EMPIRE DA

0.1

Generated by Doxygen 1.8.8

Wed Jan 14 2015 18:25:30

Contents

Chapter 1

EMPIRE Data Assimilation Documentation

Author

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

Date

Time-stamp: <2015-01-14 18:25:27 pbrowne>

1.1 EMPIRE Methods

For a list of methods implemented in EMPIRE, please click here: methods

1.2 Downloading

The current version is an *academic version*, and the user interface may be subject to change.

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

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

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

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


1.3 Compiling

1.3.1 Compilation of the source code

The Makefile must be editted for the specific compiler setup. In the main directory you will find the file `Makefile`. Edit the variables as follows:

- `FC` The fortran compiler

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

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

To compile the source code, simply then type the command

```
1 make
```

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

- empire
- alltests
- test_hqhtr
- test_q
- test_r

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

```
1 make clean
```

1.3.2 Compilation of the documentation

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

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

```
1 make docs
```

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

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

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

```
1 make doc_html
```

1.4 Customising for specific models

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

The file model_specific.f90 should be editted 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.

- reconfigure_model This is called after each observation timestep. If the observation dimension changes it should be updated here, along with the number of model timesteps until the next observation
- h This is the observation operator
- ht This is the transpose of the observation operator
- r This is the observation error covariance matrix R
- rhalf This is the square root of the observation error covariance matrix $R^{\frac{1}{2}}$
- solve_r This is a linear solve with the observation error covariance matrix, i.e. given b, find x such that $Rx = b$ or indeed, $x = R^{-1}b$
- solve_rhalf This is a linear solve with the square root of the observation error covariance matrix, i.e. given b, find x such that $R^{\frac{1}{2}}x = b$ or indeed, $x = R^{-\frac{1}{2}}b$
- q This is the model error covariance matrix Q
- qhalf This is the square root model error covariance matrix $Q^{\frac{1}{2}}$
- solve_hqht_plus_r This is a linear solve with the matrix $(HQH^T + R)$
- dist_st_ob This specifies the distance between a an element of the state vector and an element of the observation vector

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

1.5 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 $R^{-1}Ry = y$, and if $Q^{\frac{1}{2}}Q^{\frac{1}{2}}x = Qx$ for various different vectors x, y.

1.6 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.reading.ac.uk/~darc/empire.

1.7 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.

1.8 Examples

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

1.9 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

Assimilation Methods

2.1 Filters

The filters implemented in EMPIRE can be divided into two categories, particle filters and Ensemble Kalman filters

2.1.1 Particle filters

2.1.1.1 SIR Filter (Sequential Importance Resampling)

See file sir_filter
Gordon, Salmond and Smith (1993).
Model specific operations required:

- qhalf
- h
- solve_r
  The SIR filter has no parameters to be chosen.
  To select the SIR filter, in pf_parameters.dat set the following variables:
  - type = 'SI'

2.1.1.2 Equivalent Weights Particle Filter

See files proposal_filter equivalent_weights_filter
Van Leeuwen (2010).
Model specific operations required:

- qhalf
- q
- h
- ht
- solve_r
- solve_hqht_plus_r

- rhalf
  The Equivalent Weights particle filter has a number of free parameters to be chosen.
  - nudgefac
  - nfac
  - ufac
  - keep
    To select the EWPF, in pf_parameters.dat set the following variables:
  - type = 'EW'

2.1.2 Ensemble Kalman filters

2.1.2.1 LETKF (The Localised Ensemble Transform Kalman Filter)

See file letkf_analysis
`Hunt, Kostelich and Szunyogh (2007).`
Model specific operations required:

- h
- solve_rhalf
- dist_st_ob
  The LETKF has a number of free parameters to be chosen.
  - rho
  - len
    To select the LETKF, in pf_parameters.dat set the following variables:
  - type = 'ET'

2.2 Smoothers

Coming at some point in the future: LETKS (Please contact us if you want us to develop this sooner rather than later)

2.3 Variational Methods

Coming at some point in the future: 4DEnVar (Please contact us if you want us to develop this sooner rather than later)

Chapter 3

Other EMPIRE features

3.1 Generating artificial observations

EMPIRE can generate artificial observations easily and quickly.

Model specific operations required:

- h
- rhalf
- qhalf

In pf_parameters.dat set the following variables:

- gen_data = .true.
- type = 'EW'

The system then should be run with a single ensemble member and a single EMPIRE process, i.e.

```
mpirun -np 1 model : -np 1 empire
```

3.2 Observations

To use real observations (i.e. those not generated automatically in twin experiment mode) the user must change the subroutine get_observation_data in data_io.f90.

When called, get_observation_data must return the vector of observations y that corresponds to the observation on, subsequently to, the current timestep which is stored in the variable pf%timestep within the module pf_control.

3.3 Running a deterministic ensemble

EMPIRE can simply integrate forward in time an ensemble of models.

In pf_parameters.dat set the following variables:

- type = 'DE'

  Todo ADD THIS

3.4 Running a stochastic ensemble

EMPIRE can integrate forward in time an ensemble of models whilst adding stochastic forcing.

Model specific operations required:

- qhalf

In pf_parameters.dat set the following variables:

- type = 'SE'

Chapter 4

Todo List

Page Other EMPIRE features

ADD THIS

Chapter 5

Data Type Index

5.1 Data Types List

Here are the data types with brief descriptions:

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Data Type Documentation

7.1 comms Module Reference

Module containing EMPIRE coupling data.

Public Member Functions

- subroutine allocate_data
- subroutine deallocate_data
- subroutine initialise_mpi

  subroutine to make EMPIRE connections and saves details into pf_control module

Public Attributes

- integer cpl_mpi_comm

  the communicator between the empire codes and the model master nodes
- integer mype_id

  the rank of this process on MPI_COMM_WORLD
- integer myrank

  the rank of this process on CPL_MPI_COMM
- integer nproc

  the total number of processes
- integer pf_mpi_comm

  the communicator between DA processes
- integer pfrank

  the rank of this process on PF_MPI_COMM
- integer npfs

  the total number of DA processes
- integer, dimension(:), allocatable gblcount

  the number of ensemble members associated with each DA process
- integer, dimension(:), allocatable gbldisp

  the displacements of each each ensemble member relative to pfrank=0. VERY useful for mpi_gatherv and mpi_↩ scatterv on pf_mpi_comm

7.1.1 Detailed Description

Module containing EMPIRE coupling data.

Definition at line 30 of file comms.f90.

7.1.2 Member Function/Subroutine Documentation

7.1.2.1 subroutine comms::allocate_data ( )

Definition at line 47 of file comms.f90.

7.1.2.2 subroutine comms::deallocate_data ( )

Definition at line 53 of file comms.f90.

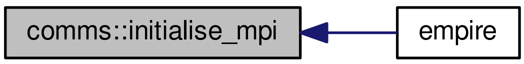
Here is the caller graph for this function:

7.1.2.3 subroutine comms::initialise_mpi ( )

subroutine to make EMPIRE connections and saves details into pf_control module

Definition at line 60 of file comms.f90.

Here is the caller graph for this function:



7.1.3 Member Data Documentation

7.1.3.1 integer comms::cpl_mpi_comm

the communicator between the empire codes and the model master nodes

Definition at line 31 of file comms.f90.

7.1.3.2 integer, dimension(:), allocatable comms::gblcount

the number of ensemble members associated with each DA process

Definition at line 39 of file comms.f90.

7.1.3.3 integer, dimension(:), allocatable comms::gbldisp

the displacements of each each ensemble member relative to pfrank=0. VERY useful for mpi_gatherv and mpi_↩
scatterv on pf_mpi_comm

Definition at line 41 of file comms.f90.

7.1.3.4 integer comms::mype_id

the rank of this process on MPI_COMM_WORLD

Definition at line 33 of file comms.f90.

7.1.3.5 integer comms::myrank

the rank of this process on CPL_MPI_COMM

Definition at line 34 of file comms.f90.

7.1.3.6 integer comms::npfs

the total number of DA processes

Definition at line 38 of file comms.f90.

7.1.3.7 integer comms::nproc

the total number of processes

Definition at line 35 of file comms.f90.

7.1.3.8 integer comms::pf_mpi_comm

the communicator between DA processes

Definition at line 36 of file comms.f90.

7.1.3.9 integer comms::pfrank

the rank of this process on PF_MPI_COMM

Definition at line 37 of file comms.f90.

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

- src/utils/comms.f90

7.2 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

7.2.1 Detailed Description

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

Definition at line 29 of file histogram.f90.

7.2.2 Member Function/Subroutine Documentation

7.2.2.1 subroutine histogram_data::kill_histogram_data ( )

subroutine to clean up arrays used in rank histograms

Definition at line 57 of file histogram.f90.

7.2.2.2 subroutine histogram_data::load_histogram_data ( )

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

Definition at line 37 of file histogram.f90.

7.2.3 Member Data Documentation

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

Definition at line 30 of file histogram.f90.

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

Definition at line 31 of file histogram.f90.

7.2.3.3 integer histogram_data::rhl_n

Definition at line 32 of file histogram.f90.

7.2.3.4 integer histogram_data::rhn_n

Definition at line 32 of file histogram.f90.

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

- src/utils/histogram.f90

7.3 hqht_plus_r Module Reference

Public Member Functions

- subroutine load_hqhtr
- subroutine hqhtr_factor
- subroutine kill_hqhtr

7.3.1 Detailed Description

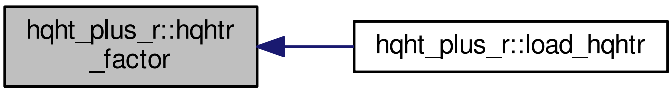
Definition at line 59 of file Rdata.f90.

7.3.2 Member Function/Subroutine Documentation

7.3.2.1 subroutine hqht_plus_r::hqhtr_factor ( )

Definition at line 69 of file Rdata.f90.

Here is the caller graph for this function:



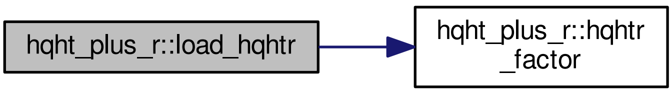
7.3.2.2 subroutine hqht_plus_r::kill_hqhtr ( )

Definition at line 74 of file Rdata.f90.

7.3.2.3 subroutine hqht_plus_r::load_hqhtr ( )

Definition at line 65 of file Rdata.f90.

Here is the call graph for this function:



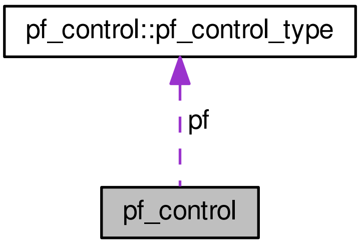
The documentation for this module was generated from the following file:

- src/data/Rdata.f90

7.4 pf_control Module Reference

module pf_control holds all the information to control the the main program

Collaboration diagram for pf_control:



Data Types

- type pf_control_type

Public Member Functions

- subroutine set_pf_controls

  subroutine to ensure pf_control data is ok
- subroutine parse_pf_parameters

  subroutine to read the namelist file and save it to pf datatype Here we read pf_parameters.dat
- subroutine allocate_pf

  subroutine to allocate space for the filtering code
- subroutine deallocate_pf

  subroutine to deallocate space for the filtering code

Public Attributes

- type(pf_control_type), save pf

  the derived data type holding all controlling data

7.4.1 Detailed Description

module pf_control holds all the information to control the the main program

Definition at line 29 of file pf_control.f90.

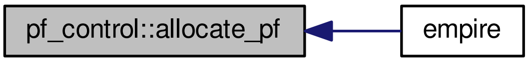
7.4.2 Member Function/Subroutine Documentation

7.4.2.1 subroutine pf_control::allocate_pf ( )

subroutine to allocate space for the filtering code

Definition at line 347 of file pf_control.f90.

Here is the caller graph for this function:



7.4.2.2 subroutine pf_control::deallocate_pf ( )

subroutine to deallocate space for the filtering code

Definition at line 369 of file pf_control.f90.

7.4.2.3 subroutine pf_control::parse_pf_parameters ( )

subroutine to read the namelist file and save it to pf datatype Here we read pf_parameters.dat

pf_parameters.dat is a fortran namelist file. As such, within it there must be a line beginning

&pf_params

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

This is just the fortran standard for namelists though.

On to the content...in any order, the pf_parameters.dat may contain the following things:

Integers:

- time_obs
- time_bwn_obs

Reals, double precision:

- nudgefac
- nfac
- ufac
- Qscale
- keep
- rho
- len

2 Characters:

- type

1 Character:

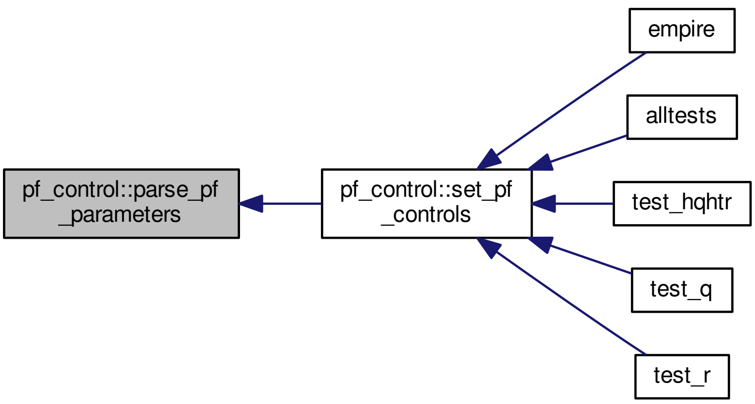
- init

Logicals:

- gen_Q
- gen_data
- use_talagrand
- use_weak
- use_var
- use_traj
- use_rmse
- human_readable

Definition at line 167 of file pf_control.f90.

Here is the caller graph for this function:

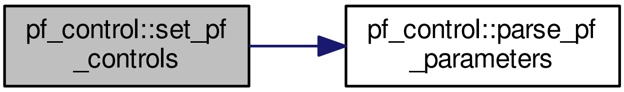


7.4.2.4 subroutine pf_control::set_pf_controls ( )

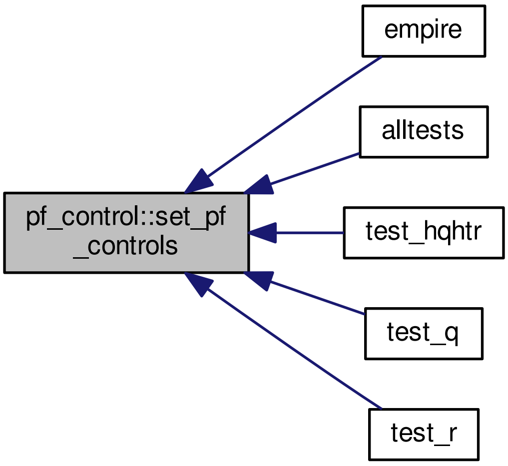
subroutine to ensure pf_control data is ok

Definition at line 100 of file pf_control.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3 Member Data Documentation

7.4.3.1 type(pf_control_type), save pf_control::pf

the derived data type holding all controlling data

Definition at line 95 of file pf_control.f90.

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

- src/controlers/pf_control.f90

7.5 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 The entries in the observation error covariance matrix R *are multiplied by the function* $\exp\left(\frac{\mathrm{dist}^2}{2\mathrm{len}^2}\right)$.
- integer couple_root

  empire master processor
- logical use_talagrand

  switch if true outputs rank histograms
- logical use_weak

  switch unused
- logical use_mean

  switch if true outputs ensemble mean
- logical use_var

  switch if true outputs ensemble variance
- logical use_traj

  switch if true outputs trajectories
- logical use_rmse

  switch if true outputs Root Mean Square Errors
- integer, dimension(:,:), allocatable talagrand

storage for rank histograms

- integer count

  number of ensemble members associated with this MPI process
- integer, dimension(:), allocatable particles

  particles associates with this MPI process
- character(2) type

  which filter to use currently this has a number of options:
- character(1) init

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

7.5.1 Detailed Description

Definition at line 31 of file pf_control.f90.

7.5.2 Member Data Documentation

7.5.2.1 integer pf_control::pf_control_type::count

number of ensemble members associated with this MPI process

Definition at line 68 of file pf_control.f90.

7.5.2.2 integer pf_control::pf_control_type::couple_root

empire master processor

Definition at line 60 of file pf_control.f90.

7.5.2.3 real(kind=kind(1.0d0)) pf_control::pf_control_type::efac

Definition at line 46 of file pf_control.f90.

7.5.2.4 logical pf_control::pf_control_type::gen_data

true generates synthetic obs for a twin experiment

Definition at line 37 of file pf_control.f90.

7.5.2.5 logical pf_control::pf_control_type::gen_q

true attempts to build up Q from long model run

Definition at line 38 of file pf_control.f90.

7.5.2.6 logical pf_control::pf_control_type::human_readable

unused

Definition at line 40 of file pf_control.f90.

7.5.2.7 character(1) pf_control::pf_control_type::init

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

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

Definition at line 78 of file pf_control.f90.

7.5.2.8 real(kind=kind(1.0d0)) pf_control::pf_control_type::keep

proportion of particles to keep in EWPF EW step

Definition at line 47 of file pf_control.f90.

7.5.2.9 real(kind=kind(1.0d0)) pf_control::pf_control_type::len

R localisation length scale The entries in the observation error covariance matrix R are multiplied by the function $\exp\left(\frac{\mathrm{dist}^2}{2\mathrm{len}^2}\right)$.

Definition at line 54 of file pf_control.f90.

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

mean state vector

Definition at line 43 of file pf_control.f90.

7.5.2.11 integer pf_control::pf_control_type::nens

the total number of ensemble members

Definition at line 32 of file pf_control.f90.

7.5.2.12 real(kind=kind(1.0d0)) pf_control::pf_control_type::nfac

standard deviation of normal distribution in mixture density

Definition at line 44 of file pf_control.f90.

7.5.2.13 real(kind=kind(1.0d0)) pf_control::pf_control_type::nudgefac

the nudging factor

Definition at line 36 of file pf_control.f90.

7.5.2.14 integer, dimension(:), allocatable pf_control::pf_control_type::particles

particles associates with this MPI process

Definition at line 69 of file pf_control.f90.

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

state vector of ensemble members on this mpi process

Definition at line 42 of file pf_control.f90.

7.5.2.16 real(kind=kind(1.0d0)) pf_control::pf_control_type::qscale

scalar to multiply Q by

Definition at line 49 of file pf_control.f90.

7.5.2.17 real(kind=kind(1.0d0)) pf_control::pf_control_type::rho

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

Definition at line 51 of file pf_control.f90.

7.5.2.18 integer, dimension(:,:), allocatable pf_control::pf_control_type::talagrand

storage for rank histograms

Definition at line 67 of file pf_control.f90.

7.5.2.19 real(kind=kind(1.0d0)) pf_control::pf_control_type::time

dunno

Definition at line 48 of file pf_control.f90.

7.5.2.20 integer pf_control::pf_control_type::time_bwn_obs

the number of model timesteps between observations

Definition at line 35 of file pf_control.f90.

7.5.2.21 integer pf_control::pf_control_type::time_obs

the number of observations we will assimilate

Definition at line 34 of file pf_control.f90.

7.5.2.22 integer pf_control::pf_control_type::timestep =0

the current timestep as the model progresses

Definition at line 41 of file pf_control.f90.

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

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

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

- EW – the Equivalent Weights particle filter

Definition at line 70 of file pf_control.f90.

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

half width of the uniform distribution in mixture density

Definition at line 45 of file pf_control.f90.

7.5.2.25 logical pf_control::pf_control_type::use_mean

switch if true outputs ensemble mean

Definition at line 63 of file pf_control.f90.

7.5.2.26 logical pf_control::pf_control_type::use_rmse

switch if true outputs Root Mean Square Errors

Definition at line 66 of file pf_control.f90.

7.5.2.27 logical pf_control::pf_control_type::use_talagrand

switch if true outputs rank histograms

Definition at line 61 of file pf_control.f90.

7.5.2.28 logical pf_control::pf_control_type::use_traj

switch if true outputs trajectories

Definition at line 65 of file pf_control.f90.

7.5.2.29 logical pf_control::pf_control_type::use_var

switch if true outputs ensemble variance

Definition at line 64 of file pf_control.f90.

7.5.2.30 logical pf_control::pf_control_type::use_weak

switch unused

Definition at line 62 of file pf_control.f90.

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

the negative log of the weights of the particles

Definition at line 33 of file pf_control.f90.

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

- src/controlers/pf_control.f90

7.6 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

7.6.1 Detailed Description

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

- the model error covariance matrix

Definition at line 30 of file Qdata.f90.

7.6.2 Member Function/Subroutine Documentation

7.6.2.1 subroutine qdata::killq ( )

SUbroutine to deallocate user data for Q

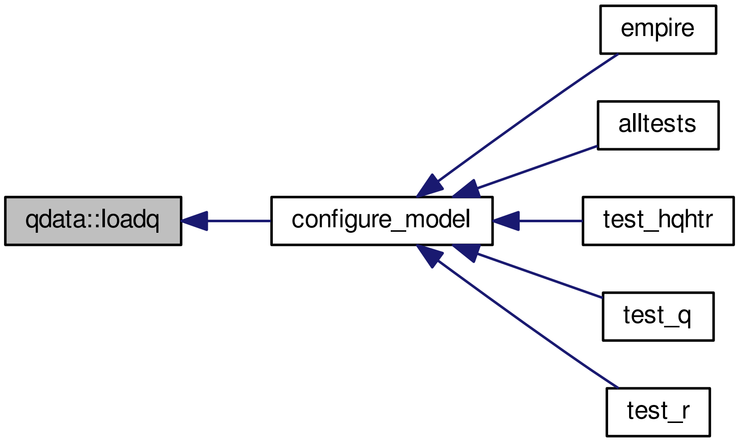
Definition at line 44 of file Qdata.f90.

7.6.2.2 subroutine qdata::loadq ( )

Subroutine to load in user data for Q.

Definition at line 38 of file Qdata.f90.

Here is the caller graph for this function:



7.6.3 Member Data Documentation

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

Definition at line 33 of file Qdata.f90.

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

Definition at line 34 of file Qdata.f90.

7.6.3.3 integer qdata::qn

Definition at line 32 of file Qdata.f90.

7.6.3.4 integer qdata::qne

Definition at line 32 of file Qdata.f90.

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

Definition at line 33 of file Qdata.f90.

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

Definition at line 35 of file Qdata.f90.

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

Definition at line 34 of file Qdata.f90.

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

- src/data/Qdata.f90

7.7 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)

7.7.1 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

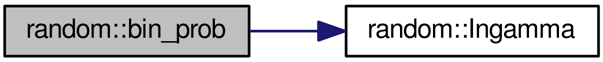
Definition at line 22 of file random_d.f90.

7.7.2 Member Function/Subroutine Documentation

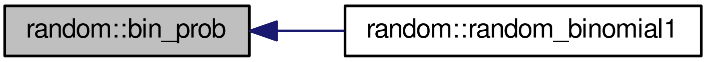
7.7.2.1 real(kind=kind(1.0d+0)) function random::bin_prob (integer, intent(in) *n*, real(kind=kind(1.0d+0)), intent(in) *p*, integer, intent(in) *r*)

Definition at line 1000 of file random_d.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.2 real (dp) function random::lngamma (real (dp), intent(in) *x*)

Definition at line 1018 of file random_d.f90.

Here is the caller graph for this function:

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

Definition at line 371 of file random_d.f90.

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

Definition at line 923 of file random_d.f90.

Here is the call graph for this function:

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

Definition at line 1082 of file random_d.f90.

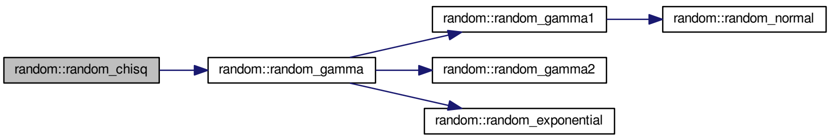
7.7.2.6 real(kind=kind(1.0d+0)) function random::random_cauchy ( )

Definition at line 1517 of file random_d.f90.

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

Definition at line 308 of file random_d.f90.

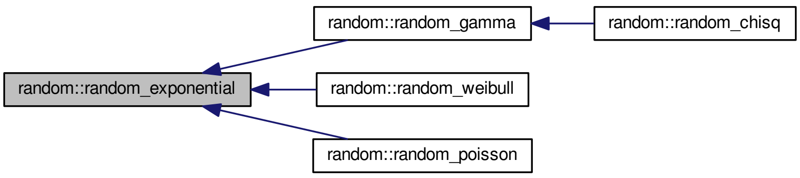
Here is the call graph for this function:



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

Definition at line 324 of file random_d.f90.

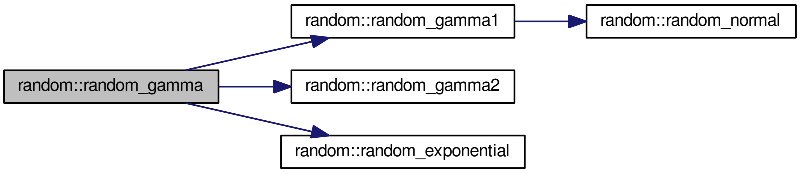
Here is the caller graph for this function:



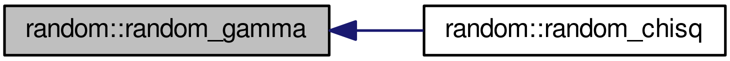
7.7.2.9 real(kind=kind(1.0d+0)) function random::random_gamma (real(kind=kind(1.0d+0)), intent(in) *s*, logical, intent(in) *first*)

Definition at line 154 of file random_d.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

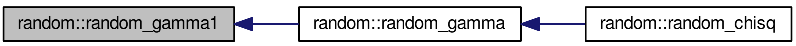


7.7.2.10 real(kind=kind(1.0d+0)) function random::random_gamma1 (real(kind=kind(1.0d+0)), intent(in) *s*, logical, intent(in) *first*)

Definition at line 189 of file random_d.f90.

Here is the call graph for this function:

Here is the caller graph for this function:



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

Definition at line 238 of file random_d.f90.

Here is the caller graph for this function:

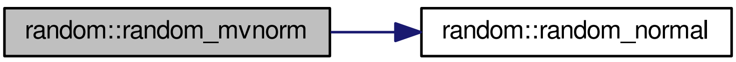
7.7.2.12 real(kind=kind(1.0d+0)) function random::random_inv_gauss (real(kind=kind(1.0d+0)), intent(in) *h*, real(kind=kind(1.0d+0)), intent(in) *b*, logical, intent(in) *first*)

Definition at line 610 of file random_d.f90.

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

Definition at line 509 of file random_d.f90.

Here is the call graph for this function:



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

Definition at line 1314 of file random_d.f90.

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

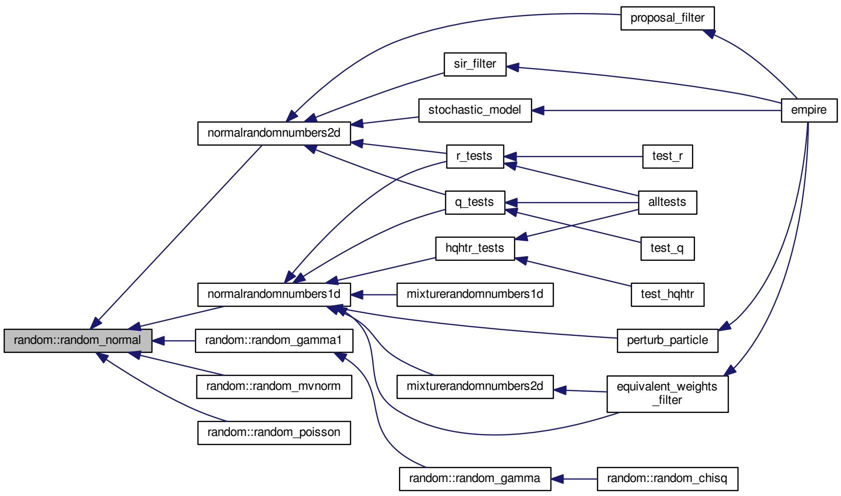
function to get random normal with zero mean and stdev 1

Returns

fn_val

Definition at line 108 of file random_d.f90.

Here is the caller graph for this function:



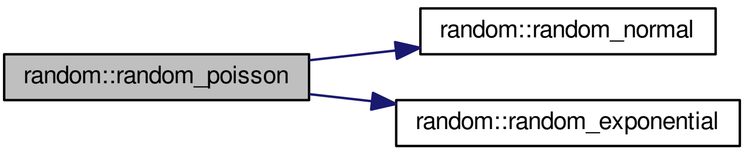
7.7.2.16 subroutine random::random_order (integer, dimension(n), intent(out) *order,* integer, intent(in) *n*)

Definition at line 1539 of file random_d.f90.

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

Definition at line 681 of file random_d.f90.

Here is the call graph for this function:



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

Definition at line 448 of file random_d.f90.

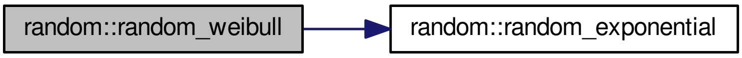
7.7.2.19 real(kind=kind(1.0d+0)) function random::random_von_mises (real(kind=kind(1.0d+0)), intent(in) *k,* logical, intent(in) *first*)

Definition at line 1389 of file random_d.f90.

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

Definition at line 351 of file random_d.f90.

Here is the call graph for this function:



7.7.2.21 subroutine random::seed_random_number (integer, intent(in) *iounit*)

Definition at line 1573 of file random_d.f90.

7.7.3 Member Data Documentation

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

Definition at line 101 of file random_d.f90.

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

- src/utils/random_d.f90

7.8 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

7.8.1 Detailed Description

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

Definition at line 29 of file Rdata.f90.

7.8.2 Member Function/Subroutine Documentation

7.8.2.1 subroutine rdata::killr ( )

SUbroutine to deallocate R data

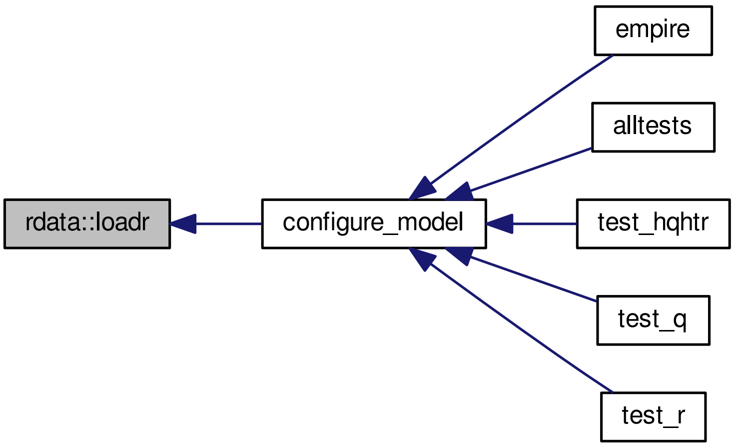
Definition at line 49 of file Rdata.f90.

7.8.2.2 subroutine rdata::loadr ( )

Subroutine to load data for R.

Definition at line 36 of file Rdata.f90.

Here is the caller graph for this function:



7.8.3 Member Data Documentation

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

Definition at line 32 of file Rdata.f90.

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

Definition at line 33 of file Rdata.f90.

7.8.3.3 integer rdata::rn

Definition at line 31 of file Rdata.f90.

7.8.3.4 integer rdata::rne

Definition at line 31 of file Rdata.f90.

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

Definition at line 32 of file Rdata.f90.

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

Definition at line 33 of file Rdata.f90.

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

- src/data/Rdata.f90

7.9 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

7.9.1 Detailed Description

Module that stores the dimension of observation and state spaces.

Definition at line 29 of file sizes.f90.

7.9.2 Member Data Documentation

7.9.2.1 integer sizes::obs_dim

size of the observation space

Definition at line 31 of file sizes.f90.

7.9.2.2 integer sizes::state_dim

dimension of the model

Definition at line 32 of file sizes.f90.

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

- src/controlers/sizes.f90

Chapter 8

File Documentation

8.1 model_specific.f90 File Reference

Functions/Subroutines

- subroutine configure_model

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

  subroutine to reset variables that may change when the observation network changes
- subroutine solve_r (obsDim, nrhs, y, v, t)

  subroutine to take an observation vector y and return v in observation space.
- subroutine solve_rhalf (obsdim, nrhs, y, v, t)

  subroutine to take an observation vector y and return v in observation space.
- subroutine solve_hqht_plus_r (obsdim, y, v, t)

  subroutine to take an observation vector y and return v in observation space.
- subroutine q (nrhs, x, Qx)

  subroutine to take a full state vector x and return Qx in state space.
- subroutine qhalf (nrhs, x, Qx)

  subroutine to take a full state vector x and return $Q^{1/2}x$ in state space.
- subroutine r (obsDim, nrhs, y, Ry, t)

  subroutine to take an observation vector x and return Rx in observation space.
- subroutine rhalf (obsDim, nrhs, y, Ry, t)

  subroutine to take an observation vector x and return Rx in observation space.
- subroutine h (obsDim, nrhs, x, hx, t)

  subroutine to take a full state vector x and return H(x) in observation space.
- subroutine ht (obsDim, nrhs, y, x, t)

  subroutine to take an observation vector y and return $x = H^T(y)$ in full state space.
- subroutine dist_st_ob (xp, yp, dis, t)

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

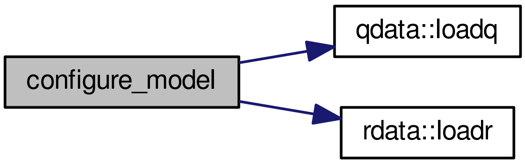
8.1.1 Function/Subroutine Documentation

8.1.1.1 subroutine configure_model ( )

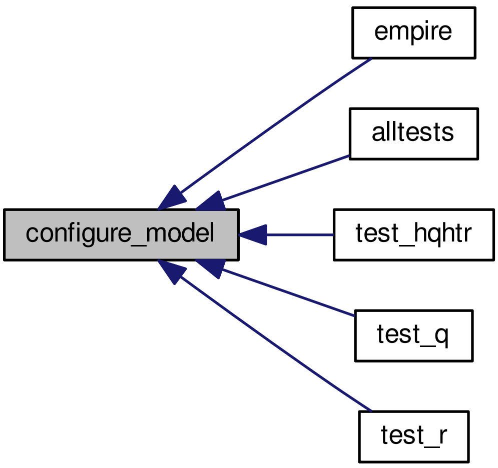
subroutine called initially to set up details and data for model specific functions

Definition at line 30 of file model_specific.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.1.2 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`	*xp*	the index in the state vector
`in`	*yp*	the index in the observation vector
`out`	*dis*	the distance between x(xp) and y(yp)

in	*t*	the current time index for observations

Definition at line 292 of file model_specific.f90.

Here is the caller graph for this function:

8.1.1.3 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.

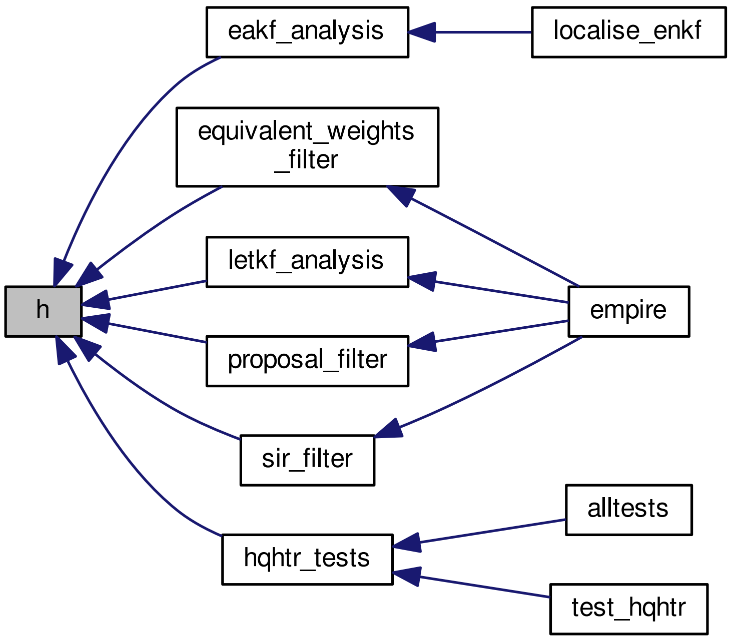
Given x compute Hx

Parameters

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

Definition at line 246 of file model_specific.f90.

Here is the caller graph for this function:



8.1.1.4 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.

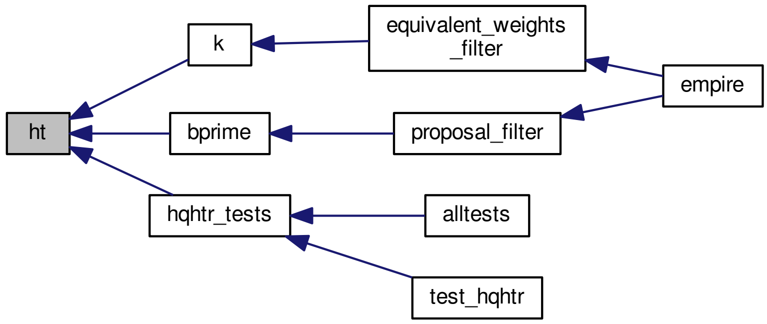
Given y compute $x = H^T(y)$

Parameters

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

Definition at line 269 of file model_specific.f90.

Here is the caller graph for this function:



8.1.1.5 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.

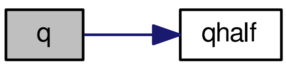
Given x compute Qx

Parameters

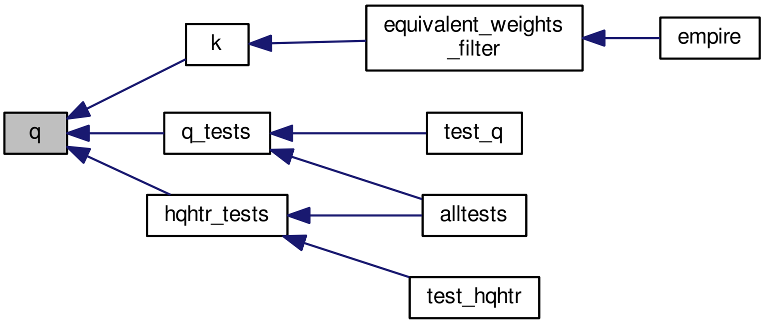
in	*nrhs*	the number of right hand sides
in	*x*	the input vector
out	*qx*	the resulting vector where Qx $= Qx$

Definition at line 156 of file model_specific.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.1.1.6 subroutine qhalf (integer, intent(in) *nrhs*, real(kind=rk), dimension(state_dim,nrhs), intent(in) *x*, real(kind=rk), dimension(state_dim,nrhs), intent(out) *Qx*)

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

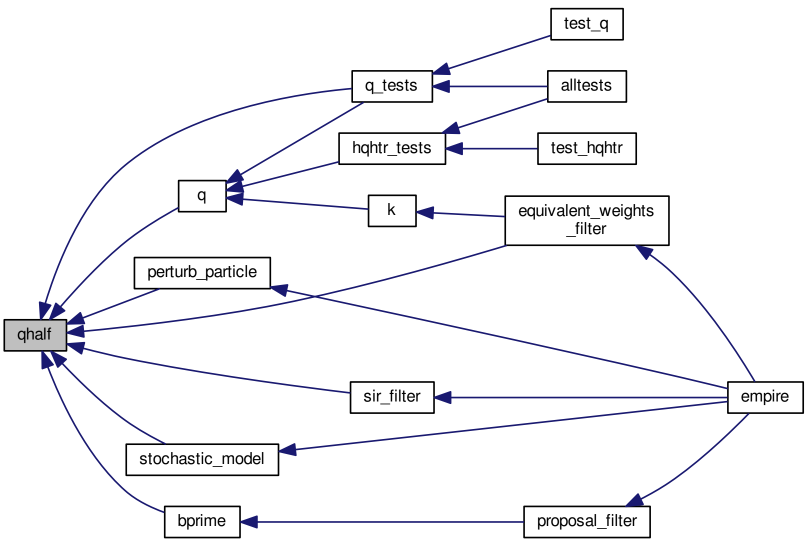
Given x compute $Q^{\frac{1}{2}}x$

Parameters

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

Definition at line 181 of file model_specific.f90.

Here is the caller graph for this function:



8.1.1.7 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.

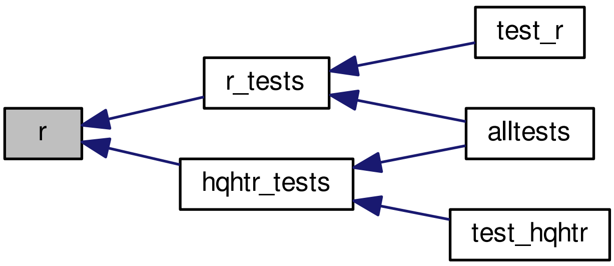
Given y compute Ry

Parameters

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

Definition at line 201 of file model_specific.f90.

Here is the caller graph for this function:

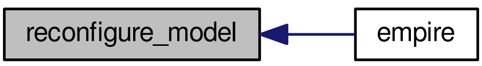


8.1.1.8 subroutine reconfigure_model ( )

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

Definition at line 70 of file model_specific.f90.

Here is the caller graph for this function:



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

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

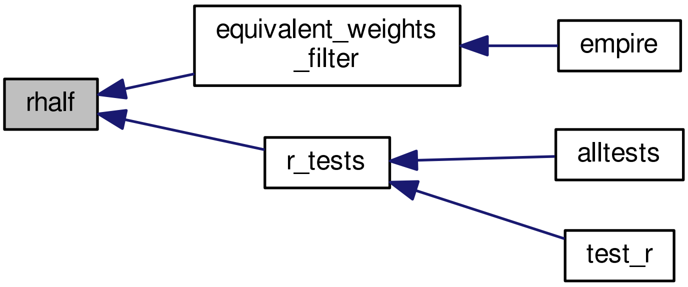
Given y compute $R^{\frac{1}{2}}y$

Parameters

in	*obsdim*	the dimension of the observations
in	*nrhs*	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

Definition at line 223 of file model_specific.f90.

Here is the caller graph for this function:



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

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

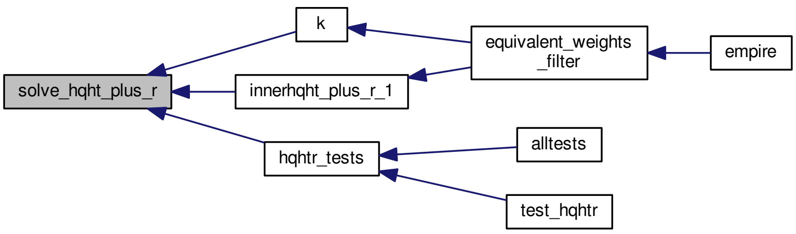
Given y find v such that $(HQH^T + R)v = y$

Parameters

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

Definition at line 136 of file model_specific.f90.

Here is the caller graph for this function:



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

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

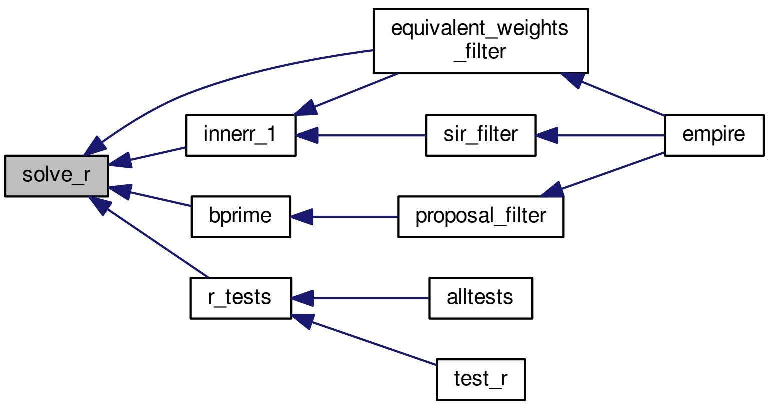
Given y find v such that $Rv = y$

Parameters

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

Definition at line 94 of file model_specific.f90.

Here is the caller graph for this function:



8.1.1.12 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.

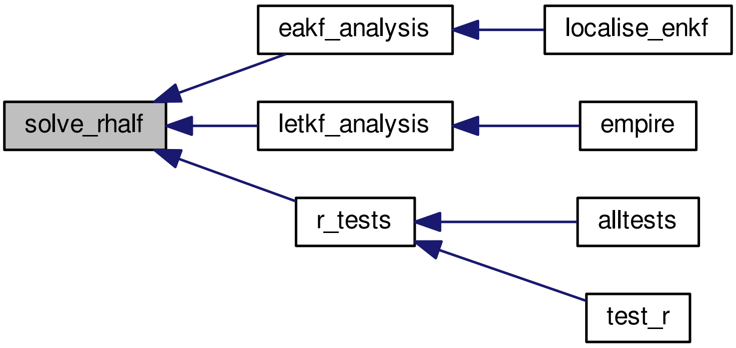
Given y find v such that $R^{\frac{1}{2}}v = y$

Parameters

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

Definition at line 114 of file model_specific.f90.

Here is the caller graph for this function:



8.2 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

8.3 src/controlers/pf_couple.f90 File Reference

Functions/Subroutines

- program empire

    the main program

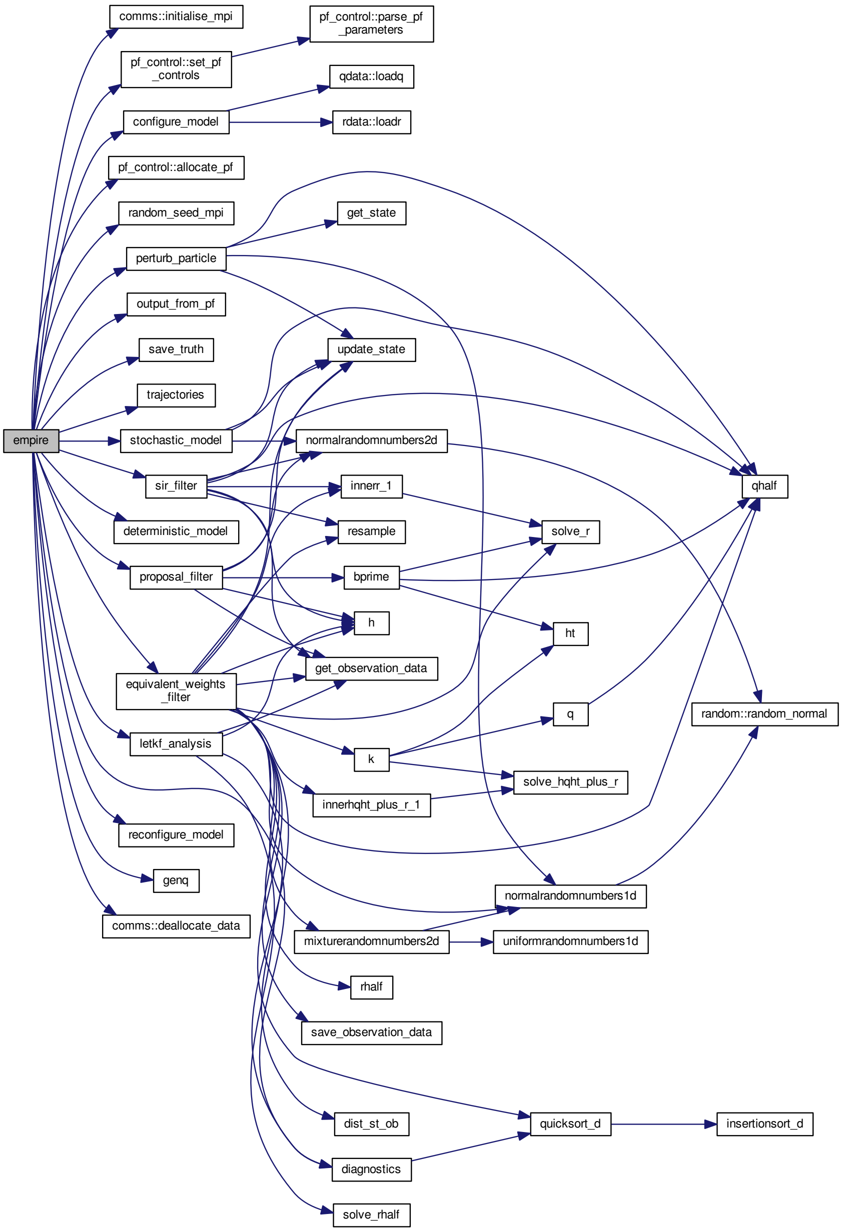
8.3.1 Function/Subroutine Documentation

8.3.1.1 program empire ( )

the main program

Definition at line 37 of file pf_couple.f90.

Here is the call graph for this function:



8.4 src/controlers/pf_parameters.dat File Reference

8.5 src/controlers/sizes.f90 File Reference

Data Types

- module sizes

    Module that stores the dimension of observation and state spaces.

8.6 src/data/Qdata.f90 File Reference

Data Types

- module qdata

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

8.7 src/data/Rdata.f90 File Reference

Data Types

- module rdata

  Module to hold user supplied data for R observation error covariance matrix.
- module hqht_plus_r

8.8 src/DOC_README.txt File Reference

8.9 src/filters/deterministic_model.f90 File Reference

Functions/Subroutines

- subroutine deterministic_model

  subroutine to simply move the model forward in time one timestep

8.9.1 Function/Subroutine Documentation

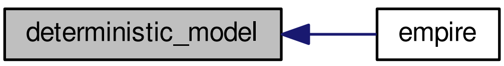
8.9.1.1 subroutine deterministic_model ( )

subroutine to simply move the model forward in time one timestep

PAB 21-05-2013

Definition at line 33 of file deterministic_model.f90.

Here is the caller graph for this function:



8.10 src/filters/eakf_analysis.f90 File Reference

Functions/Subroutines

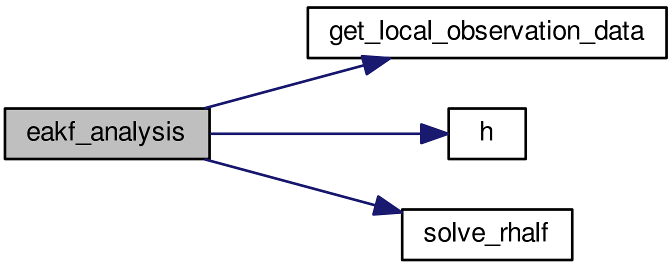
- subroutine eakf_analysis (num_hor, num_ver, this_hor, this_ver, boundary, x, N, stateDim, obsDim, rho)

8.10.1 Function/Subroutine Documentation

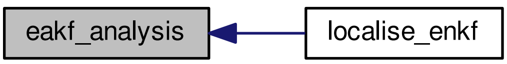
8.10.1.1 subroutine eakf_analysis (integer, intent(in) *num_hor*, integer, intent(in) *num_ver*, integer, intent(in) *this_hor*, integer, intent(in) *this_ver*, integer, intent(in) *boundary*, real(kind=rk), dimension(statedim,n), intent(inout) *x*, integer, intent(in) *N*, integer, intent(in) *stateDim*, integer, intent(in) *obsDim*, real(kind=rk), intent(in) *rho*)

Definition at line 27 of file eakf_analysis.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11 src/filters/enkf_specific.f90 File Reference

Functions/Subroutines

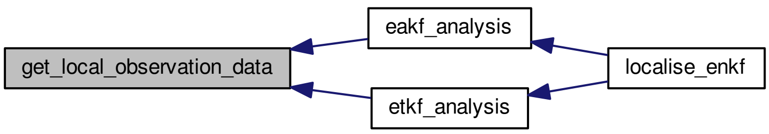
- subroutine h_local (num_hor, num_ver, this_hor, this_ver, boundary, nrhs, stateDim, x, obsDim, y)
- subroutine solve_rhalf_local (num_hor, num_ver, this_hor, this_ver, boundary, nrhs, obsDim, y, v)
- subroutine get_local_observation_data (num_hor, num_ver, this_hor, this_ver, boundary, obsDim, y)
-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

8.11.1 Function/Subroutine Documentation

8.11.1.1 subroutine get_local_observation_data (integer, intent(in) *num_hor*, integer, intent(in) *num_ver*, integer, intent(in) *this_hor*, integer, intent(in) *this_ver*, integer, intent(in) *boundary*, integer, intent(in) *obsDim*, real(kind=rk), dimension(obsdim), intent(out) *y*)

Definition at line 83 of file enkf_specific.f90.

Here is the caller graph for this function:



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

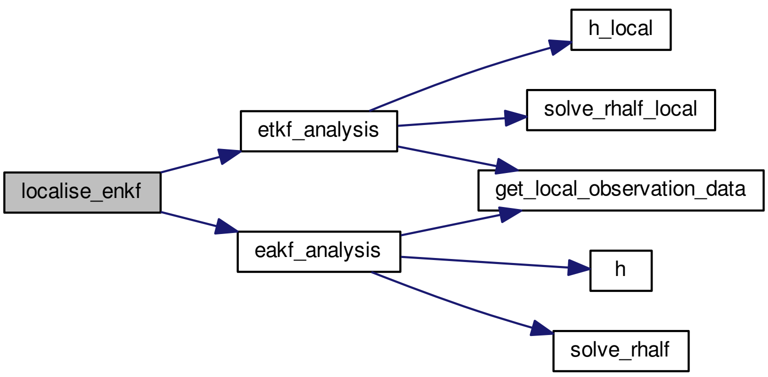
Definition at line 27 of file enkf_specific.f90.

Here is the caller graph for this function:

8.11.1.3 subroutine localise_enkf (integer, intent(in) *enkf_analysis*)

Definition at line 142 of file enkf_specific.f90.

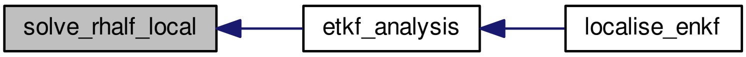
Here is the call graph for this function:



8.11.1.4 subroutine solve_rhalf_local (integer, intent(in) *num_hor*, integer, intent(in) *num_ver*, integer, intent(in) *this_hor*, integer, intent(in) *this_ver*, integer, intent(in) *boundary*, integer, intent(in) *nrhs*, integer, intent(in) *obsDim*, real(kind=rk), dimension(obsdim,nrhs), intent(in) *y*, real(kind=rk), dimension(obsdim,nrhs), intent(out) *v*)

Definition at line 69 of file enkf_specific.f90.

Here is the caller graph for this function:



8.12 src/filters/equivalent_weights_filter.f90 File Reference

Functions/Subroutines

- subroutine equivalent_weights_filter

    subroutine to do the equivalent weights step

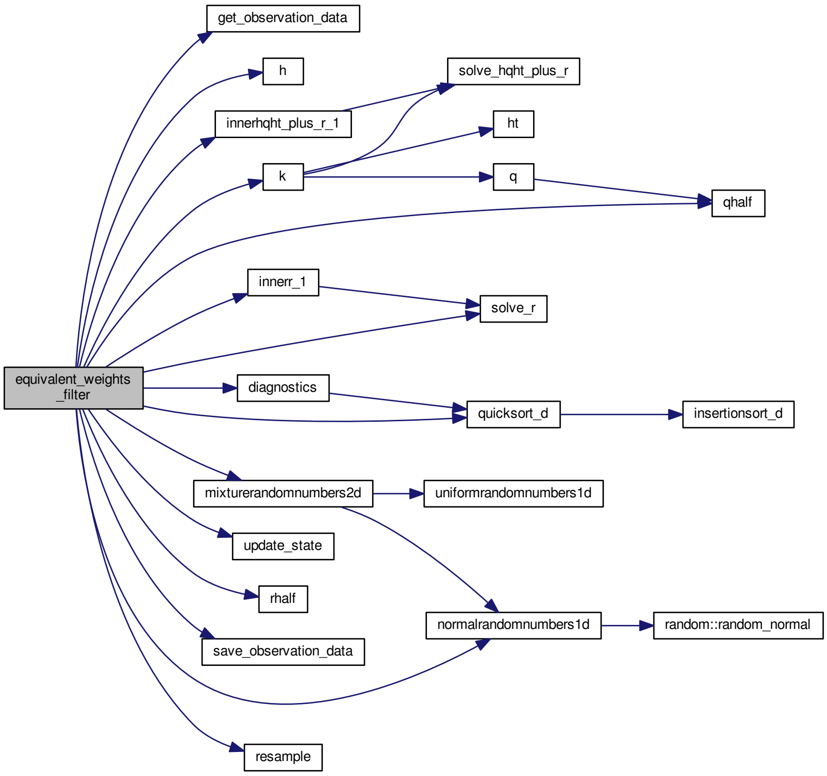
8.12.1 Function/Subroutine Documentation

8.12.1.1 subroutine equivalent_weights_filter ( )

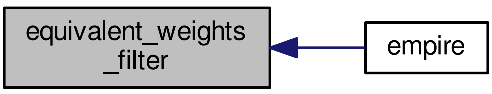
subroutine to do the equivalent weights step

Definition at line 29 of file equivalent_weights_filter.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.13 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

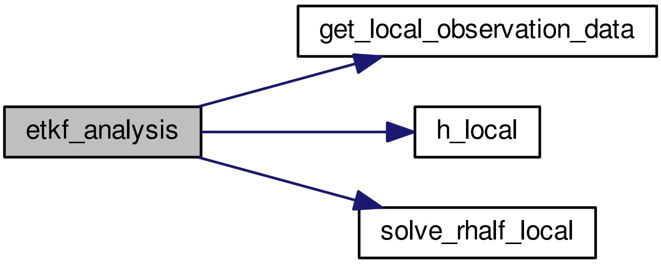
8.13.1 Function/Subroutine Documentation

8.13.1.1 subroutine etkf_analysis (integer, intent(in) *num_hor*, integer, intent(in) *num_ver*, integer, intent(in) *this_hor*, integer, intent(in) *this_ver*, integer, intent(in) *boundary*, real(kind=rk), dimension(statedim,n), intent(inout) *x*, integer, intent(in) *N*, integer, intent(in) *stateDim*, integer, intent(in) *obsDim*, real(kind=rk), intent(in) *rho*)

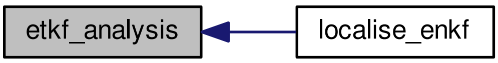
subroutine to perform the ensemble transform Kalman filter

Definition at line 34 of file etkf_analysis.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.14 src/filters/letkf_analysis.f90 File Reference

Functions/Subroutines

- subroutine letkf_analysis

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

8.14.1 Function/Subroutine Documentation

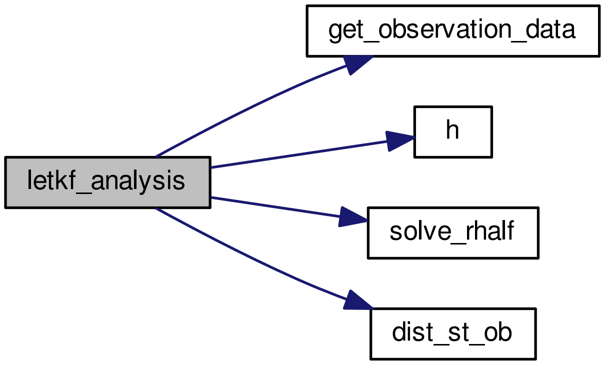
8.14.1.1 subroutine letkf_analysis ( )

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

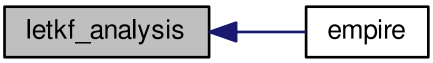
The observation

Definition at line 35 of file letkf_analysis.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.15 src/filters/proposal_filter.f90 File Reference

Functions/Subroutines

- subroutine proposal_filter

  Subroutine to perform nudging in the proposal step of EWPF.

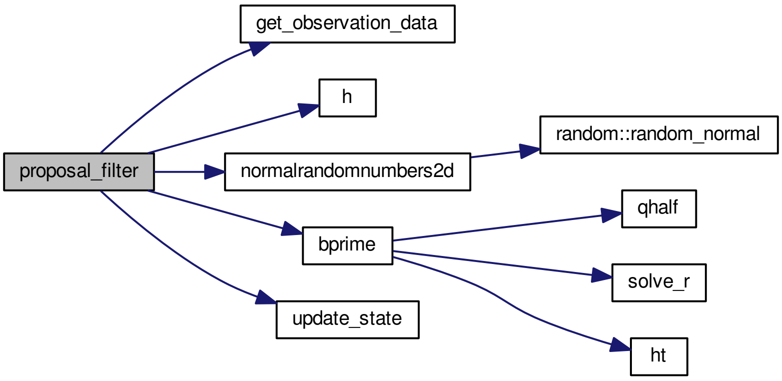
8.15.1 Function/Subroutine Documentation

8.15.1.1 subroutine proposal_filter ( )

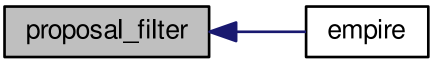
Subroutine to perform nudging in the proposal step of EWPF.

Definition at line 33 of file proposal_filter.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.16 src/filters/sir_filter.f90 File Reference

Functions/Subroutines

- subroutine sir_filter

  Subroutine to perform SIR filter (Sequential Importance Resampling)

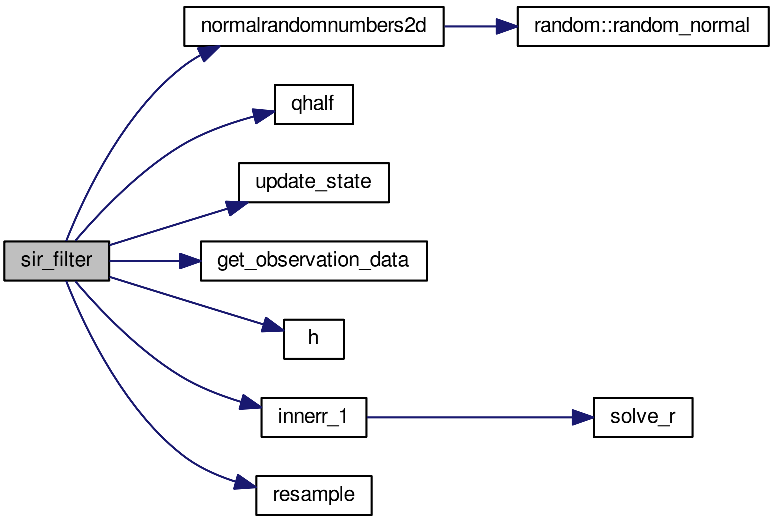
8.16.1 Function/Subroutine Documentation

8.16.1.1 subroutine sir_filter ( )

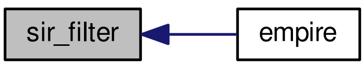
Subroutine to perform SIR filter (Sequential Importance Resampling)

Definition at line 28 of file sir_filter.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.17 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)

8.17.1 Function/Subroutine Documentation

8.17.1.1 subroutine check_scaling (real(kind=rk), dimension(state_dim), intent(in) *x*, real(kind=rk), dimension(state_dim), intent(in) *fx*, real(kind=rk), dimension(state_dim), intent(in) *b*, real(kind=rk), dimension(9), intent(inout) *scales*)

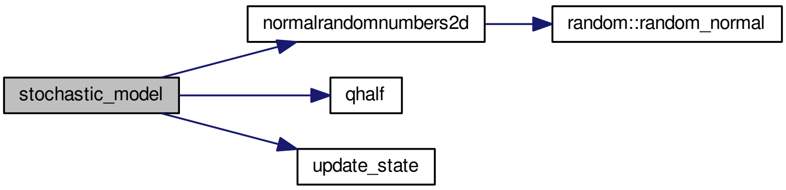
Definition at line 80 of file stochastic_model.f90.

8.17.1.2 subroutine stochastic_model ( )

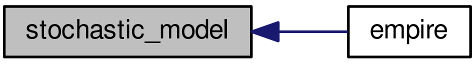
subroutine to simply move the model forward in time one timestep PAB 21-05-2013

Definition at line 32 of file stochastic_model.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.18 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.

8.18.1 Function/Subroutine Documentation

8.18.1.1 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*)

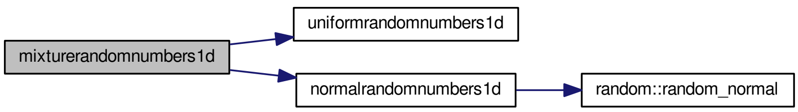
generate one dimensional vector drawn from mixture density

Parameters

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

Definition at line 90 of file gen_rand.f90.

Here is the call graph for this function:



8.18.1.2 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*)

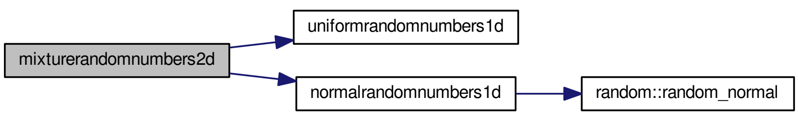
generate two dimensional vector, each drawn from mixture density

Parameters

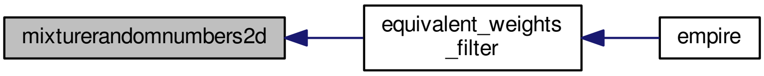
in	*mean*	Mean of normal distribution
in	*stdev*	Standard deviation of normal distribution
in	*ufac*	half-width of uniform distribution that is centered on the mean
in	*epsi*	Proportion controlling mixture draw. if random_number > epsi then draw from uniform, else normal
in	*n*	first dimension of output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

Definition at line 125 of file gen_rand.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.1.3 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*)

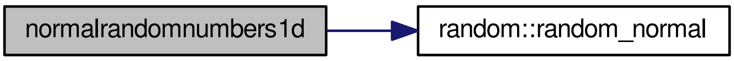
generate one dimension of Normal random numbers

Parameters

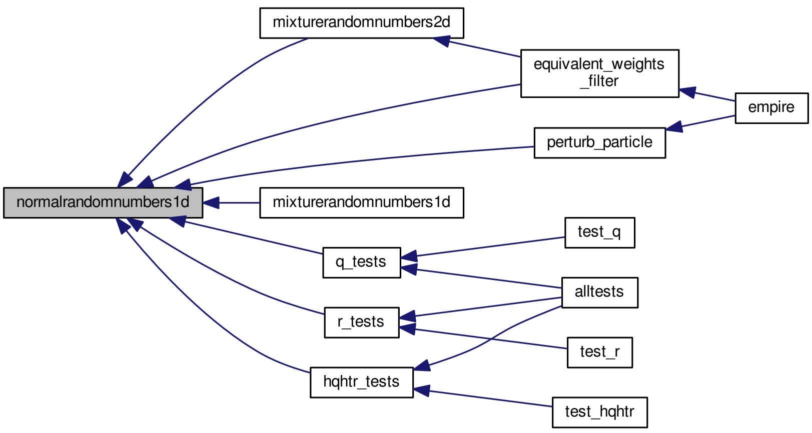
in	*n*	n size of output vector
in	*mean*	mean mean of normal distribution
in	*stdev*	stdev Standard Deviation of normal distribution
out	*phi*	phi n dimensional normal random numbers

Definition at line 43 of file gen_rand.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.1.4 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*)

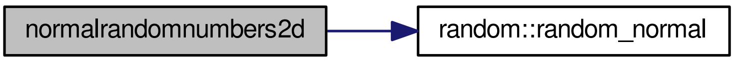
generate two dimensional Normal random numbers

Parameters

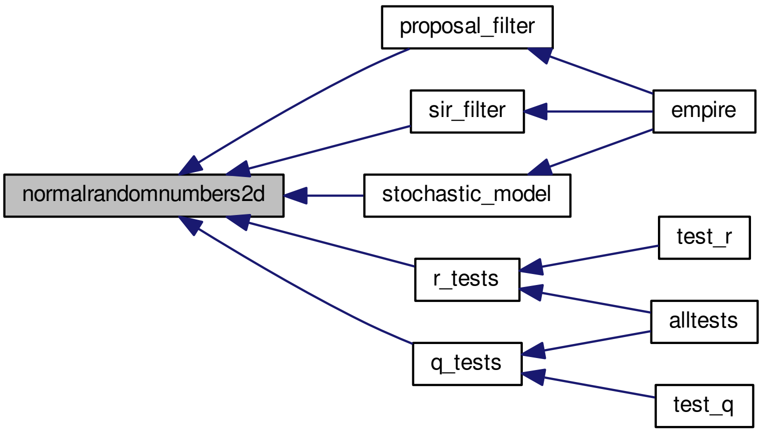
in	*n*	n first dimension of output vector
in	*k*	k second dimension of output vector
in	*mean*	mean mean of normal distribution
in	*stdev*	stdev Standard Deviation of normal distribution
out	*phi*	phi n,k dimensional normal random numbers

Definition at line 60 of file gen_rand.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.18.1.5 subroutine random_seed_mpi (integer, intent(in) *pfid*)

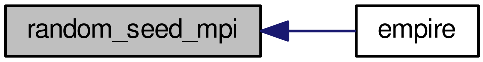
Subroutine to set the random seed across MPI threads.

Parameters

in	*pfid*	The process identifier of the MPI process

Definition at line 151 of file gen_rand.f90.

Here is the caller graph for this function:



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

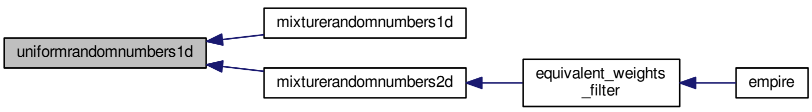
generate one dimension of uniform random numbers

Parameters

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

Definition at line 28 of file gen_rand.f90.

Here is the caller graph for this function:



8.19 src/operations/operator_wrappers.f90 File Reference

Functions/Subroutines

- subroutine k (y, x)

  Subroutine to apply K to a vector y in observation space where $K := QH^T(HQH^T+R)^{-1}$.
- subroutine innerr_1 (y, w)

  subroutine to compute the inner product with R^{-1}
- subroutine innerhqht_plus_r_1 (y, w)

  subroutine to compute the inner product with $(HQH^T+R)^{-1}$
- subroutine bprime (y, x, QHtR_1y, normaln, betan)

  subroutine to calculate nudging term and correlated random errors efficiently

8.19.1 Function/Subroutine Documentation

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

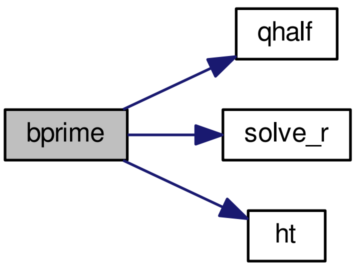
subroutine to calculate nudging term and correlated random errors efficiently

Parameters

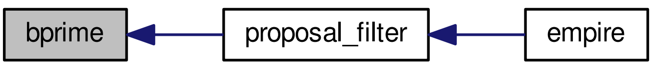
in	*y*	(obs_dim,pf%count) vectors of innovations $y-H(x^{n-1})$
out	*x*	(state_dim,pf%count) vectors of $\rho H^T R^{-1}[y-H(x^{n-1})]$
out	*QHtR_1y*	(state_dim,pf%count) vectors of $\rho Q H^T R^{-1}[y-H(x^{n-1})]$
in	*normaln*	(state_dim,pf%count) uncorrelated random vectors such that normaln(:,i)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

Definition at line 155 of file operator_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

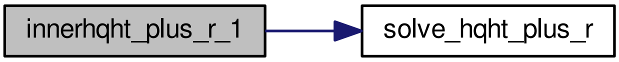
subroutine to compute the inner product with $(HQH^T+R)^{-1}$

Parameters

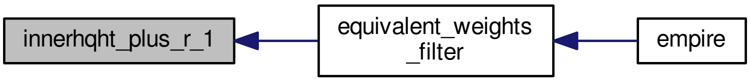
in	*y*	vector in observation space
out	*w*	scalar with value $y^T R^{-1} y$

Definition at line 91 of file operator_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

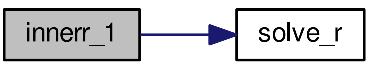
subroutine to compute the inner product with R^{-1}

Parameters

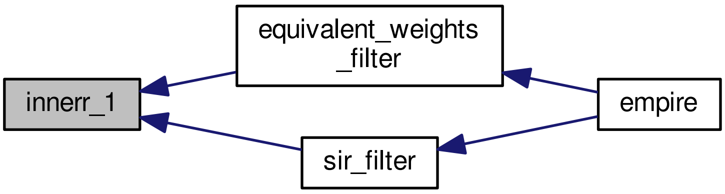
in	*y*	multiple vectors in observation space (pf%count of them)
out	*w*	multiple scalars (pf%count) where w(i) has the value $y(:,i)^T R^{-1} y(:,i)$

Definition at line 65 of file operator_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



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

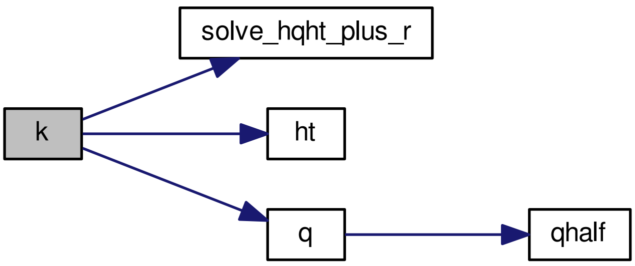
Subroutine to apply K to a vector y in observation space where $K := QH^T(HQH^T+R)^{-1}$.

Parameters

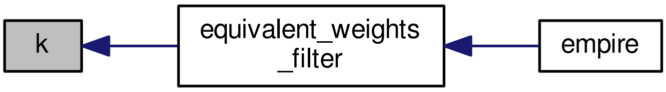
in	*y*	vector in observation space
out	*x*	vector in state space

Definition at line 32 of file operator_wrappers.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20 src/operations/perturb_particle.f90 File Reference

Functions/Subroutines

- subroutine perturb_particle (x)

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

- subroutine update_state (state, fpsi, kgain, betan)

    Subroutine to update the state.

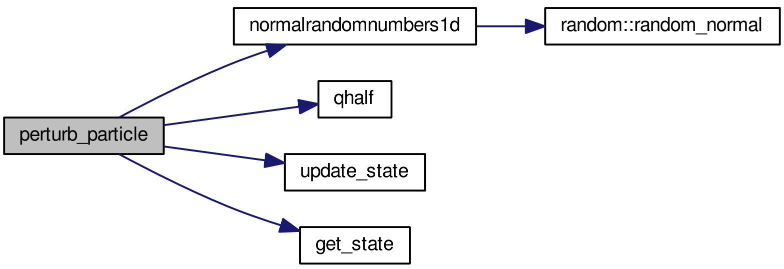
8.20.1 Function/Subroutine Documentation

8.20.1.1 subroutine perturb_particle (real(kind=rk), dimension(state_dim), intent(inout) *x*)

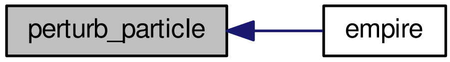
Subroutine to perturb state vector with normal random vector drawn from $\mathcal{N}(0,Q)$.

Definition at line 30 of file perturb_particle.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.1.2 subroutine update_state (real(kind=rk), dimension(state_dim), intent(out) *state*, real(kind=rk), dimension(state_dim), intent(in) *fpsi*, real(kind=rk), dimension(state_dim), intent(in) *kgain*, real(kind=rk), dimension(state_dim), intent(inout) *betan*)

Subroutine to update the state.

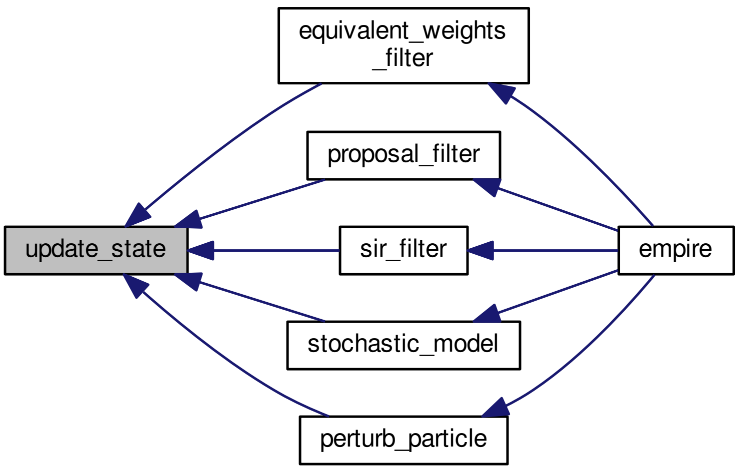
This can be changed for the specific model if it needs to be

Parameters

`in`	*fpsi*	deterministic model update $f(x^{n-1})$
`in`	*kgain*	nudging term
`in,out`	*betan*	Stochastic term
`out`	*state*	The updated state vector

Definition at line 95 of file perturb_particle.f90.

Here is the caller graph for this function:



8.21 src/operations/resample.f90 File Reference

Functions/Subroutines

- subroutine resample

    Subroutine to perform Universal Importance Resampling.

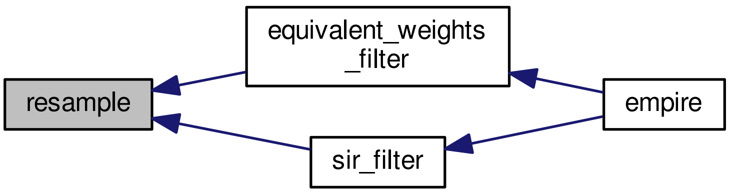
8.21.1 Function/Subroutine Documentation

8.21.1.1 subroutine resample ( )

Subroutine to perform Universal Importance Resampling.

Definition at line 28 of file resample.f90.

Here is the caller graph for this function:



8.22 src/tests/alltests.f90 File Reference

Functions/Subroutines

- program alltests

    program to run all tests of user specific functions

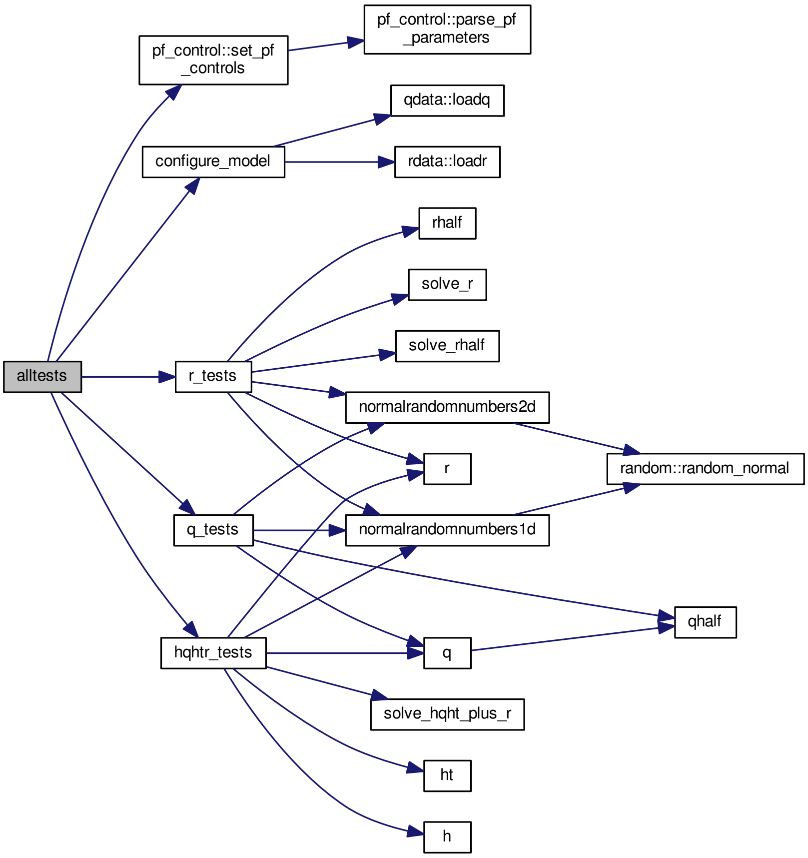
8.22.1 Function/Subroutine Documentation

8.22.1.1 program alltests ( )

program to run all tests of user specific functions

Definition at line 31 of file alltests.f90.

Here is the call graph for this function:



8.23 src/tests/test_h.f90 File Reference

8.24 src/tests/test_hqhtr.f90 File Reference

Functions/Subroutines

- program test_hqhtr

  program to run tests of user supplied linear solve

8.24.1 Function/Subroutine Documentation

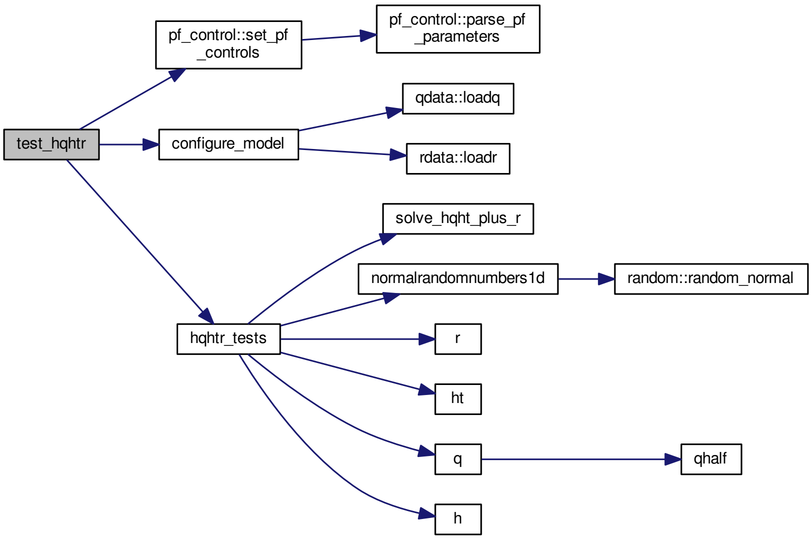
8.24.1.1 program test_hqhtr ( )

program to run tests of user supplied linear solve

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

Definition at line 33 of file test_hqhtr.f90.

Here is the call graph for this function:



8.25 src/tests/test_q.f90 File Reference

Functions/Subroutines

- program test_q

  program to run tests of user supplied model error covariance matrix

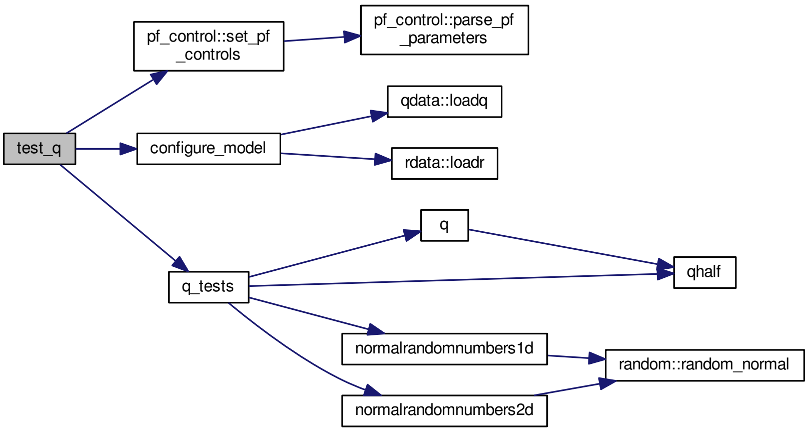
8.25.1 Function/Subroutine Documentation

8.25.1.1 program test_q ( )

program to run tests of user supplied model error covariance matrix

Definition at line 31 of file test_q.f90.

Here is the call graph for this function:



8.26 src/tests/test_r.f90 File Reference

Functions/Subroutines

- program test_r

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

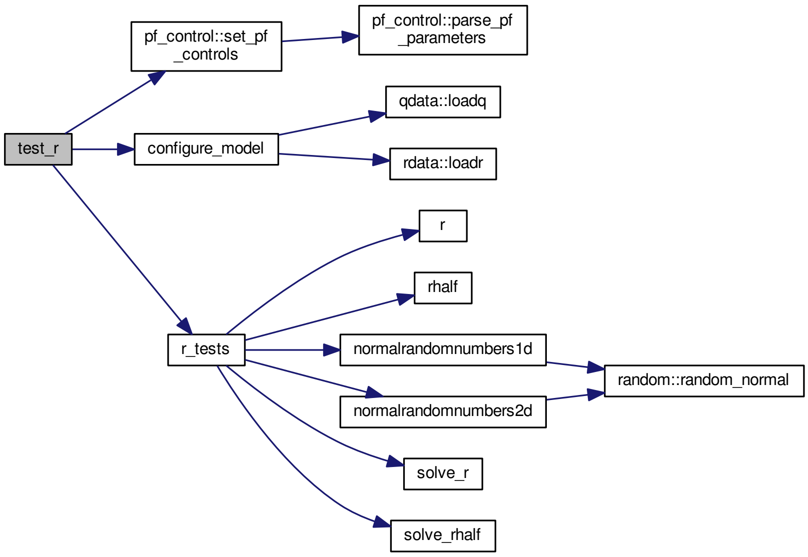
8.26.1 Function/Subroutine Documentation

8.26.1.1 program test_r ( )

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

Definition at line 31 of file test_r.f90.

Here is the call graph for this function:



8.27 src/tests/tests.f90 File Reference

Functions/Subroutines

- subroutine r_tests ()
- subroutine q_tests ()
- subroutine hqhtr_tests ()

8.27.1 Function/Subroutine Documentation

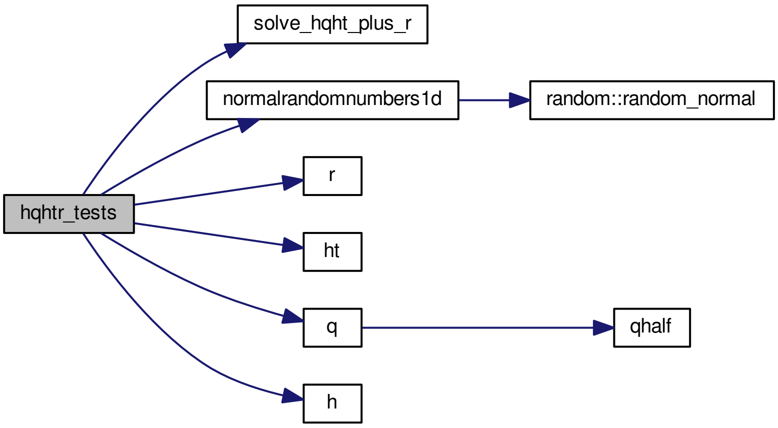
8.27.1.1 subroutine hqhtr_tests ( )

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

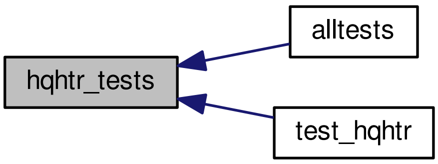
This should check the operation $(HQH^T + R)^{-1}$ is working

Definition at line 879 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

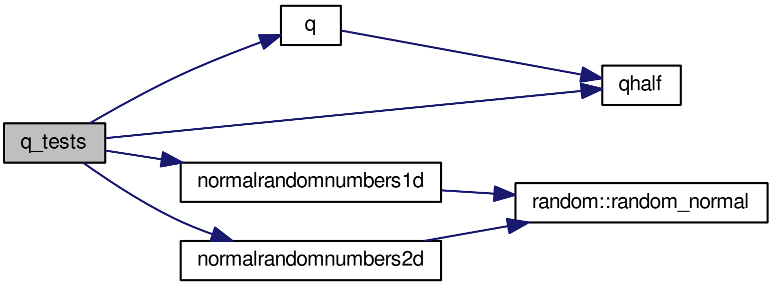


8.27.1.2 subroutine q_tests ( )

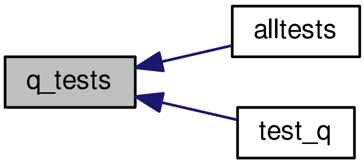
These are some tests to check that the model error covariance matrix is implemented correctly

Definition at line 675 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

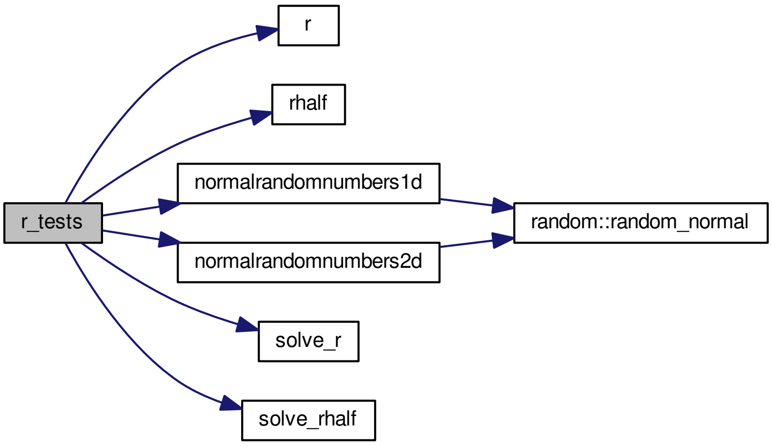


8.27.1.3 subroutine r_tests ( )

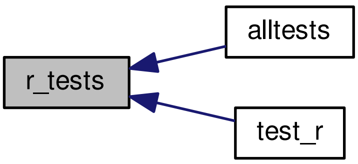
These are some tests to check that the observation error covariance matrix is implemented correctly

Definition at line 257 of file tests.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.28 src/utils/comms.f90 File Reference

Data Types

- module comms

  Module containing EMPIRE coupling data.

8.29 src/utils/data_io.f90 File Reference

Functions/Subroutines

- subroutine get_observation_data (y)

  Subroutine to read observation from a file
  Uses pftimestep to determine which observation to read.
- subroutine save_observation_data (y)

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

- subroutine save_truth (x)

  Subroutine to save truth to a file
  .

- subroutine output_from_pf

  subroutine to ouput data from the filter

- subroutine save_state (state, filename)

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

- subroutine get_state (state, filename)

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

8.29.1 Function/Subroutine Documentation

8.29.1.1 subroutine get_observation_data (real(kind=rk), dimension(obs_dim), intent(out) *y*)

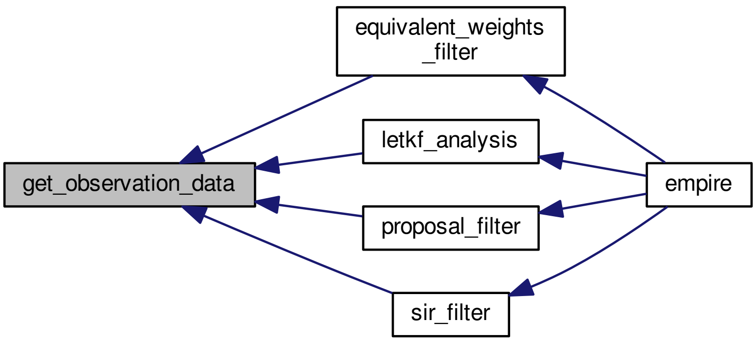
Subroutine to read observation from a file
Uses pftimestep to determine which observation to read.

Parameters

out	*y*	The observation

Definition at line 32 of file data_io.f90.

Here is the caller graph for this function:



8.29.1.2 subroutine get_state (real(kind=rk), dimension(state_dim), intent(out) *state*, character(14), intent(in) *filename*)

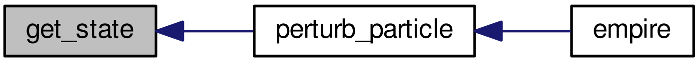
subroutine to write the state vector to a named file as an unformatted fortran file

Parameters

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

Definition at line 283 of file data_io.f90.

Here is the caller graph for this function:

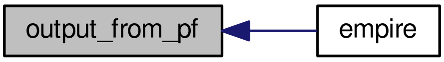


8.29.1.3 subroutine output_from_pf ( )

subroutine to ouput data from the filter

Definition at line 124 of file data_io.f90.

Here is the caller graph for this function:



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

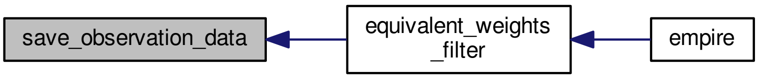
Subroutine to save observation to a file
Uses pftimestep to determine which observation to save.

Parameters

in	*y*	The observation

Definition at line 60 of file data_io.f90.

Here is the caller graph for this function:



8.29.1.5 subroutine save_state (real(kind=rk), dimension(state_dim), intent(in) *state,* character(14), intent(in) *filename*)

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

Parameters

in	*state*	the state vector
in	*filename*	the name of the file to save the state vector in

Definition at line 257 of file data_io.f90.

8.29.1.6 subroutine save_truth (real(kind=rk), dimension(state_dim), intent(in) *x*)

Subroutine to save truth to a file

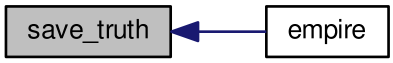
.

Parameters

in	*x*	The state vector

Definition at line 98 of file data_io.f90.

Here is the caller graph for this function:



8.30 src/utils/diagnostics.f90 File Reference

Functions/Subroutines

- subroutine diagnostics

  Subroutine to give output diagnositics such as rank histograms and trajectories.
- subroutine trajectories

  subroutine to output trajectories

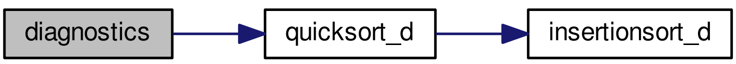
8.30.1 Function/Subroutine Documentation

8.30.1.1 subroutine diagnostics ( )

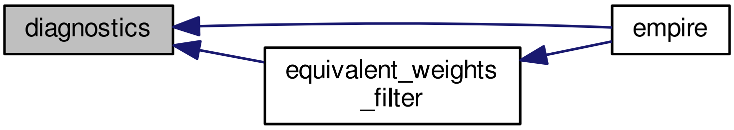
Subroutine to give output diagnositics such as rank histograms and trajectories.

Definition at line 31 of file diagnostics.f90.

Here is the call graph for this function:



Here is the caller graph for this function:

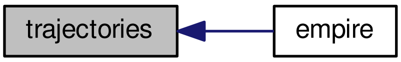


8.30.1.2 subroutine trajectories ( )

subroutine to output trajectories

Definition at line 203 of file diagnostics.f90.

Here is the caller graph for this function:



8.31 src/utils/genQ.f90 File Reference

Functions/Subroutines

- subroutine genq

  Subroutine to estimate Q from a long model run.

8.31.1 Function/Subroutine Documentation

8.31.1.1 subroutine genq ( )

Subroutine to estimate Q from a long model run.

Definition at line 28 of file genQ.f90.

Here is the caller graph for this function:

8.32 src/utils/histogram.f90 File Reference

Data Types

- module histogram_data

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

8.33 src/utils/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

8.33.1 Function/Subroutine Documentation

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

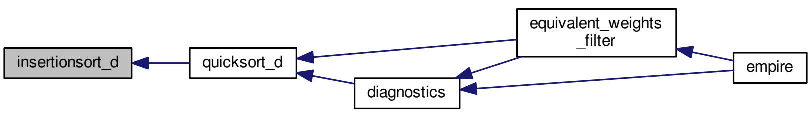
subroutine to sort using the insertionsort algorithm

Parameters

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

Definition at line 86 of file quicksort.f90.

Here is the caller graph for this function:



8.33.1.2 recursive subroutine quicksort_d (real(kind=kind(1.0d0)), dimension(na), intent(inout) *a,* integer, intent(in) *na*)

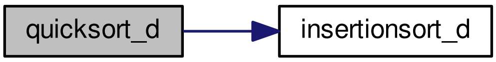
subroutine to sort using the quicksort algorithm

Parameters

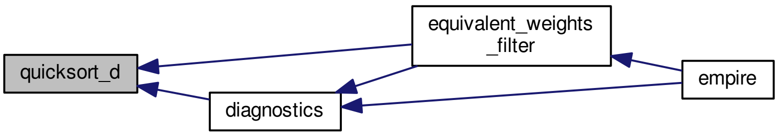
in,out	*a*	array of doubles to be sorted
in	*na*	dimension of array a

Definition at line 9 of file quicksort.f90.

Here is the call graph for this function:



Here is the caller graph for this function:



8.34 src/utils/random_d.f90 File Reference

Data Types

- module random

    A module for random number generation from the following distributions:

Index