step

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)

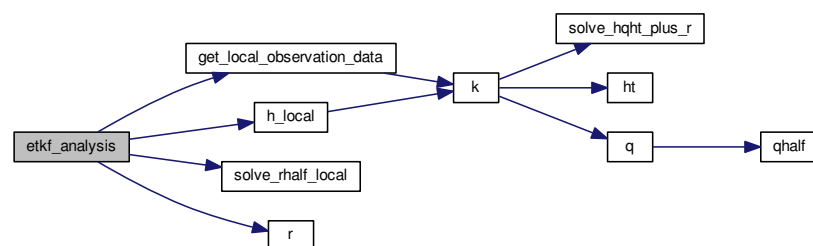
subroutine to perform the ensemble transform Kalman filter

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

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](#) (*x*, *N*, *stateDimension*, *obsDim*, *y*, *rho*, *len*, *t*)

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

5.14.1 Function/Subroutine Documentation

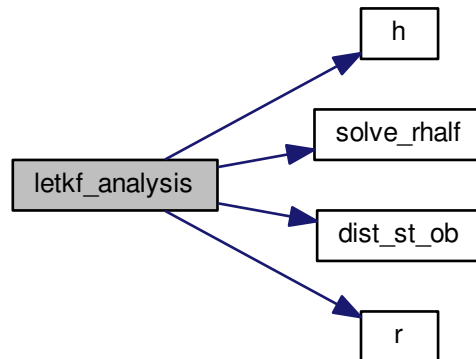
5.14.1.1 subroutine `letkf_analysis` (*real(kind=rk)*, *dimension*(*statedimension*,*n*), *intent*(*inout*) *x*, *integer*, *intent*(*in*) *N*, *integer*, *intent*(*in*) *stateDimension*, *integer*, *intent*(*in*) *obsDim*, *real(kind=rk)*, *dimension*(*obsdim*), *intent*(*in*) *y*, *real(kind=rk)*, *intent*(*in*) *rho*, *real(kind=rk)*, *intent*(*in*) *len*, *integer*, *intent*(*in*) *t*)

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

Parameters

<i>in</i>	<i>n</i>	number of ensemble members
<i>in</i>	<i>statedimension</i>	current size of state dimension
<i>in</i>	<i>obsdim</i>	total number of observations
<i>in, out</i>	<i>x</i>	Forecast ensemble on entry, analysis ensemble on exit
<i>in</i>	<i>y</i>	The observation
<i>in</i>	<i>rho</i>	Inflation parameter; forecast perturbations will be scaled by 1+rho
<i>in</i>	<i>len</i>	Localisation length scale
<i>in</i>	<i>t</i>	the timestep

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](#)

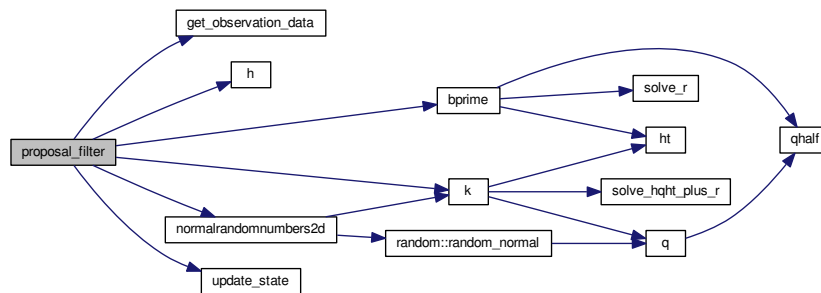
Subroutine to perform nudging in the proposal step of EWPF.

5.15.1 Function/Subroutine Documentation

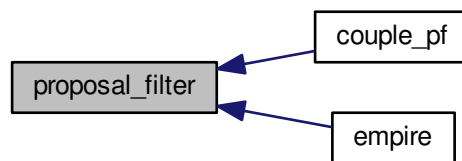
5.15.1.1 subroutine `proposal_filter` ()

Subroutine to perform nudging in the proposal step of EWPF.

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](#)

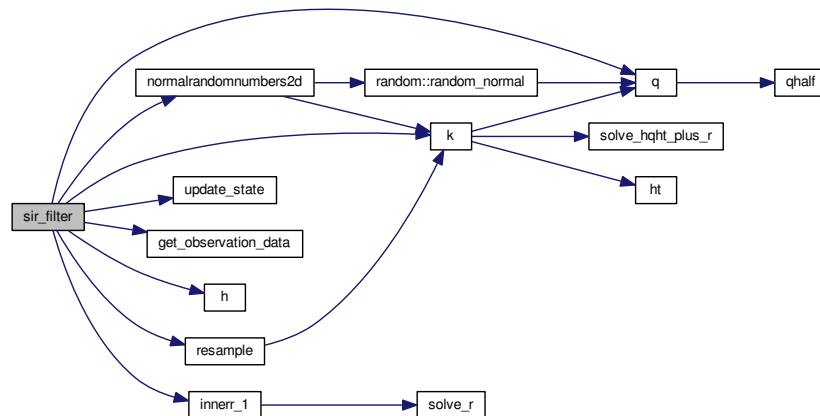
Subroutine to perform SIR filter (Sequential Importance Resampling)

5.16.1 Function/Subroutine Documentation

5.16.1.1 subroutine sir_filter ()

Subroutine to perform SIR filter (Sequential Importance Resampling)

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)

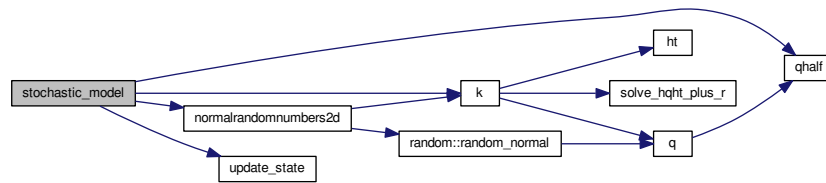
5.17.1 Function/Subroutine Documentation

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)

5.17.1.2 subroutine [stochastic_model](#) ()

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

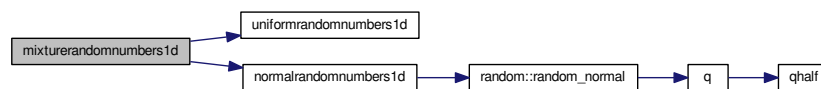
- 5.18.1.1 subroutine [mixturerandomnumbers1d](#) (`real(kind=kind(1.0d0))`, intent(in) *mean*, `real(kind=kind(1.0d0))`, intent(in) *stdev*, `real(kind=kind(1.0d0))`, intent(in) *ufac*, `real(kind=kind(1.0d0))`, intent(in) *epsi*, integer, intent(in) *n*, `real(kind=kind(1.0d0))`, dimension(n), intent(out)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

Here is the call graph for this function:



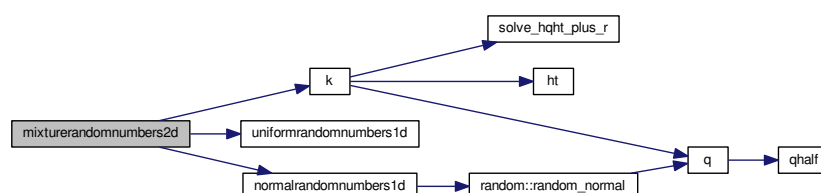
5.18.1.2 subroutine mixturerandomnumbers2d (real(kind=kind(1.0d0)), intent(in) *mean*, real(kind=kind(1.0d0)), intent(in) *stdev*, real(kind=kind(1.0d0)), intent(in) *ufac*, real(kind=kind(1.0d0)), intent(in) *epsi*, integer, intent(in) *n*, integer, intent(in) *k*, real(kind=kind(1.0d0)), dimension(n,k),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

generate two dimensional vector, each drawn from mixture density

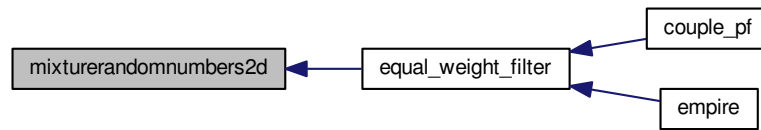
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



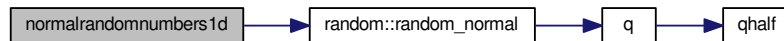
5.18.1.3 subroutine `normalrandomnumbers1d` (`real(kind=rk), intent(in) mean`, `real(kind=rk), intent(in) stdev`, `integer, intent(in) n`, `real(kind=rk), dimension(n), intent(out) phi`)

generate one dimension of Normal random numbers

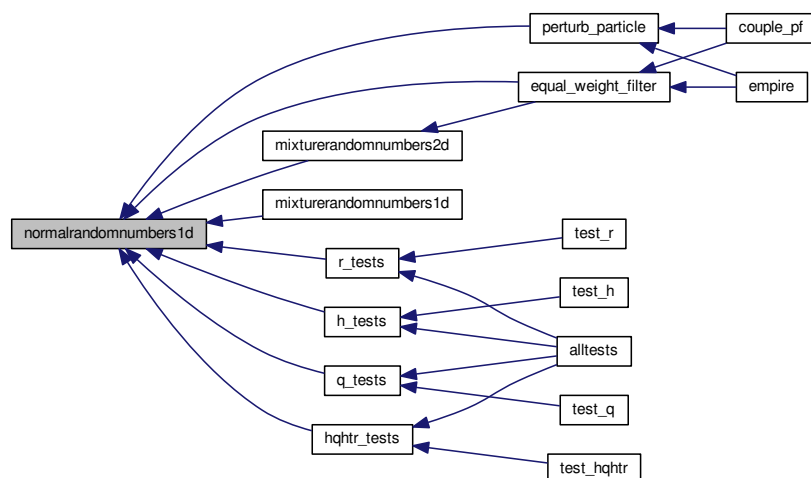
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



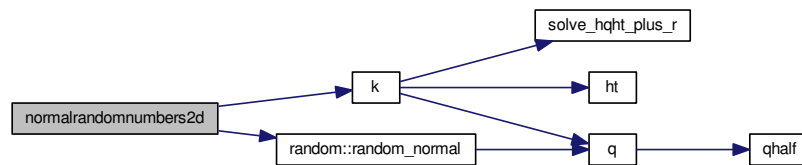
5.18.1.4 subroutine `normalrandomnumbers2d` (`real(kind=rk), intent(in) mean`, `real(kind=rk), intent(in) stdev`, `integer, intent(in) n`, `integer, intent(in) k`, `real(kind=rk), dimension(n,k), intent(out) phi`)

generate two dimensional Normal random numbers

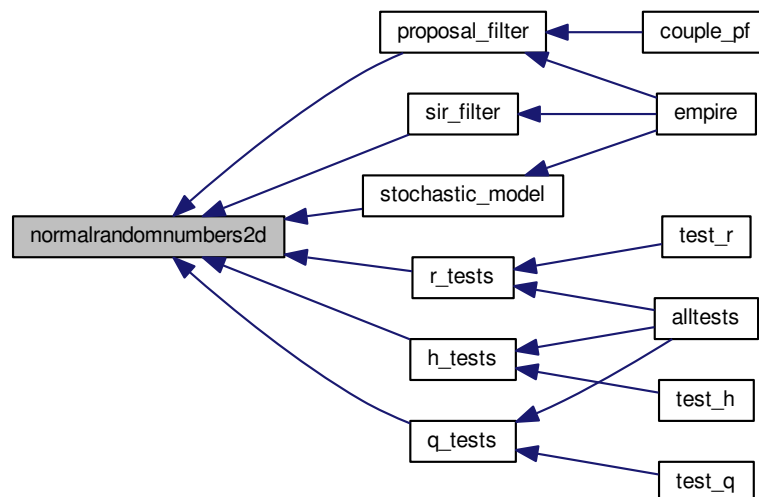
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



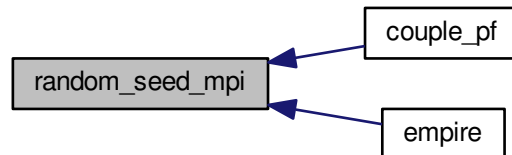
5.18.1.5 subroutine `random_seed_mpi` (`integer, intent(in) pfid`)

Subroutine to set the random seed across MPI threads.

Parameters

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

Here is the caller graph for this function:



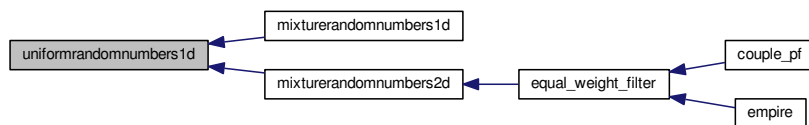
5.18.1.6 subroutine `uniformrandomnumbers1d` (`real(kind=rk)`, intent(in) *minv*, `real(kind=rk)`, intent(in) *maxv*, integer, intent(in) *n*, `real(kind=rk)`, dimension(*n*), intent(out) *phi*)

generate one dimension of uniform random numbers

Parameters

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

Here is the caller graph for this function:



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

5.19.1 Function/Subroutine Documentation

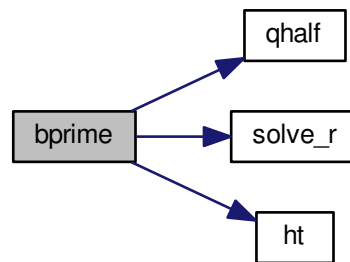
5.19.1.1 subroutine `bprime` (`real(kind=rk)`, `dimension(obs_dim,pf%count)`, `intent(in)` `y`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out)` `x`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(out)` `QHtR_1y`, `real(kind=rk)`, `dimension(state_dim,pf%count)`, `intent(in)`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

subroutine to calculate nudging term and correlated random errors efficiently

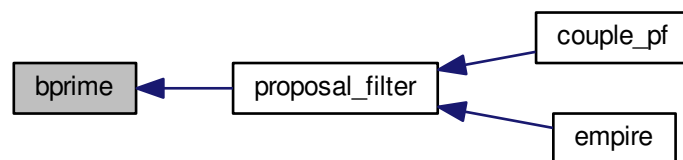
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.19.1.2 subroutine `innerqhht_plus_r_1` (`real(kind=rk)`, `dimension(obs_dim)`, `intent(in)` `y`, `real(kind=rk)`, `intent(out)` `w`)

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

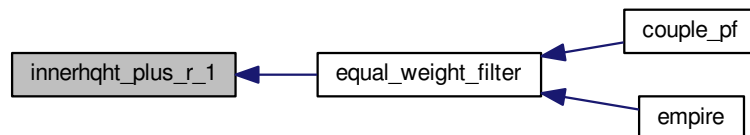
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



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}

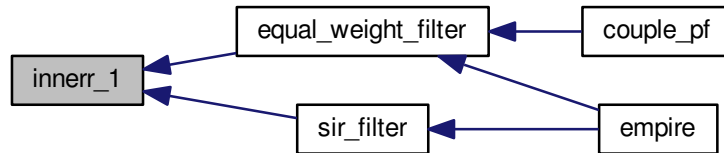
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



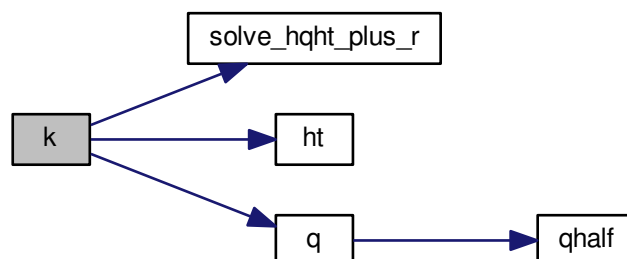
5.19.1.4 subroutine k (real(kind=rk), dimension(obs_dim,pf%count), intent(in) y, real(kind=rk), dimension(state_dim,pf%count), intent(out) x)

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

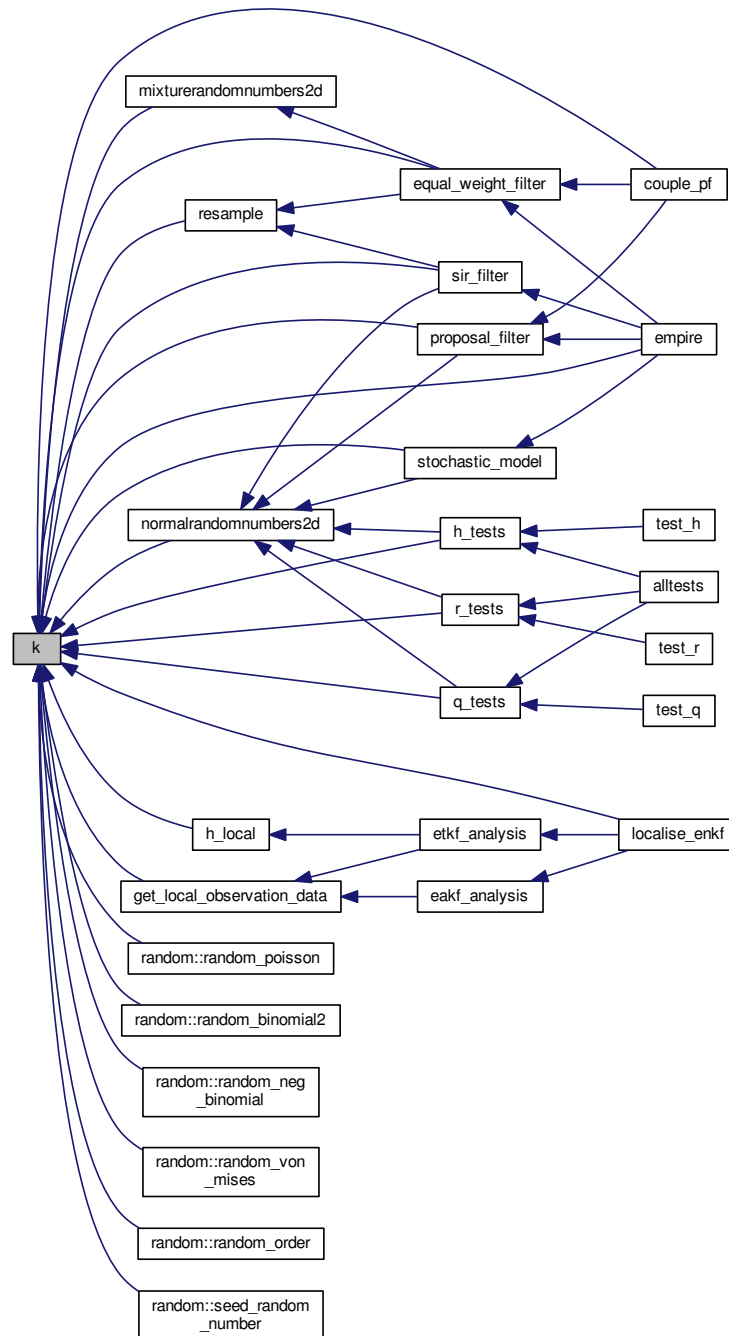
Parameters

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

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

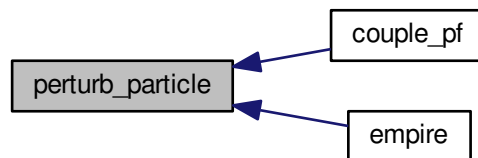
5.20.1.1 subroutine `perturb_particle` (real(kind=rk), dimension(state_dim), intent(inout) x)

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.1.2 subroutine `update_state` (real(kind=rk), dimension(state_dim), intent(out) state, real(kind=rk), dimension(state_dim), intent(in) fps, real(kind=rk), dimension(state_dim), intent(in) kgain, real(kind=rk), dimension(state_dim), intent(inout) betan)

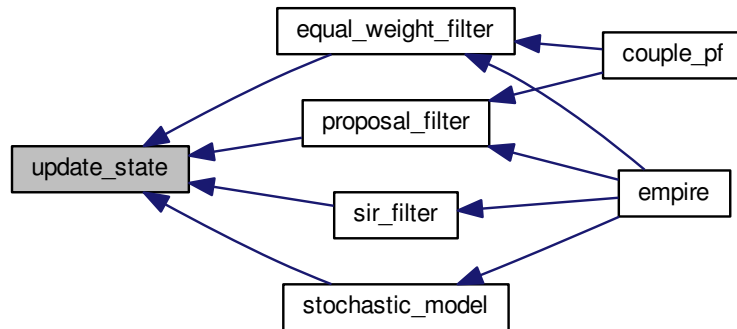
Subroutine to update the state.

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

Parameters

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

Here is the caller graph for this function:



5.21 src/operations/resample.f90 File Reference

Functions/Subroutines

- subroutine [resample](#)

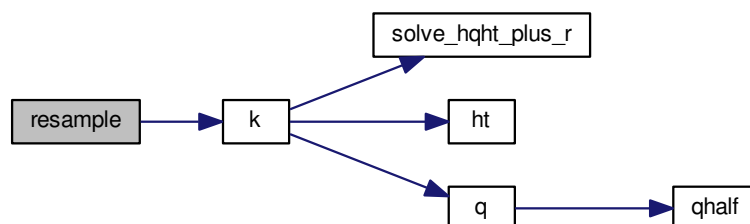
Subroutine to perform Universal Importance Resampling.

5.21.1 Function/Subroutine Documentation

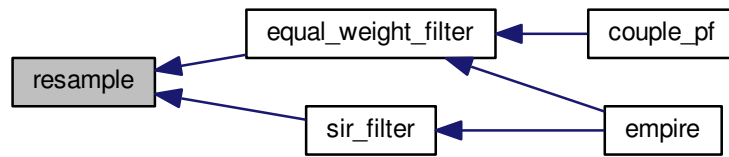
5.21.1.1 subroutine resample ()

Subroutine to perform Universal Importance Resampling.

Here is the call graph for this function:



Here is the caller graph for this function:



5.22 src/tests/alltests.f90 File Reference

Functions/Subroutines

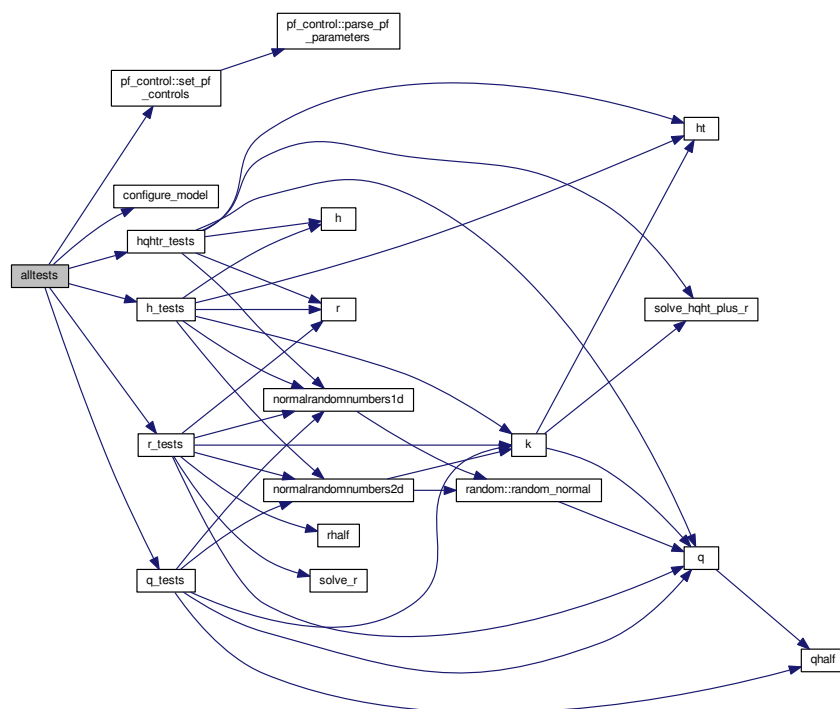
- program [alltests](#)
program to run all tests of user specific functions

5.22.1 Function/Subroutine Documentation

5.22.1.1 program alltests ()

program to run all tests of user specific functions

Here is the call graph for this function:



5.23 src/tests/test_h.f90 File Reference

Functions/Subroutines

- program [test_h](#)

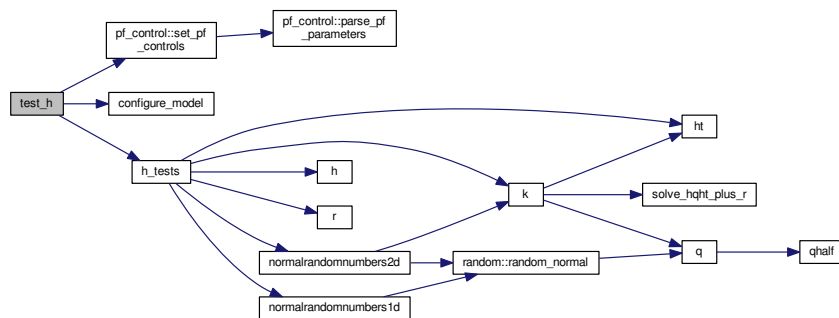
program to run tests of user supplied observation operator

5.23.1 Function/Subroutine Documentation

5.23.1.1 program test_h ()

program to run tests of user supplied observation operator

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

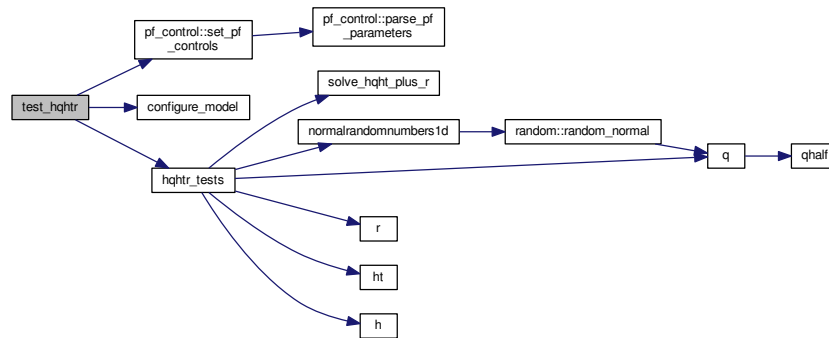
5.24.1 Function/Subroutine Documentation

5.24.1.1 program test_hqhtr ()

program to run tests of user supplied linear solve

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

Here is the call graph for this function:



5.25 src/tests/test_q.f90 File Reference

Functions/Subroutines

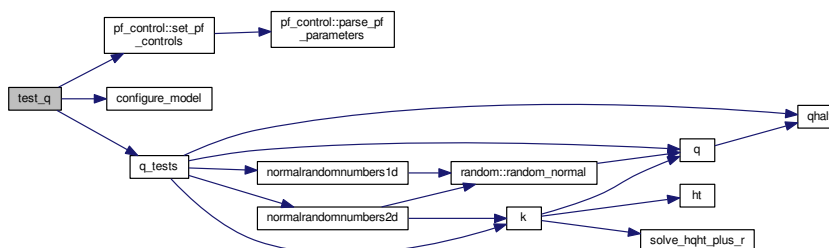
- program [test_q](#)
program to run tests of user supplied model error covariance matrix

5.25.1 Function/Subroutine Documentation

5.25.1.1 program test_q ()

program to run tests of user supplied model error covariance matrix

Here is the call graph for this function:



5.26 src/tests/test_r.f90 File Reference

Functions/Subroutines

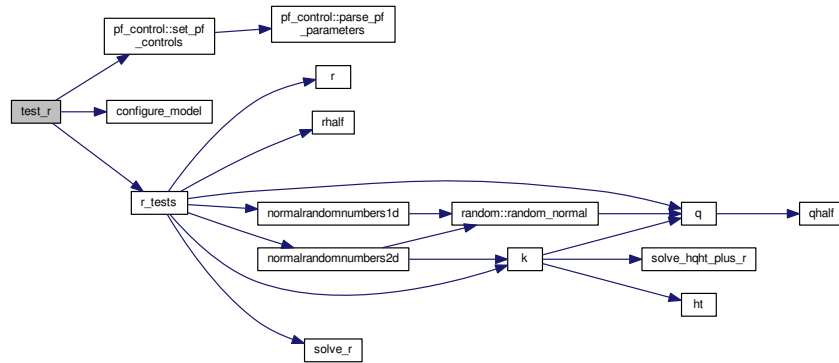
- program [test_r](#)
program to run all tests of user supplied observation error covariance matrix/

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/

Here is the call graph for this function:



5.27 src/tests/tests.f90 File Reference

Functions/Subroutines

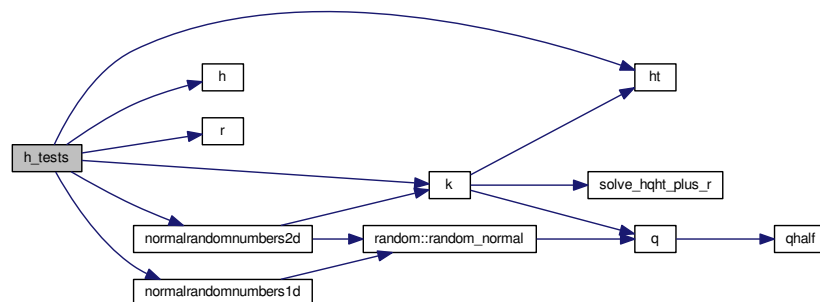
- subroutine [h_tests](#) ()
- subroutine [r_tests](#) ()
- subroutine [q_tests](#) ()
- subroutine [hqhtr_tests](#) ()

5.27.1 Function/Subroutine Documentation

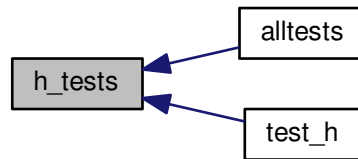
5.27.1.1 subroutine h_tests ()

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

Here is the call graph for this function:



Here is the caller graph for this function:

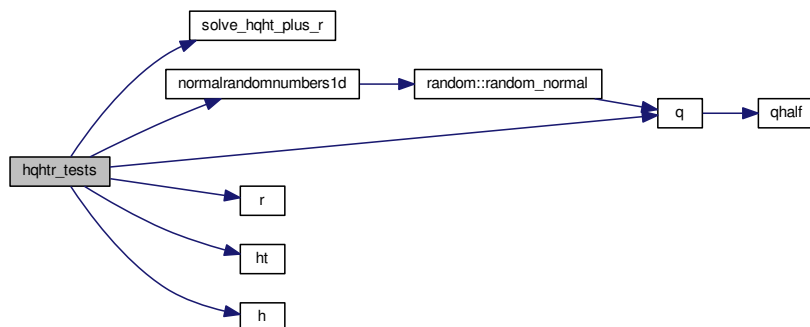


5.27.1.2 subroutine `hqhtr_tests ()`

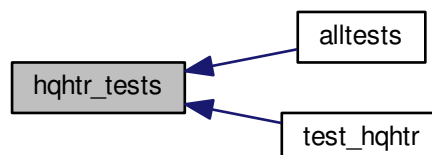
These are some tests to check that the linear solve operator is implemented correctly

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

Here is the call graph for this function:



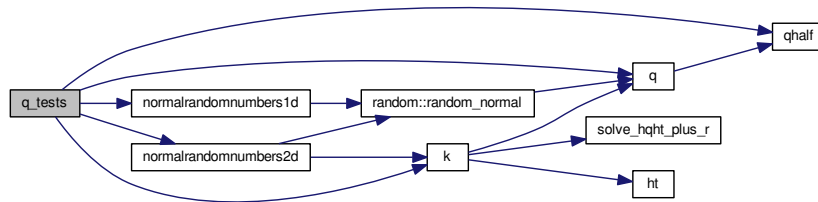
Here is the caller graph for this function:



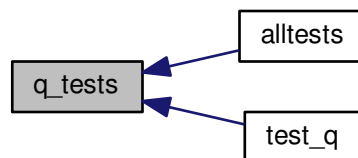
5.27.1.3 subroutine q_tests ()

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

Here is the call graph for this function:



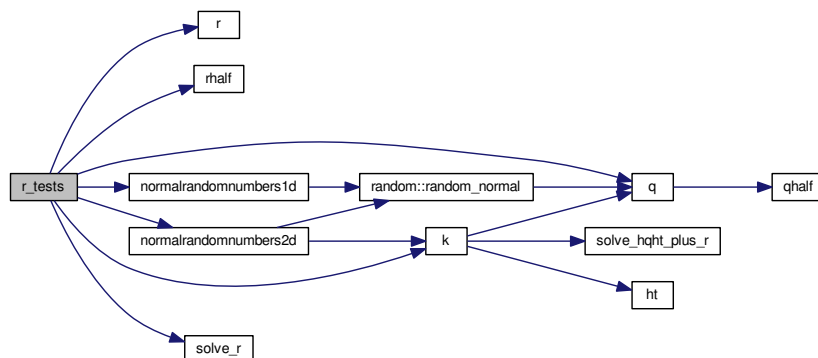
Here is the caller graph for this function:



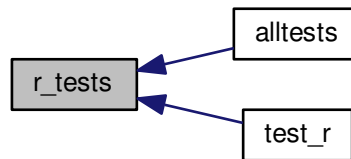
5.27.1.4 subroutine r_tests ()

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.28 src/utils/comms.f90 File Reference

Data Types

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

5.29 src/utils/data_io.f90 File Reference

Functions/Subroutines

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

5.29.1 Function/Subroutine Documentation

5.29.1.1 subroutine [get_observation_data](#) (real(kind=rk), dimension(obs_dim), intent(out) y)

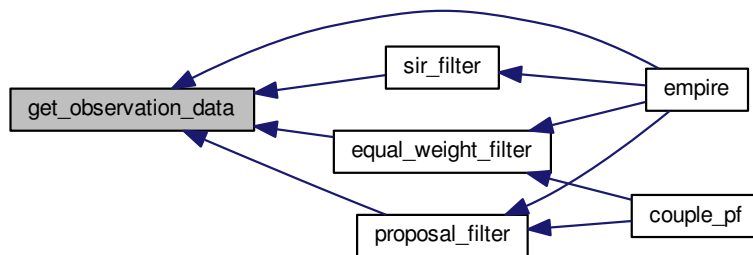
Subroutine to read observation from a file

Uses pftimestep to determine which observation to read.

Parameters

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

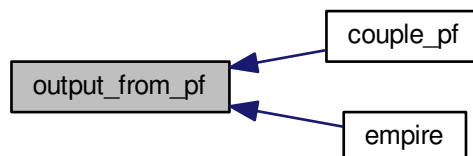
Here is the caller graph for this function:



5.29.1.2 subroutine output_from_pf ()

subroutine to ouput data from the filter

Here is the caller graph for this function:



5.29.1.3 subroutine save_observation_data (real(kind=rk), dimension(obs_dim), intent(in) y)

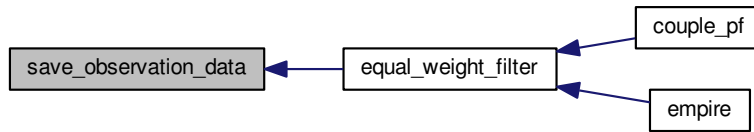
Subroutine to save observation to a file

Uses `pftimestep` to determine which observation to save.

Parameters

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

Here is the caller graph for this function:



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

Subroutine to save truth to a file

.

Parameters

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

Here is the caller graph for this function:



5.30 src/utils/diagnostics.f90 File Reference

Functions/Subroutines

- subroutine [diagnostics](#)

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

- subroutine [trajectories](#)

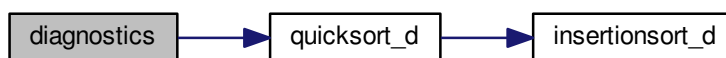
subroutine to output trajectories

5.30.1 Function/Subroutine Documentation

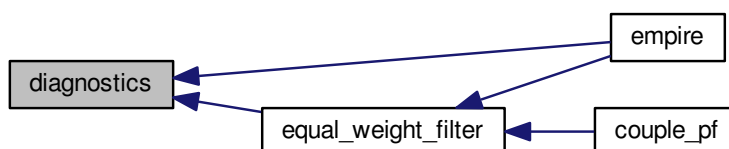
5.30.1.1 subroutine `diagnostics` ()

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

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

Here is the caller graph for this function:



5.31 src/utls/genQ.f90 File Reference

Functions/Subroutines

- subroutine [genq](#)

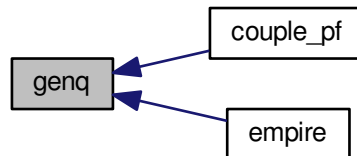
Subroutine to estimate Q from a long model run.

5.31.1 Function/Subroutine Documentation

5.31.1.1 subroutine genq ()

Subroutine to estimate Q from a long model run.

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

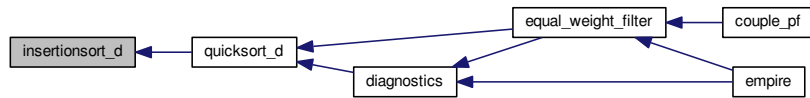
5.33.1.1 subroutine insertionsort_d (real(kind=kind(1.0d0)), dimension(na), intent(inout) A, integer, intent(in) nA)

subroutine to sort using the insertionsort algorithm

Parameters

in, out	a	array of doubles to be sorted
in	na	dimension of array a

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

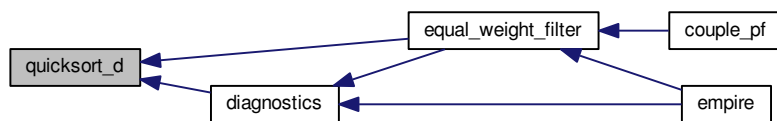
Parameters

in, out	a	array of doubles to be sorted
in	na	dimension of array a

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, [9](#)
- allocate_pf
 - pf_control, [13](#)
- alltests
 - alltests.f90, [70](#)
- alltests.f90
 - alltests, [70](#)
- bin_prob
 - random, [22](#)
- bprime
 - operator_wrappers.f90, [64](#)
- check_scaling
 - stochastic_model.f90, [58](#)
- comms, [9](#)
 - allocate_data, [9](#)
 - cpl_mpi_comm, [10](#)
 - deallocate_data, [9](#)
 - gblcount, [10](#)
 - gbldisp, [10](#)
 - initialise_mpi, [10](#)
 - mype_id, [10](#)
 - myrank, [10](#)
 - npfs, [10](#)
 - nproc, [10](#)
 - pf_mpi_comm, [10](#)
 - pfrank, [10](#)
- configure_model
 - model_specific.f90, [35](#)
- count
 - pf_control::pf_control_type, [18](#)
- couple_pf
 - old_pf_couple.f90, [47](#)
- couple_root
 - pf_control::pf_control_type, [18](#)
- cpl_mpi_comm
 - comms, [10](#)
- data_io.f90
 - get_observation_data, [76](#)
 - output_from_pf, [77](#)
 - save_observation_data, [77](#)
 - save_truth, [78](#)
- deallocate_data
 - comms, [9](#)
- deallocate_pf
 - pf_control, [14](#)
- diagnostics
 - diagnostics.f90, [78](#)
- diagnostics.f90
 - diagnostics, [78](#)
 - trajectories, [79](#)
- dist_st_ob
 - model_specific.f90, [36](#)
- dp
 - random, [31](#)
- eakf_analysis
 - eakf_analysis.f90, [50](#)
- eakf_analysis.f90
 - eakf_analysis, [50](#)
- efac
 - pf_control::pf_control_type, [18](#)
- empire
 - pf_couple.f90, [48](#)
- enkf_specific.f90
 - get_local_observation_data, [51](#)
 - h_local, [51](#)
 - localise_enkf, [52](#)
 - solve_rhalf_local, [52](#)
- equal_weight_filter
 - equivalent_weights_step.f90, [53](#)
- equivalent_weights_step.f90
 - equal_weight_filter, [53](#)
- etkf_analysis
 - etkf_analysis.f90, [54](#)
- etkf_analysis.f90
 - etkf_analysis, [54](#)
- gblcount
 - comms, [10](#)
- gbldisp
 - comms, [10](#)
- gen_Q
 - pf_parameters.dat, [49](#)
- gen_data
 - pf_control::pf_control_type, [18](#)
 - pf_parameters.dat, [49](#)
- gen_q
 - pf_control::pf_control_type, [18](#)
- gen_rand.f90
 - mixture_randomnumbers1d, [59](#)
 - mixture_randomnumbers2d, [60](#)
 - normal_randomnumbers1d, [61](#)
 - normal_randomnumbers2d, [61](#)
 - random_seed_mpi, [62](#)
 - uniform_randomnumbers1d, [63](#)
- genQ.f90

- genq, 79
- genq
 - genQ.f90, 79
- get_local_observation_data
 - enkf_specific.f90, 51
- get_observation_data
 - data_io.f90, 76
- h
 - model_specific.f90, 36
- h_local
 - enkf_specific.f90, 51
- h_tests
 - tests.f90, 73
- histogram_data, 11
 - kill_histogram_data, 11
 - load_histogram_data, 11
 - rank_hist_list, 11
 - rank_hist_nums, 11
 - rhl_n, 11
 - rhn_n, 11
- hqht_plus_r, 12
 - hqhtr_factor, 12
 - kill_hqhtr, 12
 - load_hqhtr, 12
- hqhtr_factor
 - hqht_plus_r, 12
- hqhtr_tests
 - tests.f90, 74
- ht
 - model_specific.f90, 37
- human_readable
 - pf_control::pf_control_type, 18
 - pf_parameters.dat, 49
- init
 - pf_control::pf_control_type, 18
- initialise_mpi
 - comms, 10
- innerhqht_plus_r_1
 - operator_wrappers.f90, 64
- innerr_1
 - operator_wrappers.f90, 65
- insertionsort_d
 - quicksort.f90, 80
- k
 - operator_wrappers.f90, 66
- keep
 - pf_control::pf_control_type, 18
 - pf_parameters.dat, 49
- kill_histogram_data
 - histogram_data, 11
- kill_hqhtr
 - hqht_plus_r, 12
- killq
 - qdata, 21
- killr
 - rdata, 32
- len
 - pf_control::pf_control_type, 18
- letkf_analysis
 - letkf_analysis.f90, 55
- letkf_analysis.f90
 - letkf_analysis, 55
- lngamma
 - random, 23
- load_histogram_data
 - histogram_data, 11
- load_hqhtr
 - hqht_plus_r, 12
- loadq
 - qdata, 21
- loadr
 - rdata, 32
- localise_enkf
 - enkf_specific.f90, 52
- mean
 - pf_control::pf_control_type, 18
- mixture_randomnumbers1d
 - gen_rand.f90, 59
- mixture_randomnumbers2d
 - gen_rand.f90, 60
- model_specific.f90, 35
 - configure_model, 35
 - dist_st_ob, 36
 - h, 36
 - ht, 37
 - q, 39
 - qhalf, 41
 - r, 42
 - rhalf, 43
 - solve_hqht_plus_r, 44
 - solve_r, 45
 - solve_rhalf, 46
- mype_id
 - comms, 10
- myrank
 - comms, 10
- nens
 - pf_control::pf_control_type, 19
- nfac
 - pf_control::pf_control_type, 19
 - pf_parameters.dat, 49
- normal_randomnumbers1d
 - gen_rand.f90, 61
- normal_randomnumbers2d
 - gen_rand.f90, 61
- npfs
 - comms, 10
- nproc
 - comms, 10
- nudgefac
 - pf_control::pf_control_type, 19
 - pf_parameters.dat, 49

- obs_dim
 - sizes, 33
- old_pf_couple.f90
 - couple_pf, 47
- operator_wrappers.f90
 - bprime, 64
 - innerhght_plus_r_1, 64
 - innerr_1, 65
 - k, 66
- output_from_pf
 - data_io.f90, 77
- parse_pf_parameters
 - pf_control, 14
- particles
 - pf_control::pf_control_type, 19
- perturb_particle
 - perturb_particle.f90, 68
- perturb_particle.f90
 - perturb_particle, 68
 - update_state, 68
- pf
 - pf_control, 16
- pf_control, 12
 - allocate_pf, 13
 - deallocate_pf, 14
 - parse_pf_parameters, 14
 - pf, 16
 - set_pf_controls, 15
- pf_control::pf_control_type, 16
 - count, 18
 - couple_root, 18
 - efac, 18
 - gen_data, 18
 - gen_q, 18
 - human_readable, 18
 - init, 18
 - keep, 18
 - len, 18
 - mean, 18
 - nens, 19
 - nfac, 19
 - nudgefac, 19
 - particles, 19
 - psi, 19
 - qscale, 19
 - rho, 19
 - talagrand, 19
 - time, 19
 - time_bwn_obs, 19
 - time_obs, 19
 - timestep, 19
 - type, 20
 - ufac, 20
 - use_mean, 20
 - use_rmse, 20
 - use_talagrand, 20
 - use_traj, 20
 - use_var, 20
 - use_weak, 20
 - weight, 20
- pf_couple.f90
 - empire, 48
- pf_mpi_comm
 - comms, 10
- pf_parameters.dat
 - gen_Q, 49
 - gen_data, 49
 - human_readable, 49
 - keep, 49
 - nfac, 49
 - nudgefac, 49
 - Qscale, 49
 - time_bwn_obs, 49
 - time_obs, 49
 - type, 49
 - ufac, 49
 - use_mean, 49
 - use_rmse, 49
 - use_talagrand, 49
 - use_traj, 49
 - use_var, 49
 - use_weak, 49
- pfrank
 - comms, 10
- proposal_filter
 - proposal_filter.f90, 56
- proposal_filter.f90
 - proposal_filter, 56
- psi
 - pf_control::pf_control_type, 19
- q
 - model_specific.f90, 39
- q_tests
 - tests.f90, 74
- qcol
 - qdata, 21
- qdata, 20
 - killq, 21
 - loadq, 21
 - qcol, 21
 - qdiag, 21
 - qn, 21
 - qne, 21
 - qrow, 21
 - qscale, 21
 - qval, 21
- qdiag
 - qdata, 21
- qhalf
 - model_specific.f90, 41
- qn
 - qdata, 21
- qne
 - qdata, 21
- qrow
 - qdata, 21

- Qscale
 - pf_parameters.dat, [49](#)
- qscale
 - pf_control::pf_control_type, [19](#)
 - qdata, [21](#)
- quicksort.f90
 - insertionsort_d, [80](#)
 - quicksort_d, [81](#)
- quicksort_d
 - quicksort.f90, [81](#)
- qval
 - qdata, [21](#)
- r
 - model_specific.f90, [42](#)
- r_tests
 - tests.f90, [75](#)
- random, [22](#)
 - bin_prob, [22](#)
 - dp, [31](#)
 - lngamma, [23](#)
 - random_beta, [23](#)
 - random_binomial1, [24](#)
 - random_binomial2, [24](#)
 - random_cauchy, [24](#)
 - random_chisq, [24](#)
 - random_exponential, [25](#)
 - random_gamma, [25](#)
 - random_gamma1, [26](#)
 - random_gamma2, [26](#)
 - random_inv_gauss, [27](#)
 - random_mvnorm, [27](#)
 - random_neg_binomial, [27](#)
 - random_normal, [28](#)
 - random_order, [29](#)
 - random_poisson, [29](#)
 - random_t, [30](#)
 - random_von_mises, [30](#)
 - random_weibull, [30](#)
 - seed_random_number, [31](#)
- random_beta
 - random, [23](#)
- random_binomial1
 - random, [24](#)
- random_binomial2
 - random, [24](#)
- random_cauchy
 - random, [24](#)
- random_chisq
 - random, [24](#)
- random_exponential
 - random, [25](#)
- random_gamma
 - random, [25](#)
- random_gamma1
 - random, [26](#)
- random_gamma2
 - random, [26](#)
- random_inv_gauss
 - random, [27](#)
- random_mvnorm
 - random, [27](#)
- random_neg_binomial
 - random, [27](#)
- random_normal
 - random, [28](#)
- random_order
 - random, [29](#)
- random_poisson
 - random, [29](#)
- random_seed_mpi
 - gen_rand.f90, [62](#)
- random_t
 - random, [30](#)
- random_von_mises
 - random, [30](#)
- random_weibull
 - random, [30](#)
- rank_hist_list
 - histogram_data, [11](#)
- rank_hist_nums
 - histogram_data, [11](#)
- rcol
 - rdata, [32](#)
- rdata, [31](#)
 - killr, [32](#)
 - loadr, [32](#)
 - rcol, [32](#)
 - rdiag, [32](#)
 - rn, [32](#)
 - rne, [32](#)
 - rrow, [32](#)
 - rval, [32](#)
- rdiag
 - rdata, [32](#)
- resample
 - resample.f90, [69](#)
- resample.f90
 - resample, [69](#)
- rhalf
 - model_specific.f90, [43](#)
- rhl_n
 - histogram_data, [11](#)
- rhn_n
 - histogram_data, [11](#)
- rho
 - pf_control::pf_control_type, [19](#)
- rn
 - rdata, [32](#)
- rne
 - rdata, [32](#)
- rrow
 - rdata, [32](#)
- rval
 - rdata, [32](#)
- save_observation_data
 - data_io.f90, [77](#)

- save_truth
 - data_io.f90, 78
- seed_random_number
 - random, 31
- set_pf_controls
 - pf_control, 15
- sir_filter
 - sir_filter.f90, 57
- sir_filter.f90
 - sir_filter, 57
- sizes, 32
 - obs_dim, 33
 - state_dim, 33
- solve_hqht_plus_r
 - model_specific.f90, 44
- solve_r
 - model_specific.f90, 45
- solve_rhalf
 - model_specific.f90, 46
- solve_rhalf_local
 - enkf_specific.f90, 52
- src/DOC_README.txt, 50
- src/controllers/old_pf_couple.f90, 47
- src/controllers/pf_control.f90, 47
- src/controllers/pf_couple.f90, 47
- src/controllers/pf_parameters.dat, 48
- src/controllers/sizes.f90, 50
- src/data/Qdata.f90, 50
- src/data/Rdata.f90, 50
- src/filters/eakf_analysis.f90, 50
- src/filters/enkf_specific.f90, 51
- src/filters/equivalent_weights_step.f90, 53
- src/filters/etkf_analysis.f90, 54
- src/filters/letkf_analysis.f90, 55
- src/filters/proposal_filter.f90, 56
- src/filters/sir_filter.f90, 57
- src/filters/stochastic_model.f90, 58
- src/operations/gen_rand.f90, 59
- src/operations/operator_wrappers.f90, 63
- src/operations/perturb_particle.f90, 67
- src/operations/resample.f90, 69
- src/tests/alltests.f90, 70
- src/tests/test_h.f90, 71
- src/tests/test_hqhtr.f90, 71
- src/tests/test_q.f90, 72
- src/tests/test_r.f90, 72
- src/tests/tests.f90, 73
- src/utis/comms.f90, 76
- src/utis/data_io.f90, 76
- src/utis/diagnostics.f90, 78
- src/utis/genQ.f90, 79
- src/utis/histogram.f90, 80
- src/utis/quicksort.f90, 80
- src/utis/random_d.f90, 81
- state_dim
 - sizes, 33
- stochastic_model
 - stochastic_model.f90, 58
- stochastic_model.f90
 - check_scaling, 58
 - stochastic_model, 58
- talagrand
 - pf_control::pf_control_type, 19
- test_h
 - test_h.f90, 71
- test_h.f90
 - test_h, 71
- test_hqhtr
 - test_hqhtr.f90, 71
- test_hqhtr.f90
 - test_hqhtr, 71
- test_q
 - test_q.f90, 72
- test_q.f90
 - test_q, 72
- test_r
 - test_r.f90, 73
- test_r.f90
 - test_r, 73
- tests.f90
 - h_tests, 73
 - hqhtr_tests, 74
 - q_tests, 74
 - r_tests, 75
- time
 - pf_control::pf_control_type, 19
- time_bwn_obs
 - pf_control::pf_control_type, 19
 - pf_parameters.dat, 49
- time_obs
 - pf_control::pf_control_type, 19
 - pf_parameters.dat, 49
- timestep
 - pf_control::pf_control_type, 19
- trajectories
 - diagnostics.f90, 79
- type
 - pf_control::pf_control_type, 20
 - pf_parameters.dat, 49
- ufac
 - pf_control::pf_control_type, 20
 - pf_parameters.dat, 49
- uniformrandomnumbers1d
 - gen_rand.f90, 63
- update_state
 - perturb_particle.f90, 68
- use_mean
 - pf_control::pf_control_type, 20
 - pf_parameters.dat, 49
- use_rmse
 - pf_control::pf_control_type, 20
 - pf_parameters.dat, 49
- use_talagrand
 - pf_control::pf_control_type, 20
 - pf_parameters.dat, 49

use_traj
 pf_control::pf_control_type, [20](#)
 pf_parameters.dat, [49](#)
use_var
 pf_control::pf_control_type, [20](#)
 pf_parameters.dat, [49](#)
use_weak
 pf_control::pf_control_type, [20](#)
 pf_parameters.dat, [49](#)
weight
 pf_control::pf_control_type, [20](#)