PRPSEvolution

Generated by Doxygen 1.7.6.1

Thu Jul 25 2013 10:14:23

Contents

Chapter 1

Todo List

Member PRPSEvolution::Solve::Process::findSolutionCMA_ES_MkI ()
document

Member PRPSEvolution::Solve::Process::findSolutionCMA_ES_MkII ()
document

Member PRPSEvolution::Solve::Process::findSolutionSphere (Solve::ESStrategy strategy)
document

Member PRPSEvolution::Solve::Ueber9000< T >::evaluate)(const ChromosomeT< double > &)
document

Member PRPSEvolution::Solve::Ueber9000< T >::evaluateMkI)(const ChromosomeT< double > &)
document

Member PRPSEvolution::Solve::Ueber9000< T >::evaluateMkII)(const ChromosomeT< double > &, const ChromosomeT< double > &)
document

Member PRPSEvolution::Solve::Ueber9000< T >::evaluateMkIII)(const ChromosomeT< double > &, const ChromosomeT< int > &)
document

Member PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkI (const NRmatrix< T > &A, const ChromosomeT< double > &x, const NRvector< T > &b)
documentation

Member PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > &x)
document

Member PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > &x, const ChromosomeT< int > &n)
document

Member PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > &x1, const ChromosomeT< double > &x2)

document

Member PRPSEvolution::WholeTomatoMkI_A::mkI (const NRmatrix< Doub > &A, const SearchPointType &x, const NRvector< Doub > &b) const

documentation

Member PRPSEvolution::WholeTomatoMkI_B::mkI (const NRmatrix< Doub > &A, const SearchPointType &x, const NRvector< Doub > &b) const

documentation

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 PRPSError Namespace Reference

Namespaces

- namespace FileIO

Variables

- const int okay = 0
- const int general = -1
- const int critical = 10

5.1.1 Variable Documentation

5.1.1.1 const int PRPSError::critical = 10

this is devastating

5.1.1.2 const int PRPSError::general = -1

if no other error fits

5.1.1.3 const int PRPSError::okay = 0

this ist no error

5.2 PRPSError::FileIO Namespace Reference

Variables

- const int okay = 0
- const int generalError = -1
- const int fnf = -2
- const int inputmalformed = -3

5.2.1 Variable Documentation

5.2.1.1 const int PRPSError::FileIO::fnf = -2

file not found error

5.2.1.2 const int PRPSError::FileIO::generalError = -1

if no other error fits

5.2.1.3 const int PRPSError::FileIO::inputmalformed = -3

malformed input

5.2.1.4 const int PRPSError::FileIO::okay = 0

this ist no error

5.3 PRPSEvolution Namespace Reference

Namespaces

- namespace Calibration
- namespace Exceptions
- namespace Permutate
- namespace Positioning
- namespace Solve

Classes

- struct Normalizer
- struct Constants
- struct System
- struct WholeTomatoMkI_A
- struct WholeTomatoMkI_B
- struct WholeTomatoMkII

Enumerations

- enum NormalizatioMethodes { Native, B, CMPLX, RND }

Functions

- ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkI_A, shark::soo::RealValuedObjectiveFunctionFactory)
- ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkI_B, shark::soo::RealValuedObjectiveFunctionFactory)
- ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkII, soo::RealValuedObjectiveFunctionFactory)

Variables

- const int ANTENNA_AMOUNT = 8
- const int EXPECTED_LINES_CALIBRATION_FILE = 4
- const int EXPECTED_VALUES_CALIBRATION_FILE = ANTENNA_AMOUNT
- const int EXPECTED_LINES_COORD_FILE = ANTENNA_AMOUNT
- const int EXPECTED_VALUES_COORD_FILE = 3
- const int EXPECTED_LINES_SYSTEM_INI_FILE = 2
- const int MAT_ROWS = 3
- const int MAT_COLS = 10
- const int CALIBRATION_POINTS_AVAILABLE = 4
- const int EXPECTED_LINES_MEASUREMENT_FILE = ANTENNA_AMOUNT
- const int EXPECTED_VALUES_MEASUREMENT_FILE = 2
- const int DATA_NV = 65535
- const int DEFAULT_MIN_GROUP_SIZE = 4

5.3.1 Detailed Description

This file contains structures and classes belonging to the system itself

5.3.2 Enumeration Type Documentation

5.3.2.1 enum PRPSEvolution::NormalizatioMethodes

Enumerator:

Native

B

CMPLX

RND

5.3.3 Function Documentation

5.3.3.1 PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkl_A , shark::soo::RealValuedObjectiveFunctionFactory)

5.3.3.2 PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkl_B , shark::soo::RealValuedObjectiveFunctionFactory)

5.3.3.3 PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkII , soo::RealValuedObjectiveFunctionFactory)

5.3.4 Variable Documentation

5.3.4.1 const int PRPSEvolution::ANTENNA_AMOUNT = 8

5.3.4.2 const int PRPSEvolution::CALIBRATION_POINTS_AVAILABLE = 4

5.3.4.3 const int PRPSEvolution::DATA_NV = 65535

5.3.4.4 const int PRPSEvolution::DEFAULT_MIN_GROUP_SIZE = 4

5.3.4.5 const int PRPSEvolution::EXPECTED_LINES_CALIBRATION_FILE = 4

5.3.4.6 const int PRPSEvolution::EXPECTED_LINES_COORD_FILE = ANTENNA_AMOUNT

5.3.4.7 const int PRPSEvolution::EXPECTED_LINES_MEASUREMENT_FILE = ANTENNA_AMOUNT

5.3.4.8 const int PRPSEvolution::EXPECTED_LINES_SYSTEM_INI_FILE = 2

5.3.4.9 const int PRPSEvolution::EXPECTED_VALUES_CALIBRATION_FILE = ANTENNA_AMOUNT

5.3.4.10 const int PRPSEvolution::EXPECTED_VALUES_COORD_FILE = 3

5.3.4.11 const int PRPSEvolution::EXPECTED_VALUES_MEASUREMENT_FILE = 2

5.3.4.12 const int PRPSEvolution::MAT_COLS = 10

5.3.4.13 const int PRPSEvolution::MAT_ROWS = 3

5.4 PRPSEvolution::Calibration Namespace Reference

Classes

- struct performCalibration

5.5 PRPSEvolution::Exceptions Namespace Reference

Namespaces

- namespace Calibration
- namespace FileIO
- namespace General
- namespace Permutation
- namespace Solve

5.6 PRPSEvolution::Exceptions::Calibration Namespace Reference

5.7 PRPSEvolution::Exceptions::FileIO Namespace Reference

Classes

- struct FileNotFound
- struct MalformedInput
- struct OutputFailure

5.8 PRPSEvolution::Exceptions::General Namespace Reference

Classes

- struct NotImplemented

5.9 PRPSEvolution::Exceptions::Permutation Namespace Reference

5.10 PRPSEvolution::Exceptions::Solve Namespace Reference

5.11 PRPSEvolution::Permutate Namespace Reference

Classes

- struct AntennaPermutations
- struct permuteAntennas

Functions

- int Factorial (int x)
- template<typename Iterator >
  bool next_combination (const Iterator first, Iterator k, const Iterator last)

Variables

- const int MAX_PERMUTATION_AMOUNT = 35

5.11.1 Function Documentation

5.11.1.1 int PRPSEvolution::Permutate::Factorial (int *x*) [inline]

5.11.1.2 template<typename Iterator > bool PRPSEvolution::Permutate::next_combination (const Iterator *first*, Iterator *k*, const Iterator *last*) [inline]

5.11.2 Variable Documentation

5.11.2.1 const int PRPSEvolution::Permutate::MAX_PERMUTATION_AMOUNT = 35

The maximum amount of Permutations for one reference antenna, we need this const-expression for the template

5.12 PRPSEvolution::Positioning Namespace Reference

Classes

- struct CoordContainer

5.13 PRPSEvolution::Solve Namespace Reference

Classes

- class PostProcessing
- class PreProcessing
- class Process
- class Process_MkII
- struct ProblemDimensions
- struct solveresult_t
- struct Ueber9000

Enumerations

- enum SelectBy { ConditionNumber, Random, AllPossible, Best10ByCN, AllFrom4Ant }
- enum ESStrategy { OnePlusOne, MuPlusLambda, MuCommaLambda, MuCommaLambda_MkII, MuPlusLambda_MkII, CMA_ES_MkI, CMA_ES_MkII }
- enum Models { WholeTomatoMkI, WholeTomatoMkII, TestSphere }

Functions

- double meanFromVector (std::vector< double > &res)

Variables

- const int nConfigsForProcessing = 1
- std::mutex wMutex
- int _i_ = 0

5.13.1 Enumeration Type Documentation

5.13.1.1 enum PRPSEvolution::Solve::ESStrategy

Represents the ES-strategy to find a solution

Enumerator:

OnePlusOne

$$[1+1]-ES$$

MuPlusLambda

$$[\mu+\lambda]-ES$$

MuCommaLambda

$$[\mu,\lambda]-ES$$

MuCommaLambda_MkII

MuPlusLambda_MkII

CMA_ES_MkI

CMA_ES_MkII

5.13.1.2 enum PRPSEvolution::Solve::Models

Models are defined here

Enumerator:

WholeTomatoMkI

WholeTomatoMkII

TestSphere

5.13.1.3 enum PRPSEvolution::Solve::SelectBy

Represents the selection method for the Matrix A that will be used for the solution

Enumerator:

ConditionNumber

Random

AllPossible

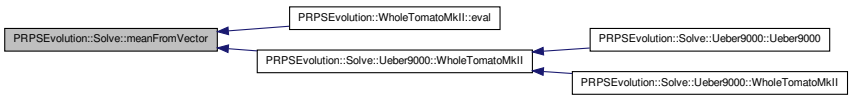
Best10ByCN

AllFrom4Ant

5.13.2 Function Documentation

5.13.2.1 double PRPSEvolution::Solve::meanFromVector (std::vector< double > & *res*) [inline]

Here is the caller graph for this function:



5.13.3 Variable Documentation

5.13.3.1 int PRPSEvolution::Solve::_i_ = 0

5.13.3.2 const int PRPSEvolution::Solve::nConfigsForProcessing = 1

5.13.3.3 std::mutex PRPSEvolution::Solve::wMutex

Chapter 6

Class Documentation

6.1 PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T > Struct Template Reference

`#include <permutate.h>`

Public Member Functions

- AntennaPermutations (void)

Static Public Member Functions

- static void dump_matrix (NRmatrix< T > mat)
- static void dump_matrix_2_file (std::ofstream &f, NRmatrix< T > mat)

Public Attributes

- std::array< NRmatrix< T >, N_MAT > mat
- std::array< std::string, N_MAT > names

template<std::size_t N_MAT, typename T> struct PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >

6.1.1 Constructor & Destructor Documentation

6.1.1.1 template<std::size_t N_MAT, typename T > PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >::AntennaPermutations (void) [inline]

6.1.2 Member Function Documentation

6.1.2.1 template<std::size_t N_MAT, typename T > static void PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >::dump_matrix (NRmatrix< T > *mat*) [inline, static]

6.1.2.2 template<std::size_t N_MAT, typename T > static void PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >::dump_matrix_2_file (std::ofstream & *f,* NRmatrix< T > *mat*) [inline, static]

6.1.3 Member Data Documentation

6.1.3.1 template<std::size_t N_MAT, typename T > std::array< NRmatrix< T >, N_MAT > PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >::mat

6.1.3.2 template<std::size_t N_MAT, typename T > std::array< std::string, N_MAT > PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >::names

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

- trunk/libPermutate/permutate.h

6.2 PRPSEvolution::Constants Struct Reference

`#include <prpsevolutionsystem.h>`

Public Member Functions

- Constants ()
- Constants (const PRPSEvolution::Constants &c)

Public Attributes

- double a_1
- double a_2
- double lambda
- double f_mess
- double c_0

6.2.1 Constructor & Destructor Documentation

6.2.1.1 PRPSEvolution::Constants::Constants ( ) [inline]

6.2.1.2 PRPSEvolution::Constants::Constants (const PRPSEvolution::Constants & *c*) [inline]

6.2.2 Member Data Documentation

6.2.2.1 double PRPSEvolution::Constants::a_1

6.2.2.2 double PRPSEvolution::Constants::a_2

6.2.2.3 double PRPSEvolution::Constants::c_0

6.2.2.4 double PRPSEvolution::Constants::f_mess

6.2.2.5 double PRPSEvolution::Constants::lambda

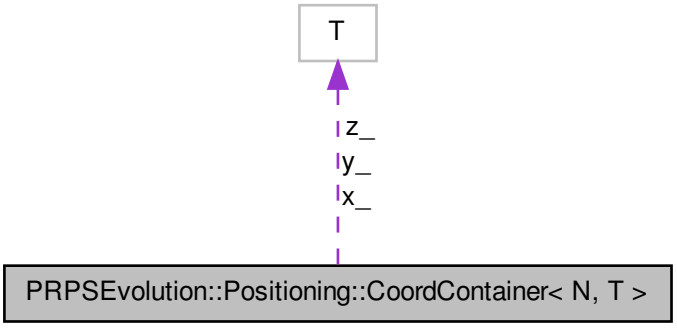
The documentation for this struct was generated from the following file:

- trunk/libPRPSSystem/prpsevolutionsystem.h

6.3 PRPSEvolution::Positioning::CoordContainer< N, T > Struct Template Reference

`#include <coords.h>`

Collaboration diagram for PRPSEvolution::Positioning::CoordContainer< N, T >:



Public Types

- typedef T value_type

Public Member Functions

- CoordContainer ()
- template<typename T1 >
  CoordContainer (T1 init)
- T & operator[] (std::size_t i)

Public Attributes

- T x_ [N]
- T y_ [N]
- T z_ [N]

template<std::size_t N, typename T> struct PRPSEvolution::Positioning::CoordContainer< N, T >

6.3.1 Member Typedef Documentation

6.3.1.1 template<std::size_t N, typename T> typedef T PRPSEvolution::Positioning::CoordContainer< N, T >::value_type

6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<std::size_t N, typename T > PRPSEvolution::Positioning::CoordContainer< N, T >::CoordContainer ( )

6.3.2.2 template<std::size_t N, typename T > template<typename T1 > PRPSEvolution::Positioning::CoordContainer< N, T >::CoordContainer (T1 *init*)

6.3.3 Member Function Documentation

6.3.3.1 template<std::size_t N, typename T > T & PRPSEvolution::Positioning::CoordContainer< N, T >::operator[] (std::size_t *i*)

6.3.4 Member Data Documentation

6.3.4.1 template<std::size_t N, typename T> T PRPSEvolution::Positioning::CoordContainer< N, T >::x_[N]

6.3.4.2 template<std::size_t N, typename T> T PRPSEvolution::Positioning::CoordContainer< N, T >::y_[N]

6.3.4.3 template<std::size_t N, typename T> T PRPSEvolution::Positioning::CoordContainer< N, T >::z_[N]

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

- trunk/include/coords.h

6.4 PRPSEvolution::Exceptions::FileIO::FileNotFound Struct Reference

`#include <PRPSEvolutionFIOExceptions.h>`

Public Member Functions

- const char * what () const noexcept

6.4.1 Member Function Documentation

6.4.1.1 const char* PRPSEvolution::Exceptions::FileIO::FileNotFound::what ( ) const [inline]

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

- trunk/include/PRPSEvolutionFIOExceptions.h

6.5 PRPSEvolution::Exceptions::FileIO::MalformedInput Struct Reference

`#include <PRPSEvolutionFIOExceptions.h>`

Public Member Functions

- const char * what () const noexcept

6.5.1 Member Function Documentation

6.5.1.1 const char* PRPSEvolution::Exceptions::FileIO::MalformedInput::what ( ) const [inline]

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

- trunk/include/PRPSEvolutionFIOExceptions.h

6.6 PRPSEvolution::Normalizer< N, T > Struct Template - Reference

`#include <normalizer.h>`

Public Member Functions

- Normalizer (NormalizatioMethodes method)
- std::array< T, N > complexNorm (const std::array< T, N > &p, const std::array< T, N > &a)
- std::array< T, N > randNorm ()
- std::array< T, N > normalize (std::array< T, N > phase, std::array< T, N > amp)

Public Attributes

- NormalizatioMethodes Method

template<std::size_t N, typename T> struct PRPSEvolution::Normalizer< N, T >

6.6.1 Constructor & Destructor Documentation

6.6.1.1 template<std::size_t N, typename T > PRPSEvolution::Normalizer< N, T >::Normalizer (NormalizatioMethodes *method*) [inline]

Constructor

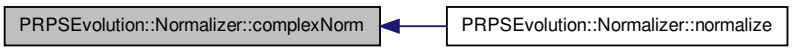
Parameters

in	*method*	Selects the Normalization function

6.6.2 Member Function Documentation

6.6.2.1 template<std::size_t N, typename T > std::array<T, N> PRPSEvolution::-Normalizer< N, T >::complexNorm (const std::array< T, N > & *p*, const std::array< T, N > & *a*) [inline]

Here is the caller graph for this function:



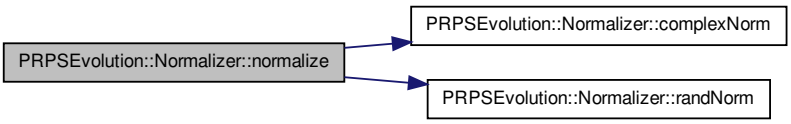
6.6.2.2 template<std::size_t N, typename T > std::array<T, N> PRPSEvolution::-Normalizer< N, T >::normalize (std::array< T, N > *phase*, std::array< T, N > *amp*) [inline]

Calculates the normalizations

Parameters

in	*phase*	The measured phase data
in	*amp*	The measured amplitude data

Here is the call graph for this function:



6.6.2.3 template<std::size_t N, typename T > std::array<T, N> PRPSEvolution::Normalizer< N, T >::randNorm ( ) [inline]

Here is the caller graph for this function:

6.6.3 Member Data Documentation

6.6.3.1 template<std::size_t N, typename T > NormalizatioMethodes PRPSEvolution::Normalizer< N, T >::Method

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

- trunk/libNormalizer/normalizer.h

6.7 PRPSEvolution::Exceptions::General::NotImplemented Struct - Reference

`#include <PRPSEvolutionGeneralExceptions.h>`

Public Member Functions

- const char * what () const noexcept

6.7.1 Detailed Description

Throw this if a Method is not implemented

6.7.2 Member Function Documentation

6.7.2.1 const char* PRPSEvolution::Exceptions::General::NotImplemented::what ( ) const [inline]

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

- trunk/include/PRPSEvolutionGeneralExceptions.h

6.8 PRPSEvolution::Exceptions::FileIO::OutputFailure Struct Reference

`#include <PRPSEvolutionFIOExceptions.h>`

Public Member Functions

- const char * what () const noexcept

6.8.1 Member Function Documentation

6.8.1.1 const char* PRPSEvolution::Exceptions::FileIO::OutputFailure::what ( ) const [inline]

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

- trunk/include/PRPSEvolutionFIOExceptions.h

6.9 PRPSEvolution::Calibration::performCalibration< N_ANTA, N_CALPOS, T > Struct Template Reference

`#include <calib.h>`

Public Member Functions

- performCalibration ()

6.9.1 Detailed Description

template<std::size_t N_ANTA, std::size_t N_CALPOS, typename T>struct PRPSEvolution::Calibration::performCalibration< N_ANTA, N_CALPOS, T >

This will perform the calibration stuff

6.9.2 Constructor & Destructor Documentation

6.9.2.1 template<std::size_t N_ANTA, std::size_t N_CALPOS, typename T > PRPSEvolution::Calibration::performCalibration< N_ANTA, N_CALPOS, T >::performCalibration ( )

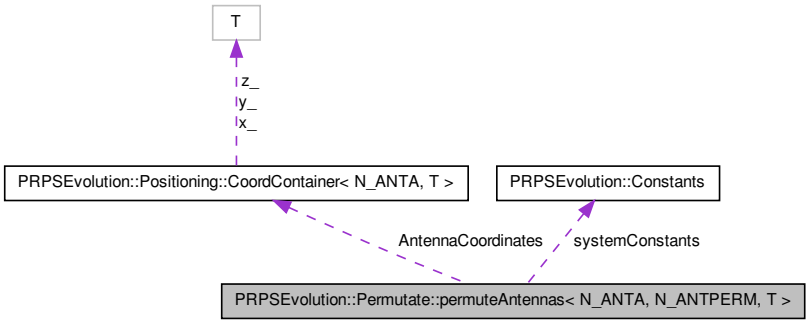
The documentation for this struct was generated from the following file:

- trunk/libCalibration/calib.h

6.10 PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T > Struct Template Reference

`#include <permutate.h>`

Collaboration diagram for PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >:



Public Member Functions

- permuteAntennas (const PRPSEvolution::Constants c)
- int rCoordFile ()
- int computePermutations (const PRPSEvolution::Constants &co)
- template<std::size_t NN, std::size_t MM>
  const NRmatrix< T > computeMatrix (const int ref, const int a1, const int a2, const int a3, const PRPSEvolution::Constants &co)
- NRmatrix< T > compute_d_k0_Mat ()
- void dumpConfigurationsToFile ()
- void dump_matrices_2_file ()

Public Attributes

- int ref
- PRPSEvolution::Constants systemConstants
- Positioning::CoordContainer < N_ANTA, T > AntennaCoordinates
- std::array < AntennaPermutations < N_ANTPERM, Doub >, N_ANTA > configurations
- NRmatrix< T > d_k0_mat

6.10.1 Detailed Description

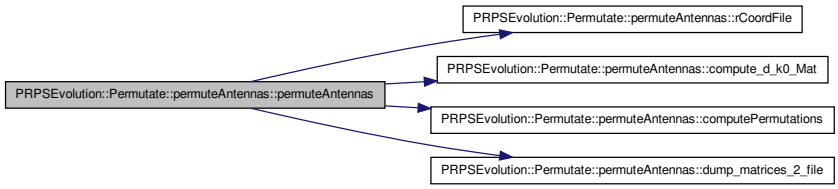
template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T>struct PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >

This will collect some stuff for calculating the permutation of the antennas

6.10.2 Constructor & Destructor Documentation

6.10.2.1 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::permuteAntennas (const PRPSEvolution::Constants *c*)

Here is the call graph for this function:



6.10.3 Member Function Documentation

6.10.3.1 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > NRmatrix< T > PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::compute_d_k0_Mat ( )

Here is the caller graph for this function:

6.10.3.2 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > template<std::size_t NN, std::size_t MM> const NRmatrix< T > PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::computeMatrix (const int *ref*, const int *a1*, const int *a2*, const int *a3*, const PRPSEvolution::Constants & *co*)

This method will compute all the possible permutations based on the given reference antenna

See also

ref

Parameters

in	*ref*	The reference antenna
in	*a1*	First antenna
in	*a2*	Second antenna
in	*a3*	Third antenna

6.10.3.3 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > int PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::computePermutations (const PRPSEvolution::Constants & *co*)

This method handles the computation of the antenna permutations

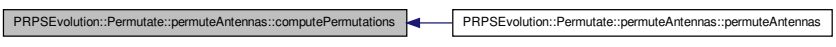
Parameters

in	*co*	Constant structure with the system constants we need

See also

PRPSEvolution::Constants

Here is the caller graph for this function:



6.10.3.4 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > void PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::dump_matrices_2_file ( )

This method will dump all the Antennas to an output file

Here is the caller graph for this function:

6.10.3.5 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> void PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::dumpConfigurationsToFile ( )

6.10.3.6 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T > int PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::rCoordFile ( )

Load the csv-file containing the coordinates and store it into the container.

Here is the caller graph for this function:

6.10.4 Member Data Documentation

6.10.4.1 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> Positioning::CoordContainer< N_ANTA, T > PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::AntennaCoordinates

6.10.4.2 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> std::array< AntennaPermutations< N_ANTPERM, Doub >, N_ANTA> PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::configurations

6.10.4.3 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> NRmatrix<T> PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::d_k0_mat

6.10.4.4 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> int PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::ref

6.10.4.5 template<std::size_t N_ANTA, std::size_t N_ANTPERM, typename T> PRPSEvolution::Constants PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >::systemConstants

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

- trunk/libPermutate/permutate.h

6.11 PRPSEvolution::Solve::PostProcessing Class Reference

`#include <postprocessing.h>`

Public Member Functions

- PostProcessing ()

6.11.1 Constructor & Destructor Documentation

6.11.1.1 PRPSEvolution::Solve::PostProcessing::PostProcessing ( ) [inline]

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

- trunk/libSolve/postprocessing.h

6.12 PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure > Class Template Reference

`#include <preprocessing.h>`

Public Member Functions

- PreProcessing (const std::array< AntennaPermutations< Permutate::MAX_PERMUTATION_AMOUNT, Doub >, N_ANTA > &, const NRmatrix< T > &, const int, const int)

Public Attributes

- std::vector< NRmatrix< T > > matrices
- std::vector< NRvector< T > > vectors
- std::vector< std::string > names
- int antennas

template<std::size_t N_ANTA, std::size_t N_Configs, typename T, typename T_Measure> class PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >

6.12.1 Constructor & Destructor Documentation

6.12.1.1 template<std::size_t N_ANTA, std::size_t N_Configs, typename T , typename T_Measure > PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >::PreProcessing (const std::array< AntennaPermutations< Permutate::MAX_PERMUTATION_AMOUNT, Doub >, N_ANTA > & *precalculatedMatrices,* const NRmatrix< T > & *d_k0s,* const int *finalAntAmount,* const int *offset*)

Construct the object an perform neccessary PreProcessing steps.

1. Read out the measurements from the given interface (e.g. a file)
2. Normalize everything
3. Select the matrices for further processing
4. Fill the matrices with the information
5. Precalculate the
$$c_{k0}$$
-Vector
6. Store matrices to make them availiable in the next steps

Parameters

in	*precalculatedMatrices*	Array containing the precalculated matricex from prior processing steps, This Array contains the static array for all possible permutations of the Antennas
in	*d_k0s*	This Array contains the d_{k0} , wich denotes the euklidean distances between the -Antennas
in	*finalAntAmount*	This field determines the Amount of Matrices we want to use for a calculation

6.12.2 Member Data Documentation

6.12.2.1 template<std::size_t N_ANTA, std::size_t N_Configs, typename T, typename T_Measure> int PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >::antennas

Amount of antennas for the solution

6.12.2.2 template<std::size_t N_ANTA, std::size_t N_Configs, typename T, typename T_Measure> std::vector<NRmatrix<T> > PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >::matrices

The precalcultated matrices for a solution

6.12.2.3 template<std::size_t N_ANTA, std::size_t N_Configs, typename T, typename T_Measure> std::vector< std::string > PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >::names

The "Names" of the matrices for a solution

6.12.2.4 template<std::size_t N_ANTA, std::size_t N_Configs, typename T, typename T_Measure> std::vector< NRvector< T > > PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_Measure >::vectors

The b-vectors for the solution

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

- trunk/libSolve/preprocessing.h

6.13 PRPSEvolution::Solve::ProblemDimensions Struct Reference

`#include <solve.h>`

Static Public Attributes

- static const int WholeTomato = 7
- static const int WholeTomatoMkI = 10
- static const int WholeTomatoMkI_A = 10
- static const int WholeTomatoMkI_B = 7
- static const int WholeTomatoMkII = 3
- static const int Sphere = 10
- static const int Rosenbrock = 15

6.13.1 Detailed Description

This gathers the problemdimensions of the defined fitness functions

6.13.2 Member Data Documentation

6.13.2.1 const int PRPSEvolution::Solve::ProblemDimensions::Rosenbrock = 15 [static]

6.13.2.2 const int PRPSEvolution::Solve::ProblemDimensions::Sphere = 10 [static]

6.13.2.3 const int PRPSEvolution::Solve::ProblemDimensions::WholeTomato = 7 [static]

6.13.2.4 const int PRPSEvolution::Solve::ProblemDimensions::WholeTomatoMkI = 10 [static]

6.13.2.5 const int PRPSEvolution::Solve::ProblemDimensions::WholeTomatoMkI_A = 10 [static]

6.13.2.6 const int PRPSEvolution::Solve::ProblemDimensions::WholeTomatoMkI_B = 7 [static]

6.13.2.7 const int PRPSEvolution::Solve::ProblemDimensions::WholeTomatoMkII = 3 [static]

The minimal dimension for this problem, depending on the amount of antennas used this number will increase

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

- trunk/libSolve/solve.h

6.14 PRPSEvolution::Solve::Process Class Reference

`#include <process.h>`

Public Member Functions

- Process ()
- Process (const Process &p)
- double getLastSolutionFitness ()
- template<typename T >
  T findSolutionSphere (Solve::ESStrategy strategy)
- template<typename T >
  T findSolutionCMA_ES_MkI ()
- template<typename T >
  T findSolutionCMA_ES_MkII ()

- template<typename T >
  T findSolutionSolveSingle (const NRmatrix< Doub > &A_selected, const NRvector< Doub > &b_selected, const std::vector< std::string > &names_selected, const int ants, const PRPSEvolution::Solve::ESStrategy strategy, const int seed)
- template<typename T >
  T findSolution (const std::vector< NRmatrix< Doub >> &A_selected, const std::vector< NRvector< Doub >> &b_selected, const std::vector< std::string > &names_selected, const int ants, const PRPSEvolution::Solve::ESStrategy strategy, const int seed)
- int sq (int i)
- void setMinSolutionFitness (double value)
- void setSeed (unsigned int value)
- void incrementFileCounter ()
- void resetFileCounter ()

Public Attributes

- int f_count = 0

6.14.1 Detailed Description

Find solutions for the possible matrices

6.14.2 Constructor & Destructor Documentation

6.14.2.1 PRPSEvolution::Solve::Process::Process ( ) [inline]

Constructor

6.14.2.2 PRPSEvolution::Solve::Process::Process (const Process & *p*) [inline]

6.14.3 Member Function Documentation

6.14.3.1 template<typename T > T PRPSEvolution::Solve::Process::findSolution (const std::vector< NRmatrix< Doub >> & *A_selected*, const std::vector< NRvector< Doub >> & *b_selected*, const std::vector< std::string > & *names_selected*, const int *ants*, const PRPSEvolution::Solve::ESStrategy *strategy*, const int *seed*) [inline]

Find a Solution for a given pair of matrices

Parameters

in	*A_selected*	The matrix A to use in this solution
in	*b_selected*	The c_k0' vector for this solution

Returns

The solution

Here is the caller graph for this function:

6.14.3.2 template<typename T > T PRPSEvolution::Solve::Process::findSolutionCMA_ES_MkI ( ) [inline]

Todo document

Returns

The solution

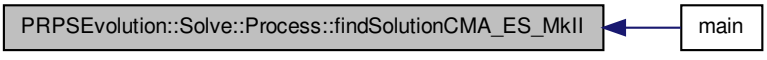
6.14.3.3 template<typename T > T PRPSEvolution::Solve::Process::findSolutionCMA_ES_MkII ( ) [inline]

Todo document

Returns

The solution

Here is the caller graph for this function:



6.14.3.4 template<typename T > T PRPSEvolution::Solve::Process::findSolutionSolveSingle (const NRmatrix< Doub > & *A_selected*, const NRvector< Doub > & *b_selected*, const std::vector< std::string > & *names_selected*, const int *ants*, const PRPSEvolution::Solve::ESStrategy *strategy*, const int *seed*) [inline]

Find a Solution for a given pair of matrices

Parameters

in	*A_selected*	The matrix A to use in this solution
in	*b_selected*	The c_k0' vector for this solution

Returns

The solution

6.14.3.5 template<typename T > T PRPSEvolution::Solve::Process::findSolutionSphere (Solve::ESStrategy *strategy*) [inline]

Set the ES-Strategy

Parameters

in	*Strategy*	The selected strategy

Todo document

Returns

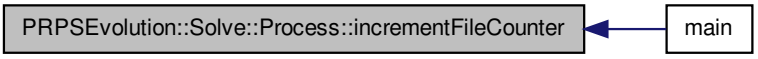
The solution

Here is the caller graph for this function:

6.14.3.6 double PRPSEvolution::Solve::Process::getLastSolutionFitness ( ) [inline]

6.14.3.7 void PRPSEvolution::Solve::Process::incrementFileCounter ( ) [inline]

Here is the caller graph for this function:



6.14.3.8 void PRPSEvolution::Solve::Process::resetFileCounter ( ) [inline]

Here is the caller graph for this function:

6.14.3.9 void PRPSEvolution::Solve::Process::setMinSolutionFitness (double *value*) [inline]

Sets the min. solution fitness we want to achieve.

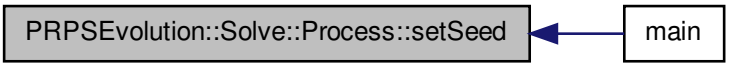
Parameters

in	*value*	The new value for the solution fitness

Here is the caller graph for this function:

6.14.3.10 void PRPSEvolution::Solve::Process::setSeed (unsigned int *value*) [inline]

Here is the caller graph for this function:



6.14.3.11 int PRPSEvolution::Solve::Process::sq (int *i*) [inline]

6.14.4 Member Data Documentation

6.14.4.1 int PRPSEvolution::Solve::Process::f_count = 0

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

- trunk/libSolve/process.h

6.15 PRPSEvolution::Solve::Process_MkII Class Reference

`#include <processMkII.h>`

Public Member Functions

- Process_MkII ()

- Process_MkII (NRmatrix< Doub > Mat, NRvector< Doub > Vect, std::string - Name)
- Process_MkII (std::vector< NRmatrix< Doub >> Mats, std::vector< NRvector< Doub >> Vects, std::vector< std::string > Names)
- Process_MkII (std::vector< NRmatrix< Doub >> Mats, std::vector< NRvector< Doub >> Vects, std::vector< std::string > Names, std::vector< std::vector< int >> IDs, double Epsilon)
- int WholeTomatoMkII (int dimension)
- int WholeTomatoMkI_A ()
- int WholeTomatoMkI_B ()
- int Process_MkII_test ()
- void setEpsilon (double Value)
- void setOutputFilePath (std::string file)
- void setOutputFilePathBase (std::string file)
- void setPrintLastOnly (void)
- void incrementFileCounter (void)
- void resetFileCounter ()
- void toggleVariant ()

6.15.1 Constructor & Destructor Documentation

6.15.1.1 PRPSEvolution::Solve::Process_MkII::Process_MkII ( ) [inline]

Here is the caller graph for this function:

6.15.1.2 PRPSEvolution::Solve::Process_MkII::Process_MkII (NRmatrix< Doub > *Mat*, NRvector< Doub > *Vect*, std::string *Name*) [inline]

Here is the call graph for this function:

6.15.1.3 PRPSEvolution::Solve::Process_MkII::Process_MkII (std::vector< NRmatrix< Doub >> *Mats*, std::vector< NRvector< Doub >> *Vects*, std::vector< std::string > *Names*) [inline]

Here is the call graph for this function:

6.15.1.4 PRPSEvolution::Solve::Process_MkII::Process_MkII (std::vector< NRmatrix< Doub >> *Mats*, std::vector< NRvector< Doub >> *Vects*, std::vector< std::string > *Names*, std::vector< std::vector< int >> *IDs*, double *Epsilon*) [inline]

Here is the call graph for this function:

6.15.2 Member Function Documentation

6.15.2.1 void PRPSEvolution::Solve::Process_MkII::incrementFileCounter (void) [inline]

increment the File counter

Here is the caller graph for this function:

6.15.2.2 int PRPSEvolution::Solve::Process_MkII::Process_MkII_test ( ) [inline]

Here is the caller graph for this function:

6.15.2.3 void PRPSEvolution::Solve::Process_MkII::resetFileCounter ( ) [inline]

6.15.2.4 void PRPSEvolution::Solve::Process_MkII::setEpsilon (double *Value*) [inline]

6.15.2.5 void PRPSEvolution::Solve::Process_MkII::setOutputFilePath (std::string *file*) [inline]

6.15.2.6 void PRPSEvolution::Solve::Process_MkII::setOutputFilePathBase (std::string *file*) [inline]

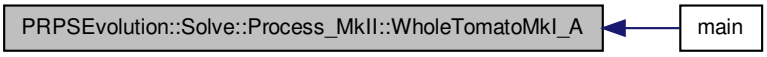
Here is the caller graph for this function:

6.15.2.7 void PRPSEvolution::Solve::Process_MkII::setPrintLastOnly (void) [inline]

6.15.2.8 void PRPSEvolution::Solve::Process_MkII::toggleVariant ( ) [inline]

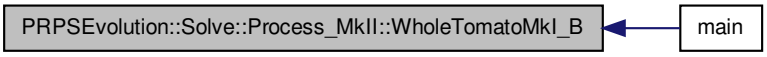
6.15.2.9 int PRPSEvolution::Solve::Process_MkII::WholeTomatoMkI_A ( ) [inline]

Here is the caller graph for this function:



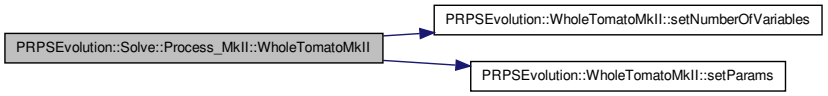
6.15.2.10 int PRPSEvolution::Solve::Process_MkII::WholeTomatoMkI_B ( ) [inline]

Here is the caller graph for this function:



6.15.2.11 int PRPSEvolution::Solve::Process_MkII::WholeTomatoMkII (int *dimension*) [inline]

Here is the call graph for this function:



Here is the caller graph for this function:

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

- trunk/libSolve/processMkII.h

6.16 PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return > Struct Template Reference

`#include <solveresult.h>`

Public Attributes

- T_Store1 valCont
- T_Store2 valDis
- T_Return fitness
- int iterations
- int duration
- bool converged

6.16.1 Detailed Description

template<typename T_Store1, typename T_Store2, typename T_Return>struct PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >

Stores the final state of a solution

6.16.2 Member Data Documentation

6.16.2.1 template<typename T_Store1, typename T_Store2, typename T_Return> bool PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::converged

Indicates whether the build in convergence criterium was applied, or not

6.16.2.2 template<typename T_Store1, typename T_Store2, typename T_Return> int PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::duration

The processing time for this solution

6.16.2.3 template<typename T_Store1, typename T_Store2, typename T_Return> T_Return PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::fitness

Whrere the result is stored The fitness value

6.16.2.4 template<typename T_Store1, typename T_Store2, typename T_Return> int PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::iterations

The amount of iterations needed for this result

6.16.2.5 template<typename T_Store1, typename T_Store2, typename T_Return> T_Store1 PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::valCont

6.16.2.6 template<typename T_Store1, typename T_Store2, typename T_Return> T_Store2 PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >::valDis

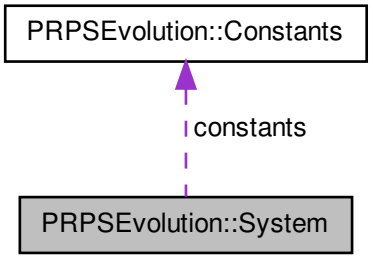
The documentation for this struct was generated from the following file:

- trunk/libSolve/solveresult.h

6.17 PRPSEvolution::System Struct Reference

`#include <prpsevolutionsystem.h>`

Collaboration diagram for PRPSEvolution::System:



Public Member Functions

- System ()
- System (const PRPSEvolution::System &s)
- int rPRPSIniFile ()

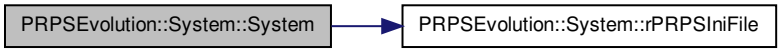
Public Attributes

- PRPSEvolution::Constants constants
- std::string fn

6.17.1 Constructor & Destructor Documentation

6.17.1.1 PRPSEvolution::System::System ( ) [inline]

Here is the call graph for this function:



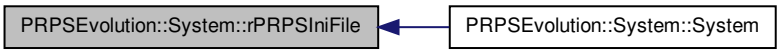
6.17.1.2 PRPSEvolution::System::System (const PRPSEvolution::System & *s*) [inline]

copy constructor

6.17.2 Member Function Documentation

6.17.2.1 int PRPSEvolution::System::rPRPSIniFile ( ) [inline]

Here is the caller graph for this function:



6.17.3 Member Data Documentation

6.17.3.1 PRPSEvolution::Constants PRPSEvolution::System::constants

6.17.3.2 std::string PRPSEvolution::System::fn

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

- trunk/libPRPSSystem/prpsevolutionsystem.h

6.18 PRPSEvolution::Solve::Ueber9000< T > Struct Template Reference

`#include <ueber9000.h>`

Public Member Functions

- Ueber9000 ()
- Ueber9000 (int i)
- Ueber9000 (const Ueber9000 &me)
- Ueber9000 (const NRmatrix< T > A_selected, const NRvector< T > b_selected)
- Ueber9000 (const std::vector< NRmatrix< T >> As, const std::vector< NRvector< T >> bs, const std::vector< std::string > namess, const int numOAnts, const int select)

- std::vector< std::vector< int > > parseIdxFromNames (const std::vector< std::string > &namess)
- double WholeTomato (const ChromosomeT< double > &x)
- double WholeTomatoMkII (const ChromosomeT< double > &x)
- double WholeTomatoMkII (const ChromosomeT< double > &x1, const ChromosomeT< double > &x2)
- double WholeTomatoMkII (const ChromosomeT< double > &x, const ChromosomeT< int > &n)
- double WholeTomato (const NRmatrix< T > &A, const ChromosomeT< double > &x, const NRvector< T > &b)
- double WholeTomatoMkI (const NRmatrix< T > &A, const ChromosomeT< double > &x, const NRvector< T > &b)
- double WholeTomatoMkII (const NRmatrix< T > &A, const ChromosomeT< double > &x, const NRvector< T > &b)
- double SuWi_WavenumberVariation (const ChromosomeT< double > &n)
- double SuWi_PositionVariation (const ChromosomeT< double > &pos)
- double fitnessSphere (const ChromosomeT< double > &c)
- double fitnessSphereMkII (const ChromosomeT< double > &c1, const ChromosomeT< double > &c2)
- double fitnessRosenbrock (const ChromosomeT< double > &c)
- double fitnessAckley (const std::vector< double > &x)

Public Attributes

- double(Ueber9000< double >::* evaluate)(const ChromosomeT< double > &)
- double(Ueber9000< double >::* evaluateMkI)(const ChromosomeT< double > &)
- double(Ueber9000< double >::* evaluateMkII)(const ChromosomeT< double > &, const ChromosomeT< double > &)
- double(Ueber9000< double >::* evaluateMkIII)(const ChromosomeT< double > &, const ChromosomeT< int > &)
- int Dimension
- std::vector< NRmatrix< T > > A
- std::vector< NRvector< T > > b
- std::vector< std::string > names
- std::vector< std::vector< int > > idxs
- int evaluations = 0

6.18.1 Detailed Description

template<typename T>struct PRPSEvolution::Solve::Ueber9000< T >

Collect the fitness functions. Make sure they are static so we can function-pointer to them.

6.18.2 Constructor & Destructor Documentation

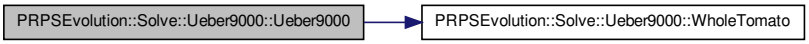
6.18.2.1 template<typename T> PRPSEvolution::Solve::Ueber9000< T >::Ueber9000 ( ) [inline]

Default constructor

6.18.2.2 template<typename T> PRPSEvolution::Solve::Ueber9000< T >::Ueber9000 (int *i*) [inline]

6.18.2.3 template<typename T> PRPSEvolution::Solve::Ueber9000< T >::Ueber9000 (const Ueber9000< T > & *me*) [inline]

Here is the call graph for this function:



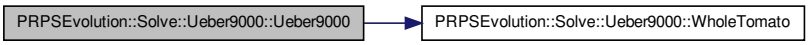
6.18.2.4 template<typename T> PRPSEvolution::Solve::Ueber9000< T >::Ueber9000 (const NRmatrix< T > *A_selected*, const NRvector< T > *b_selected*) [inline]

Construct Ueber9000 to use the WholeTomato as fitness function

Parameters

in	*A_selected*	The matrix A for this Solution
in	*c_k0_-selected*	The vector b for this Solution

Here is the call graph for this function:



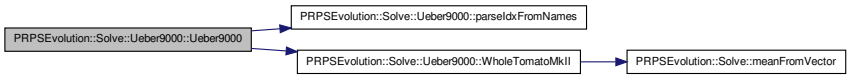
6.18.2.5 template<typename T> PRPSEvolution::Solve::Ueber9000< T >::Ueber9000 (const std::vector< NRmatrix< T >> *As*, const std::vector< NRvector< T >> *bs*, const std::vector< std::string > *namess*, const int *numOAnts*, const int *select*) [inline]

Construct Ueber9000 to use the WholeTomato as fitness function

Parameters

in	*As*	The matrices A to get a solution from
in	*bs*	The vectors b
in	*namess*	The Names of the matrices in As
in	*numOAnts*	The number of antennas used in the matrices in As
in	*select*	Selects the WholeTomato-Version

Here is the call graph for this function:



6.18.3 Member Function Documentation

6.18.3.1 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::fitnessAckley (const std::vector< double > & *x*) [inline]

The infamous Ackley-function

6.18.3.2 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::fitnessRosenbrock (const ChromosomeT< double > & *c*) [inline]

The Rosenbrock implementation

6.18.3.3 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::fitnessSphere (const ChromosomeT< double > & *c*) [inline]

This ist the fitness function used in the EA algorithm

6.18.3.4 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::fitnessSphereMkII (const ChromosomeT< double > & *c1*, const ChromosomeT< double > & *c2*) [inline]

This ist the fitness function used in the EA algorithm. This implementation uses two input vectors of the same datatype for test purpose of multi chromosome optimization

6.18.3.5 template<typename T> std::vector<std::vector<int> > PRPSEvolution::Solve::Ueber9000< T >::parseIdxFromNames (const std::vector< std::string > & *namess*) [inline]

This function will parse the indeces used for a solution

Parameters

in	*namess*	Contains the "Name" of each matrix we want to use in this solution

Returns

A two dimensional vector with the indeces of each antenna for each matrix

Here is the caller graph for this function:

6.18.3.6 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::SuWi_PositionVariation (const ChromosomeT< double > & *pos*) [inline]

Approach 3 based on the thoughts of by S. Winter

6.18.3.7 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::SuWi_WavenumberVariation (const ChromosomeT< double > & *n*) [inline]

Approach 2 based on the thoughts of S. Winter. Here we want to optimize the wavenumbers

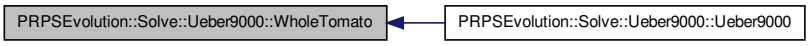
6.18.3.8 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomato (const ChromosomeT< double > & *x*) [inline]

This method basically wraps around the real WholeTomato-function. Maps the function so that it can be used with the evaluate-method

Parameters

in	*x*	The vector x

Here is the caller graph for this function:



6.18.3.9 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomato (const NRmatrix< T > & *A*, const ChromosomeT< double > & *x*, const NRvector< T > & *b*) [inline]

This approach will solve the scene defined by the 10x3 matrix The approach is described in the Master-Thesis of C.Gnip Basically solves the linear equation

$$r = \mathbf{A}\mathbf{x} - \mathbf{b}$$

Parameters

in	*A*	The 10x3 Matrix that ist used in this solution
in	*x*	The vector containing the variables
in	*b*	Representing the vector b

Returns

The residuum of the equation system representing the "Fitness" of the given - Solution in

See also

x

6.18.3.10 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkl (const NRmatrix< T > & *A*, const ChromosomeT< double > & *x*, const NRvector< T > & *b*) [inline]

Todo documentation

Parameters

in	*A*	The 10x3 Matrix that ist used in this solution
in	*x*	The vector containing the variables
in	*b*	Representing the vector b

Returns

The residuum of the equation system representing the "Fitness" of the given - Solution in

See also

x

6.18.3.11 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > & *x*) [inline]

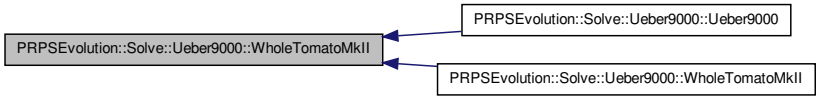
Todo document

Parameters

in	*x*	The vector x containing the

Here is the call graph for this function:

Here is the caller graph for this function:



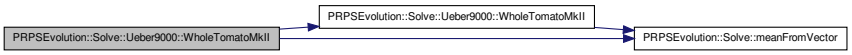
6.18.3.12 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > & *x1*, const ChromosomeT< double > & *x2*) [inline]

Todo document

Parameters

in	*x*	The vector x containing the

Here is the call graph for this function:



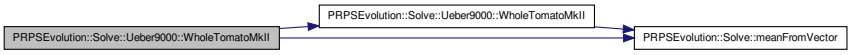
6.18.3.13 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const ChromosomeT< double > & *x*, const ChromosomeT< int > & *n*) [inline]

Todo document

Parameters

in	*x*	The vector x containing the

Here is the call graph for this function:



6.18.3.14 template<typename T> double PRPSEvolution::Solve::Ueber9000< T >::WholeTomatoMkII (const NRmatrix< T > & *A*, const ChromosomeT< double > & *x*, const NRvector< T > & *b*) [inline]

This function contains the implementation of the whole model. This approach will solve calculate the 10x3 matrix described in the Master-Thesis of C.Gnip Basically solves the linear equation

$$r = \mathbf{Ax} - \mathbf{b}$$

Parameters

in	*A*	The 10x3 Matrix that ist used in this solution
in	*x*	The vector containing the variables
in	*b*	Representing the vector b

Returns

The residuum of the equation system representing the "Fitness" of the given - Solution in

See also

x

6.18.4 Member Data Documentation

6.18.4.1 template<typename T> std::vector<NRmatrix< T > > PRPSEvolution::Solve::Ueber9000< T >::A

The Matrices we need to solve the Problem

6.18.4.2 template<typename T> std::vector<NRvector< T > > PRPSEvolution::Solve::Ueber9000< T >::b

The b-vector needed to find a Solution

6.18.4.3 template<typename T> int PRPSEvolution::Solve::Ueber9000< T >::Dimension

The Dimension of the Problem

6.18.4.4 template<typename T> double(Ueber9000<double>::* PRPSEvolution::Solve::Ueber9000< T >::evaluate)(const ChromosomeT< double > &)

Todo document

6.18.4.5 template<typename T> double(Ueber9000<double>::* PRPSEvolution::Solve::Ueber9000< T >::evaluateMkI)(const ChromosomeT< double > &)

Todo document

6.18.4.6 template<typename T> double(Ueber9000<double>::* PRPSEvolution::Solve::Ueber9000< T >::evaluateMkII)(const ChromosomeT< double > &, const ChromosomeT< double > &)

Todo document

6.18.4.7 template<typename T> double(Ueber9000<double>::* PRPSEvolution::Solve::Ueber9000< T >::evaluateMkIII)(const ChromosomeT< double > &, const ChromosomeT< int > &)

Todo document

6.18.4.8 template<typename T> int PRPSEvolution::Solve::Ueber9000< T >::evaluations = 0

6.18.4.9 template<typename T> std::vector<std::vector<int> > PRPSEvolution::Solve::Ueber9000< T >::idxs

6.18.4.10 template<typename T> std::vector<std::string> PRPSEvolution::Solve::Ueber9000< T >::names

The names for the Solution (contains the contributing antennas)

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

- trunk/libSolve/ueber9000.h

6.19 PRPSEvolution::WholeTomatoMkI_A Struct Reference

`#include <WholeTomatoMkI_A.h>`

Public Member Functions

- WholeTomatoMkI_A (unsigned int numberOfVariables=7)
- std::string name () const

  From INameable: return the class name.
- std::size_t numberOfVariables () const
- bool hasScalableDimensionality () const
- void setNumberOfVariables (std::size_t numberOfVariables)
- void configure (const PropertyTree &node)
- void proposeStartingPoint (SearchPointType &x) const
- double eval (const SearchPointType &x) const
- void setParams (const NRmatrix< Doub > &M, const NRvector< Doub > &v)
- void setMat (const NRmatrix< Doub > &M)
- void setVec (const NRvector< Doub > &v)
- double mkI (const NRmatrix< Doub > &A, const SearchPointType &x, const NRvector< Doub > &b) const

6.19.1 Constructor & Destructor Documentation

6.19.1.1 PRPSEvolution::WholeTomatoMkl_A::WholeTomatoMkl_A (unsigned int *numberOfVariables* = 7) [inline]

6.19.2 Member Function Documentation

6.19.2.1 void PRPSEvolution::WholeTomatoMkl_A::configure (const PropertyTree & *node*) [inline]

6.19.2.2 double PRPSEvolution::WholeTomatoMkl_A::eval (const SearchPointType & *x*) const [inline]

6.19.2.3 bool PRPSEvolution::WholeTomatoMkl_A::hasScalableDimensionality ( ) const [inline]

6.19.2.4 double PRPSEvolution::WholeTomatoMkl_A::mkl (const NRmatrix< Doub > & *A*, const SearchPointType & *x*, const NRvector< Doub > & *b*) const [inline]

Todo documentation

Parameters

in	*A*	The 10x3 Matrix that ist used in this solution
in	*x*	The vector containing the variables
in	*b*	Representing the vector b

Returns

The residuum of the equation system representing the "Fitness" of the given - Solution in

See also

x

6.19.2.5 std::string PRPSEvolution::WholeTomatoMkl_A::name ( ) const [inline]

From INameable: return the class name.

6.19.2.6 std::size_t PRPSEvolution::WholeTomatoMkl_A::numberOfVariables ( ) const [inline]

6.19.2.7 void PRPSEvolution::WholeTomatoMkl_A::proposeStartingPoint (SearchPointType & *x*) const [inline]

6.19.2.8 void PRPSEvolution::WholeTomatoMkI_A::setMat (const NRmatrix< Doub > & *M*) [inline]

6.19.2.9 void PRPSEvolution::WholeTomatoMkI_A::setNumberOfVariables (std::size_t *numberOfVariables*) [inline]

6.19.2.10 void PRPSEvolution::WholeTomatoMkI_A::setParams (const NRmatrix< Doub > & *M*, const NRvector< Doub > & *v*) [inline]

6.19.2.11 void PRPSEvolution::WholeTomatoMkI_A::setVec (const NRvector< Doub > & *v*) [inline]

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

- trunk/libSolve/Objectivefunctions/WholeTomatoMkI_A.h

6.20 PRPSEvolution::WholeTomatoMkI_B Struct Reference

`#include <WholeTomatoMkI_B.h>`

Public Member Functions

- WholeTomatoMkI_B (unsigned int numberOfVariables=7)
- std::string name () const

  From INameable: return the class name.
- std::size_t numberOfVariables () const
- bool hasScalableDimensionality () const
- void setNumberOfVariables (std::size_t numberOfVariables)
- void configure (const PropertyTree &node)
- void proposeStartingPoint (SearchPointType &x) const
- double eval (const SearchPointType &x) const
- void setParams (const NRmatrix< Doub > &M, const NRvector< Doub > &v)
- void setMat (const NRmatrix< Doub > &M)
- void setVec (const NRvector< Doub > &v)
- double mkI (const NRmatrix< Doub > &A, const SearchPointType &x, const NRvector< Doub > &b) const

6.20.1 Constructor & Destructor Documentation

6.20.1.1 PRPSEvolution::WholeTomatoMkI_B::WholeTomatoMkI_B (unsigned int *numberOfVariables* = 7) [inline]

6.20.2 Member Function Documentation

6.20.2.1 void PRPSEvolution::WholeTomatoMkl_B::configure (const PropertyTree & *node*) [inline]

6.20.2.2 double PRPSEvolution::WholeTomatoMkl_B::eval (const SearchPointType & *x*) const [inline]

6.20.2.3 bool PRPSEvolution::WholeTomatoMkl_B::hasScalableDimensionality ( ) const [inline]

6.20.2.4 double PRPSEvolution::WholeTomatoMkl_B::mkl (const NRmatrix< Doub > & *A*, const SearchPointType & *x*, const NRvector< Doub > & *b*) const [inline]

Todo documentation

Parameters

in	*A*	The 10x3 Matrix that ist used in this solution
in	*x*	The vector containing the variables
in	*b*	Representing the vector b

Returns

The residuum of the equation system representing the "Fitness" of the given - Solution in

See also

x

6.20.2.5 std::string PRPSEvolution::WholeTomatoMkl_B::name ( ) const [inline]

From INameable: return the class name.

6.20.2.6 std::size_t PRPSEvolution::WholeTomatoMkl_B::numberOfVariables ( ) const [inline]

6.20.2.7 void PRPSEvolution::WholeTomatoMkl_B::proposeStartingPoint (SearchPointType & *x*) const [inline]

6.20.2.8 void PRPSEvolution::WholeTomatoMkl_B::setMat (const NRmatrix< Doub > & *M*) [inline]

6.20.2.9 void PRPSEvolution::WholeTomatoMkl_B::setNumberOfVariables (std::size_t *numberOfVariables*) [inline]

6.20.2.10 void PRPSEvolution::WholeTomatoMkI_B::setParams (const NRmatrix< Doub > & *M*, const NRvector< Doub > & *v*) [inline]

6.20.2.11 void PRPSEvolution::WholeTomatoMkI_B::setVec (const NRvector< Doub > & *v*) [inline]

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

- trunk/libSolve/Objectivefunctions/WholeTomatoMkI_B.h

6.21 PRPSEvolution::WholeTomatoMkII Struct Reference

`#include <WholeTomatoMkII.h>`

Public Types

- typedef AbstractOptimizer < shark::VectorSpace< double > , double, SingleObjectiveResultSet < typename shark::VectorSpace < double >::PointType > > base_type
- typedef base_type::ObjectiveFunctionType ObjectiveFunctionType

Public Member Functions

- WholeTomatoMkII (unsigned int numberOfVariables=5)
- std::string name () const

  From INameable: return the class name.
- std::size_t numberOfVariables () const
- bool hasScalableDimensionality () const
- void setNumberOfVariables (std::size_t numberOfVariables)
- void configure (const PropertyTree &node)
- void proposeStartingPoint (SearchPointType &x) const
- double eval (const SearchPointType &p) const
- void setParams (const std::vector< NRmatrix< Doub >> &M, const std::vector< NRvector< Doub >> &v, const std::vector< std::string > &n)
- void setParams (const std::vector< NRmatrix< Doub >> &M, const std::vector< NRvector< Doub >> &v, const std::vector< std::vector< int >> &i)
- void setMats (const std::vector< NRmatrix< Doub >> &M)
- void setVecs (const std::vector< NRvector< Doub >> &v)
- void setNames (const std::vector< std::string > &n)
- void setIdx (const std::vector< std::vector< int >> &i)
- double mkII (const NRmatrix< Doub > &A, const double *x, const NRvector< Doub > &b) const

6.21.1 Member Typedef Documentation

6.21.1.1 typedef AbstractOptimizer<shark::VectorSpace< double >,double,SingleObjectiveResultSet<typename shark::VectorSpace< double >::PointType> > PRPSEvolution::WholeTomatoMkII::base_type

6.21.1.2 typedef base_type::ObjectiveFunctionType PRPSEvolution::WholeTomatoMkII::ObjectiveFunctionType

6.21.2 Constructor & Destructor Documentation

6.21.2.1 PRPSEvolution::WholeTomatoMkII::WholeTomatoMkII (unsigned int *numberOfVariables* = 5) [inline]

6.21.3 Member Function Documentation

6.21.3.1 void PRPSEvolution::WholeTomatoMkII::configure (const PropertyTree & *node*) [inline]

6.21.3.2 double PRPSEvolution::WholeTomatoMkII::eval (const SearchPointType & *p*) const [inline]

Here is the call graph for this function:

6.21.3.3 bool PRPSEvolution::WholeTomatoMkII::hasScalableDimensionality ( ) const [inline]

6.21.3.4 double PRPSEvolution::WholeTomatoMkII::mkII (const NRmatrix< Doub > & *A*, const double * *x*, const NRvector< Doub > & *b*) const [inline]

6.21.3.5 std::string PRPSEvolution::WholeTomatoMkII::name ( ) const [inline]

From INameable: return the class name.

6.21.3.6 std::size_t PRPSEvolution::WholeTomatoMkII::numberOfVariables ( ) const [inline]

6.21.3.7 void PRPSEvolution::WholeTomatoMkII::proposeStartingPoint (SearchPointType & *x*) const [inline]

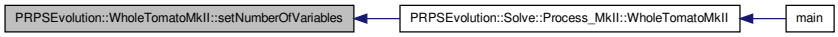
6.21.3.8 void PRPSEvolution::WholeTomatoMkII::setIdx (const std::vector< std::vector< int >> & *i*) [inline]

6.21.3.9 void PRPSEvolution::WholeTomatoMkII::setMats (const std::vector< NRmatrix< Doub >> & *M*) [inline]

6.21.3.10 void PRPSEvolution::WholeTomatoMkII::setNames (const std::vector< std::string > & *n*) [inline]

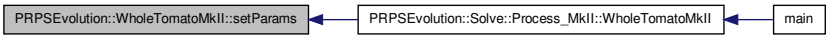
6.21.3.11 void PRPSEvolution::WholeTomatoMkII::setNumberOfVariables (std::size_t *numberOfVariables*) [inline]

Here is the caller graph for this function:



6.21.3.12 void PRPSEvolution::WholeTomatoMkII::setParams (const std::vector< NRmatrix< Doub >> & *M*, const std::vector< NRvector< Doub >> & *v*, const std::vector< std::string > & *n*) [inline]

Here is the caller graph for this function:



6.21.3.13 void PRPSEvolution::WholeTomatoMkII::setParams (const std::vector< NRmatrix< Doub >> & *M*, const std::vector< NRvector< Doub >> & *v*, const std::vector< std::vector< int >> & *i*) [inline]

6.21.3.14 void PRPSEvolution::WholeTomatoMkII::setVecs (const std::vector< NRvector< Doub >> & *v*) [inline]

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

- trunk/libSolve/Objectivefunctions/WholeTomatoMkII.h

Chapter 7

File Documentation

7.1 trunk/CMakeFiles/CompilerIdC/CMakeCCompilerId.c File - Reference

Defines

- #define COMPILER_ID ""
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""

Functions

- int main (int argc, char *argv[])

Variables

- char const * info_compiler = "]"
- char const * info_platform = "]"
- char const * info_arch = "]"

7.1.1 Define Documentation

7.1.1.1 #define ARCHITECTURE_ID ""

7.1.1.2 #define COMPILER_ID ""

7.1.1.3 #define PLATFORM_ID ""

7.1.2 Function Documentation

7.1.2.1 int main (int *argc,* char * *argv[]*)

7.1.3 Variable Documentation

7.1.3.1 char const* info_arch = "]"

7.1.3.2 char const* info_compiler = "]"

7.1.3.3 char const* info_platform = "]"

7.2 trunk/CMakeFiles/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Defines

- #define COMPILER_ID ""
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""

Functions

- int main (int argc, char *argv[])

Variables

- char const * info_compiler = "]"
- char const * info_platform = "]"
- char const * info_arch = "]"

7.2.1 Define Documentation

7.2.1.1 #define ARCHITECTURE_ID ""

7.2.1.2 #define COMPILER_ID ""

7.2.1.3 #define PLATFORM_ID ""

7.2.2 Function Documentation

7.2.2.1 int main (int *argc,* char * *argv[]*)

7.2.3 Variable Documentation

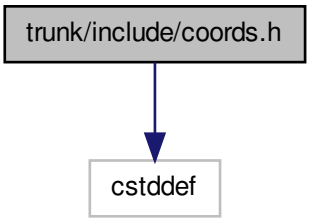
7.2.3.1 char const* info_arch = "]"

7.2.3.2 char const* info_compiler = "]"

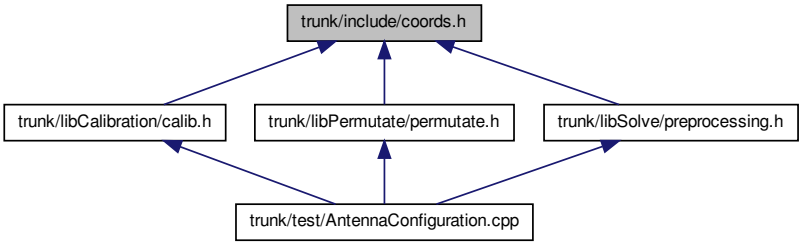
7.2.3.3 char const* info_platform = "]"

7.3 trunk/include/coords.h File Reference

`#include <cstddef>` Include dependency graph for coords.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Positioning::CoordContainer< N, T >

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Positioning

7.4 trunk/include/prps.h File Reference

Variables

- const int ANTENNA_AMOUNT = 8
- const int EXPECTED_LINES = 10
- const int EXPECTED_VALUES = 10

7.4.1 Variable Documentation

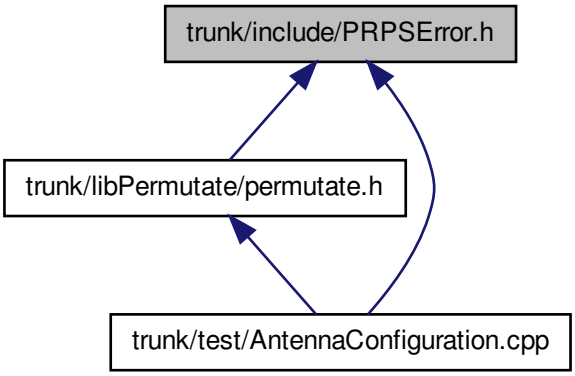
7.4.1.1 const int ANTENNA_AMOUNT = 8

7.4.1.2 const int EXPECTED_LINES = 10

7.4.1.3 const int EXPECTED_VALUES = 10

7.5 trunk/include/PRPSError.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace PRPSError
- namespace PRPSError::FileIO

Variables

- const int PRPSError::FileIO::okay = 0
- const int PRPSError::FileIO::generalError = -1
- const int PRPSError::FileIO::fnf = -2
- const int PRPSError::FileIO::inputmalformed = -3
- const int PRPSError::okay = 0
- const int PRPSError::general = -1
- const int PRPSError::critical = 10

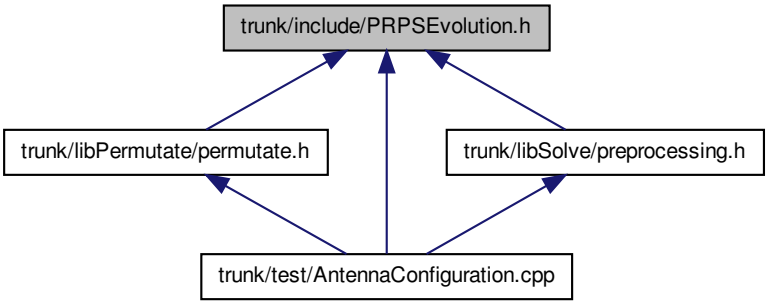
7.5.1 Detailed Description

Date

2013|Jun|18 This file cantains definitions belonging to the PRPSError-namespace. It is split into sub-namespaces for keeping thing nicely small.

7.6 trunk/include/PRPSEvolution.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace PRPSEvolution

Variables

- const int PRPSEvolution::ANTENNA_AMOUNT = 8
- const int PRPSEvolution::EXPECTED_LINES_CALIBRATION_FILE = 4

- const int PRPSEvolution::EXPECTED_VALUES_CALIBRATION_FILE = ANTENNA_AMOUNT
- const int PRPSEvolution::EXPECTED_LINES_COORD_FILE = ANTENNA_AMOUNT
- const int PRPSEvolution::EXPECTED_VALUES_COORD_FILE = 3
- const int PRPSEvolution::EXPECTED_LINES_SYSTEM_INI_FILE = 2
- const int PRPSEvolution::MAT_ROWS = 3
- const int PRPSEvolution::MAT_COLS = 10
- const int PRPSEvolution::CALIBRATION_POINTS_AVAILABLE = 4
- const int PRPSEvolution::EXPECTED_LINES_MEASUREMENT_FILE = ANTENNA_AMOUNT
- const int PRPSEvolution::EXPECTED_VALUES_MEASUREMENT_FILE = 2
- const int PRPSEvolution::DATA_NV = 65535
- const int PRPSEvolution::DEFAULT_MIN_GROUP_SIZE = 4

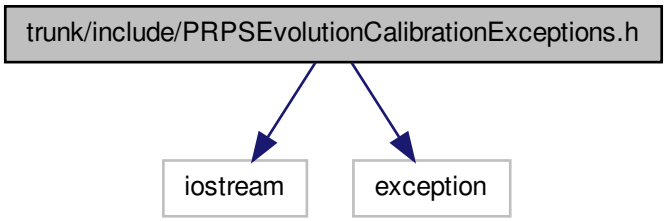
7.6.1 Detailed Description

Date

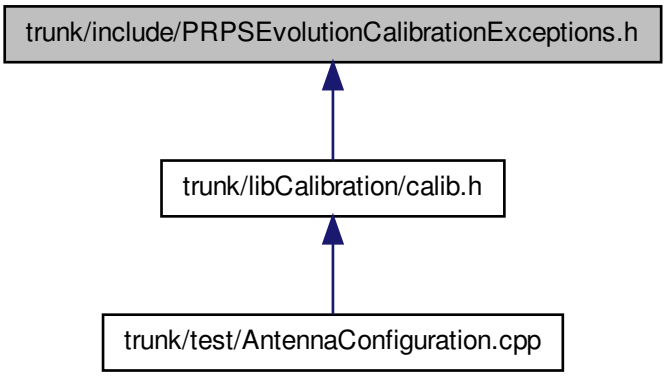
2013|Jun|18 This file collects definitions belonging to the PRPSEvolution-namespace. Especially const. defines.

7.7 trunk/include/PRPSEvolutionCalibrationExceptions.h File - Reference

`#include <iostream>` `#include <exception>` Include dependency graph for PRPSEvolutionCalibrationExceptions.h:



This graph shows which files directly or indirectly include this file:

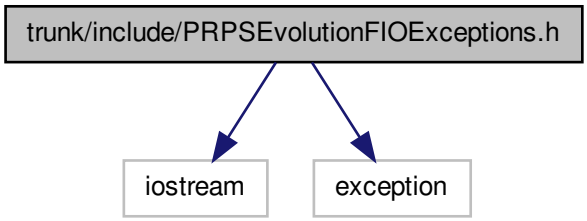


Namespaces

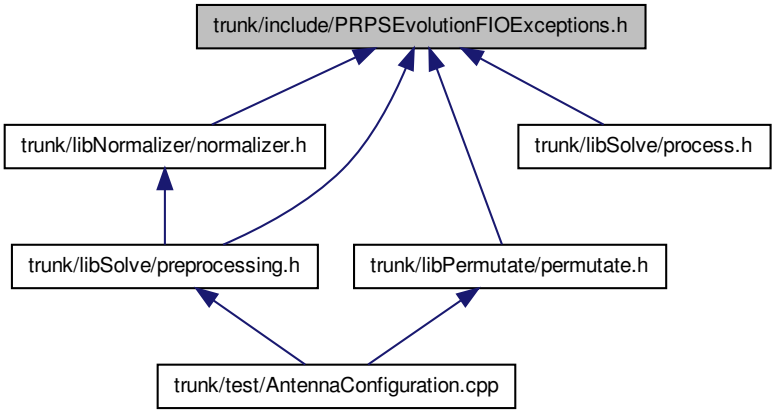
- namespace PRPSEvolution
- namespace PRPSEvolution::Exceptions
- namespace PRPSEvolution::Exceptions::Calibration

7.8 trunk/include/PRPSEvolutionFIOExceptions.h File Reference

`#include <iostream>` `#include <exception>` Include dependency graph for PRPSEvolutionFIOExceptions.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Exceptions::FileIO::FileNotFound
- struct PRPSEvolution::Exceptions::FileIO::MalformedInput
- struct PRPSEvolution::Exceptions::FileIO::OutputFailure

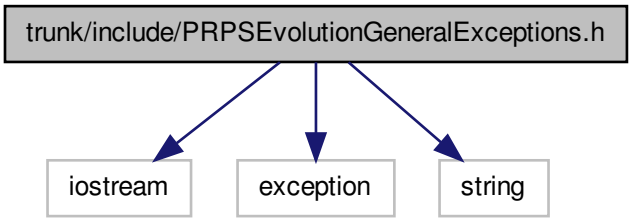
Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Exceptions
- namespace PRPSEvolution::Exceptions::FileIO

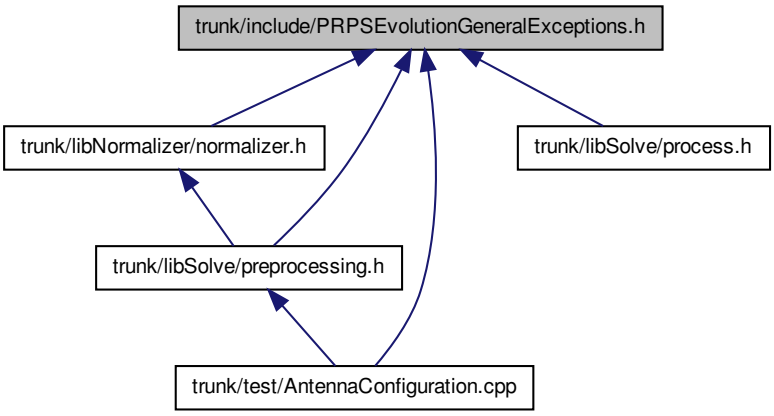
7.9 trunk/include/PRPSEvolutionGeneralExceptions.h File Reference

```
#include <iostream> #include <exception> #include <string> ×
```

Include dependency graph for PRPSEvolutionGeneralExceptions.h:



This graph shows which files directly or indirectly include this file:



Classes

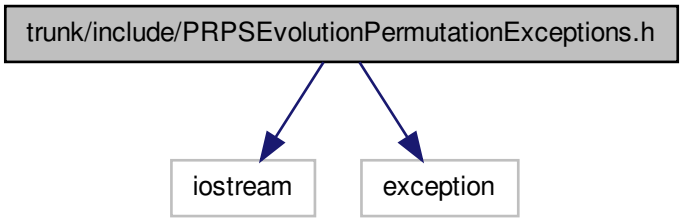
- struct PRPSEvolution::Exceptions::General::NotImplemented

Namespaces

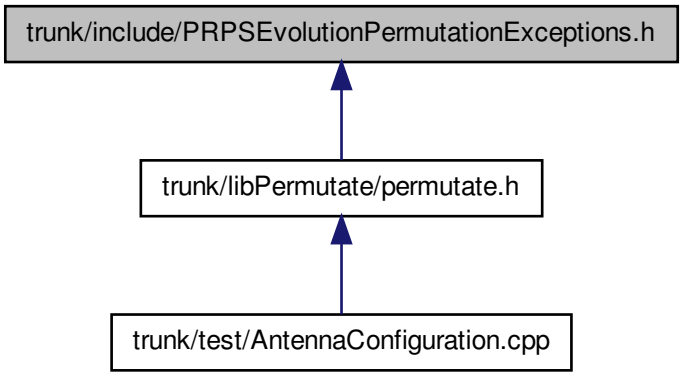
- namespace PRPSEvolution
- namespace PRPSEvolution::Exceptions
- namespace PRPSEvolution::Exceptions::General

7.10 trunk/include/PRPSEvolutionPermutationExceptions.h File - Reference

`#include <iostream>` `#include <exception>` Include dependency graph for PRPSEvolutionPermutationExceptions.h:



This graph shows which files directly or indirectly include this file:



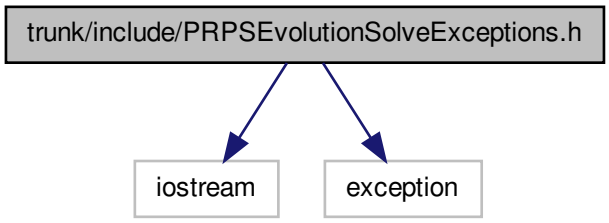
Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Exceptions

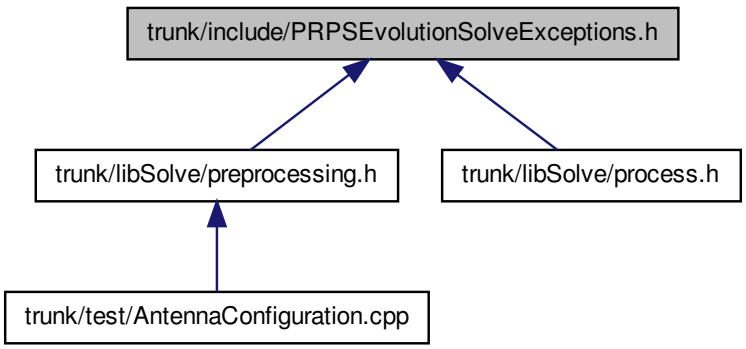
- namespace PRPSEvolution::Exceptions::Permutation

7.11 trunk/include/PRPSEvolutionSolveExceptions.h File Reference

`#include <iostream>` `#include <exception>` Include dependency graph for PRPSEvolutionSolveExceptions.h:



This graph shows which files directly or indirectly include this file:



Namespaces

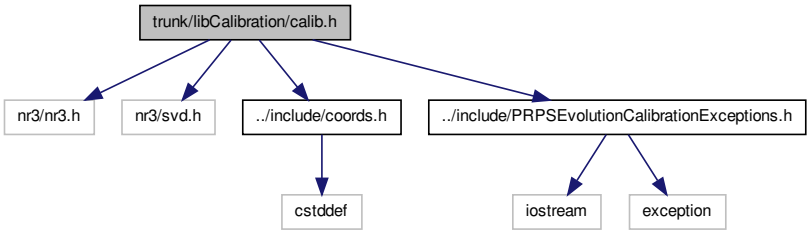
- namespace PRPSEvolution

- namespace PRPSEvolution::Exceptions
- namespace PRPSEvolution::Exceptions::Solve

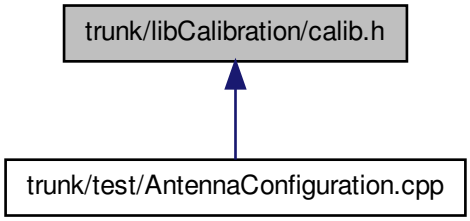
7.12 trunk/libCalibration/calib.cpp File Reference

7.13 trunk/libCalibration/calib.h File Reference

`#include <nr3/nr3.h> #include <nr3/svd.h> #include "../include/coords.h" #include "../include/PRPSEvolutionCalibrationExceptions.h"` Include dependency graph for calib.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Calibration::performCalibration< N_ANTA, N_CALPOS, T >

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Calibration

7.13.1 Detailed Description

Date

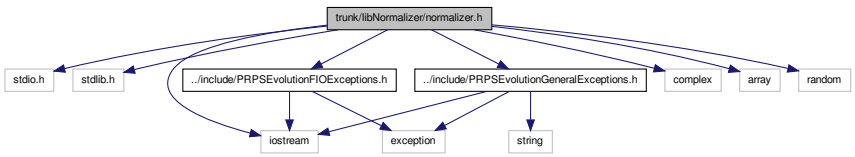
2013|Jun|25

7.14 trunk/libNormalizer/normalizer.cpp File Reference

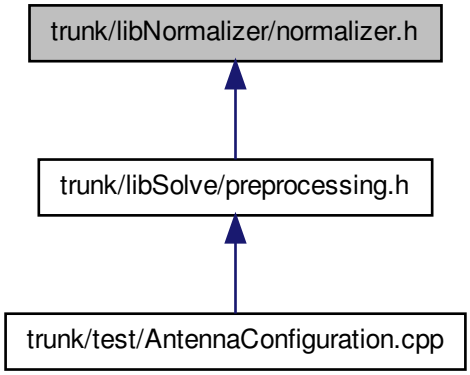
7.15 trunk/libNormalizer/normalizer.h File Reference

```
#include <stdio.h> #include <stdlib.h> #include <iostream> ×
#include "../include/PRPSEvolutionGeneralExceptions.-
h"    #include "../include/PRPSEvolutionFIOExceptions.h" ×
#include <complex>  #include <array>  #include <random> ×
```

Include dependency graph for normalizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Normalizer< N, T >

Namespaces

- namespace PRPSEvolution

Enumerations

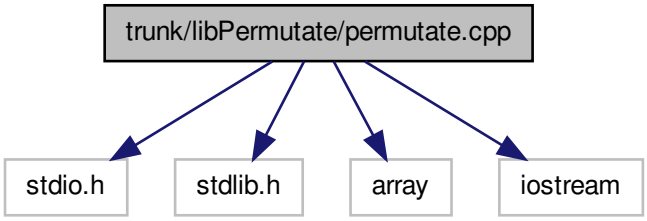
- enum PRPSEvolution::NormalizatioMethodes { PRPSEvolution::Native, PRPSEvolution::B, PRPSEvolution::CMPLX, PRPSEvolution::RND }

7.15.1 Detailed Description

Collects normalizations for the input data

7.16 trunk/libPermutate/permutate.cpp File Reference

`#include <stdio.h> #include <stdlib.h> #include <array> ×`
`#include <iostream>` Include dependency graph for permutate.cpp:



Functions

- void test2 ()

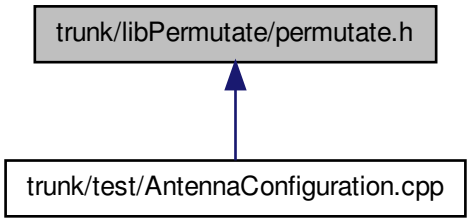
7.16.1 Function Documentation

7.16.1.1 void test2 ( )

7.17 trunk/libPermutate/permutate.h File Reference

`#include <stdio.h> #include <stdlib.h> #include <iterator> ×`
`#include <iostream> #include <algorithm> #include <array> ×`
`#include <string> #include "../include/coords.h" #include "../include/PRPSEvolution.h" #include "../include/PRPS-EvolutionPermutationExceptions.h" #include "../include/-PRPSEvolutionFIOExceptions.h" #include "../include/PRPS-Error.h" #include "../libPRPSSystem/prpsevolutionsystem.-h" #include "nr3/nr3.h" #include "nr3/svd.h"` Include dependency graph for permutate.h:

This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Permutate::AntennaPermutations< N_MAT, T >
- struct PRPSEvolution::Permutate::permuteAntennas< N_ANTA, N_ANTPERM, T >

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Permutate

Functions

- int PRPSEvolution::Permutate::Factorial (int x)
- template<typename Iterator >
  bool PRPSEvolution::Permutate::next_combination (const Iterator first, Iterator k, const Iterator last)

Variables

- const int PRPSEvolution::Permutate::MAX_PERMUTATION_AMOUNT = 35

7.17.1 Detailed Description

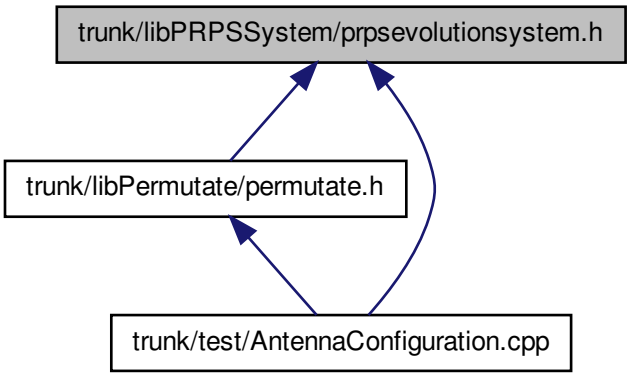
Date

2013|Jun|25

7.18 trunk/libPRPSSystem/prpsevolutionsystem.cpp File Reference

7.19 trunk/libPRPSSystem/prpsevolutionsystem.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Constants
- struct PRPSEvolution::System

Namespaces

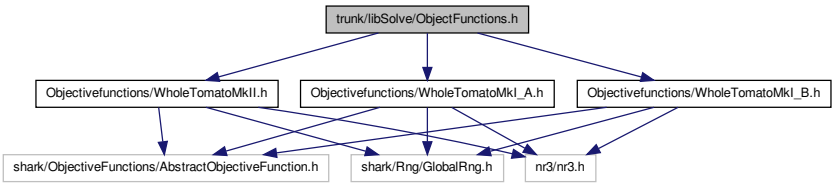
- namespace PRPSEvolution

7.20 trunk/libSolve/ObjectFunctions.cpp File Reference

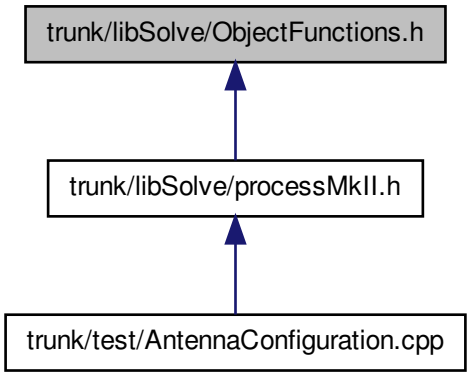
7.21 trunk/libSolve/ObjectFunctions.h File Reference

```
#include "Objectivefunctions/WholeTomatoMkII.h"   #include
"Objectivefunctions/WholeTomatoMkI_A.h" #include "Objectivefunctions/-
```

WholeTomatoMkI_B.h" Include dependency graph for ObjectFunctions.h:



This graph shows which files directly or indirectly include this file:

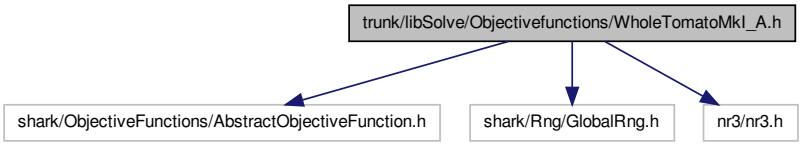


7.22 trunk/libSolve/Objectivefunctions/WholeTomatoMkI.cpp File - Reference

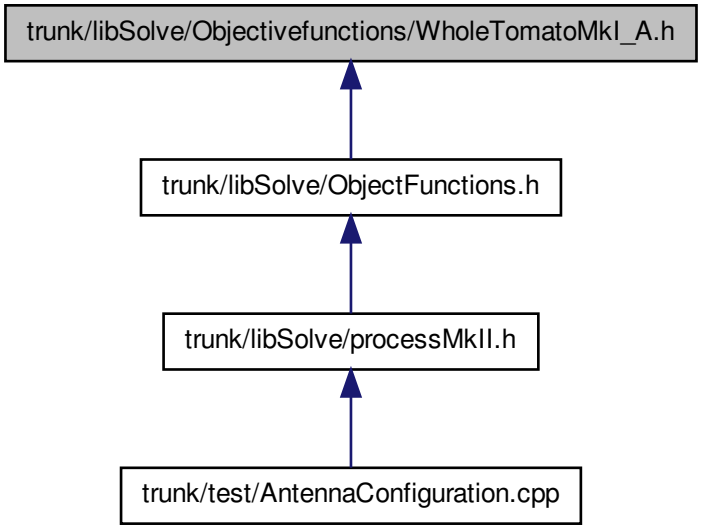
7.23 trunk/libSolve/Objectivefunctions/WholeTomatoMkI_A.h File - Reference

```
#include <shark/ObjectiveFunctions/AbstractObjective-
Function.h>   #include <shark/Rng/GlobalRng.h>   #include
```

<nr3/nr3.h> Include dependency graph for WholeTomatoMkI_A.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::WholeTomatoMkI_A

Namespaces

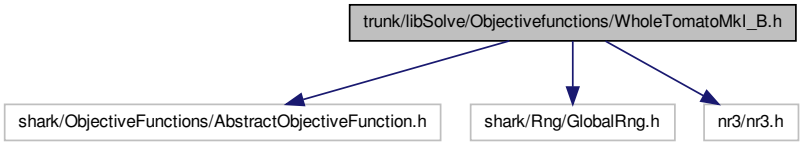
- namespace PRPSEvolution

Functions

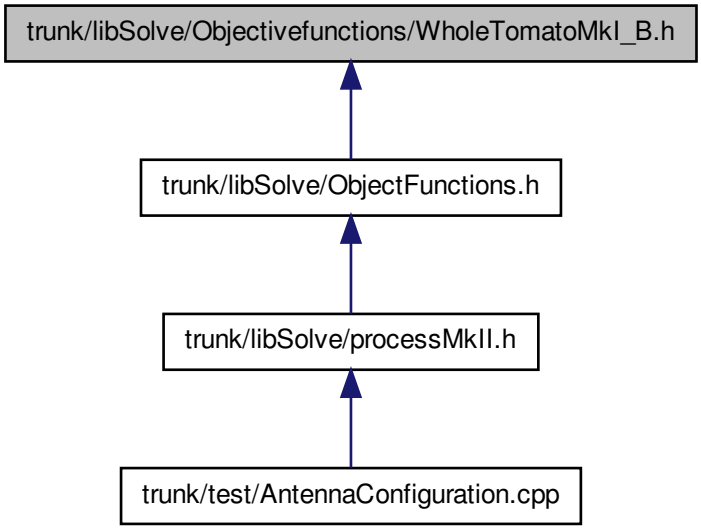
- PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkl_A, shark::soo::RealValuedObjectiveFunctionFactory)

7.24 trunk/libSolve/Objectivefunctions/WholeTomatoMkl_B.h File - Reference

`#include <shark/ObjectiveFunctions/AbstractObjective-Function.h>` `#include <shark/Rng/GlobalRng.h>` `#include <nr3/nr3.h>` Include dependency graph for WholeTomatoMkl_B.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::WholeTomatoMkI_B

Namespaces

- namespace PRPSEvolution

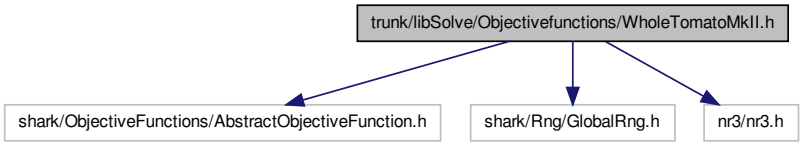
Functions

- PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkI_B, shark::soo::RealValuedObjectiveFunctionFactory)

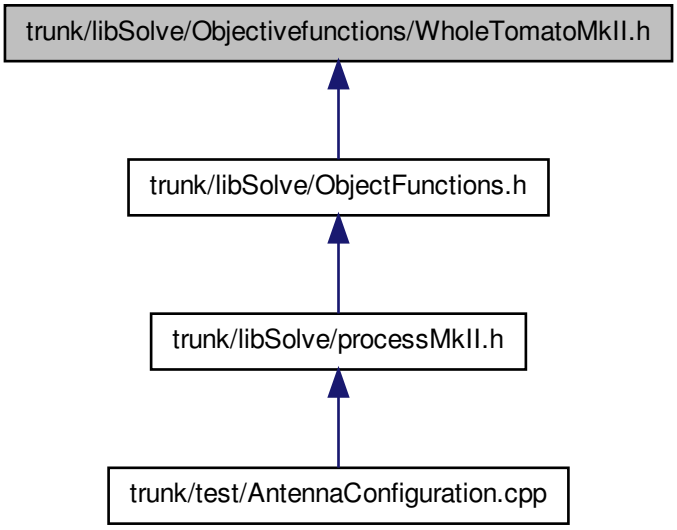
7.25 trunk/libSolve/Objectivefunctions/WholeTomatoMkII.h File Reference

```
#include <shark/ObjectiveFunctions/AbstractObjective-
Function.h>   #include <shark/Rng/GlobalRng.h>   #include
```

<nr3/nr3.h> Include dependency graph for WholeTomatoMkII.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::WholeTomatoMkII

Namespaces

- namespace PRPSEvolution

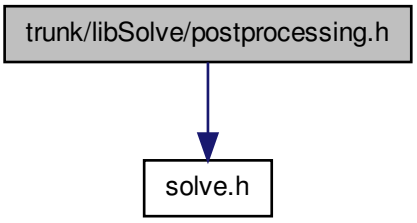
Functions

- PRPSEvolution::ANNOUNCE_SINGLE_OBJECTIVE_FUNCTION (WholeTomatoMkII, soo::RealValuedObjectiveFunctionFactory)

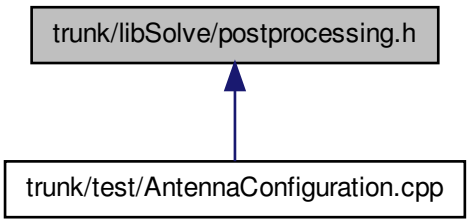
7.26 trunk/libSolve/postprocessing.cpp File Reference

7.27 trunk/libSolve/postprocessing.h File Reference

`#include "solve.h"` Include dependency graph for postprocessing.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PRPSEvolution::Solve::PostProcessing

Namespaces

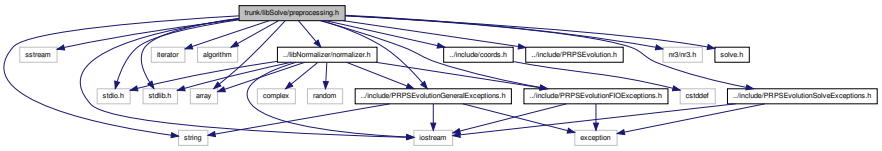
- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

7.28 trunk/libSolve/preprocessing.cpp File Reference

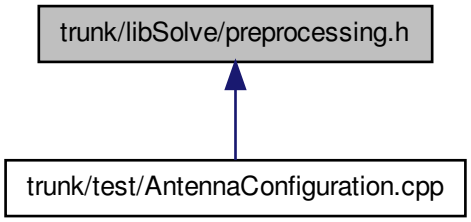
7.29 trunk/libSolve/preprocessing.h File Reference

```
#include <iostream> #include <sstream> #include <string> ×
#include <stdio.h> #include <stdlib.h> #include <iterator> ×
#include <algorithm>  #include <array>  #include "../lib-
Normalizer/normalizer.h"    #include "../include/coords.h"
#include "../include/PRPSEvolution.h" #include "../include/-
PRPSEvolutionSolveExceptions.h"     #include "../include/-
PRPSEvolutionFIOExceptions.h"     #include "../include/PR-
PSEvolutionGeneralExceptions.h"     #include "nr3/nr3.h" ×
#include "solve.h"
```

Include dependency graph for preprocessing.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PRPSEvolution::Solve::PreProcessing< N_ANTA, N_Configs, T, T_-Measure >

Namespaces

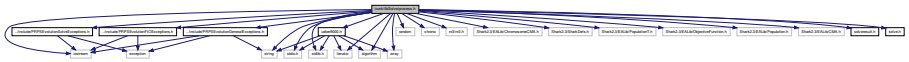
- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

7.30 trunk/libSolve/process.cpp File Reference

7.31 trunk/libSolve/process.h File Reference

```
#include <iostream> #include <string> #include <random>
#include <stdio.h> #include <chrono> #include <stdlib.-
h> #include <iterator> #include <algorithm> #include
<array> #include "nr3/nr3.h" #include <Shark2.3/EALib/-
ChromosomeCMA.h> #include <Shark2.3/SharkDefs.h> #include
<Shark2.3/EALib/PopulationT.h> #include <Shark2.3/EALib/-
ObjectiveFunction.h> #include <Shark2.3/EALib/Population.-
h> #include <Shark2.3/EALib/CMA.h> #include "../include/-
PRPSEvolutionSolveExceptions.h" #include "../include/P-
RPSEvolutionFIOExceptions.h" #include "../include/PRPS-
EvolutionGeneralExceptions.h" #include "solveresult.h" ×
#include "solve.h" #include "ueber9000.h"
```

Include dependency graph for process.h:



Classes

- class PRPSEvolution::Solve::Process

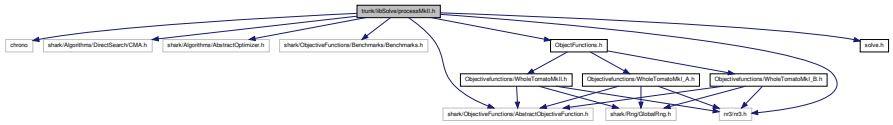
Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

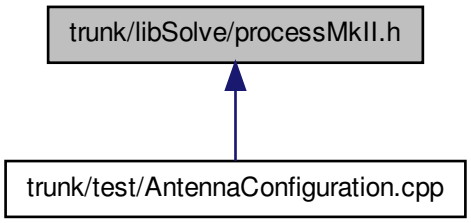
7.32 trunk/libSolve/processMkII.cpp File Reference

7.33 trunk/libSolve/processMkII.h File Reference

`#include <chrono>` `#include <shark/Algorithms/DirectSearch/CMA.h>` `#include <shark/Algorithms/AbstractOptimizer.h>` `#include <shark/ObjectiveFunctions/Benchmarks/Benchmarks.h>` `#include <shark/ObjectiveFunctions/AbstractObjectiveFunction.h>` `#include "solve.h"` `#include "ObjectFunctions.h"` `#include <nr3/nr3.h>` Include dependency graph for processMkII.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PRPSEvolution::Solve::Process_MkII

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

Defines

- #define STUFF(Function, Vars)
- #define SOLVE(MODEL)
- #define SOLVE_AND_WRITE(MODEL)

7.33.1 Define Documentation

7.33.1.1 #define SOLVE(*MODEL*)

Value:

```
shark::CMA cma;
       \
         cma.init( MODEL );
       \
         do { cma.step( MODEL ); } while(cma.solution().value > epsilon );
       \
```

7.33.1.2 #define SOLVE_AND_WRITE(*MODEL*)

Value:

```
shark::CMA cma;
       \
         cma.init( MODEL );
       \
         do {
       \
                  cma.step( MODEL );
       \
                  f << model.evaluationCounter() << " "
       \
                                                 << cma.solution().value << " "
       \
                                                 << cma.solution().point << " "
       \
                                                 << cma.sigma()
       \
                                                 << std::endl;
       \
         } while(cma.solution().value > epsilon
       \
                  && model.evaluationCounter() < maxEvaluations);
       \
```

7.33.1.3 #define STUFF(*Function, Vars*)

Value:

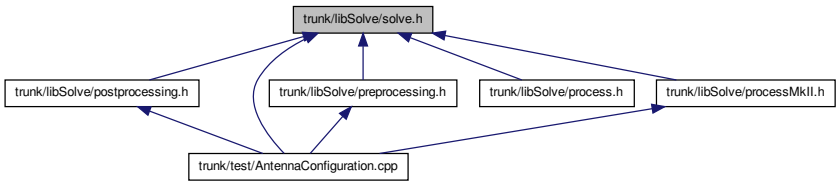
```
Function model(Vars);
       \
```

```
  model.setNumberOfVariables(Vars);
\
  shark::CMA cma;
\
  cma.init( model );
\
  do { cma.step( model ); } while(cma.solution().value > epsilon );
\
```

7.34 trunk/libSolve/solve.cpp File Reference

7.35 trunk/libSolve/solve.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Solve::ProblemDimensions

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

Enumerations

- enum PRPSEvolution::Solve::SelectBy { PRPSEvolution::Solve::ConditionNumber, PRPSEvolution::Solve::Random, PRPSEvolution::Solve::AllPossible, PRPSEvolution::Solve::Best10ByCN, PRPSEvolution::Solve::AllFrom4Ant }
- enum PRPSEvolution::Solve::ESStrategy { PRPSEvolution::Solve::OnePlusOne, PRPSEvolution::Solve::MuPlusLambda, PRPSEvolution::Solve::MuCommaLambda, PRPSEvolution::Solve::MuCommaLambda_MkII, PRPSEvolution::Solve::MuPlusLambda_MkII, PRPSEvolution::Solve::CMA_ES_MkI, PRPSEvolution::Solve::CMA_ES_MkII }
- enum PRPSEvolution::Solve::Models { PRPSEvolution::Solve::WholeTomatoMkI, PRPSEvolution::Solve::WholeTomatoMkII, PRPSEvolution::Solve::TestSphere }

Functions

- double PRPSEvolution::Solve::meanFromVector (std::vector< double > &res)

Variables

- const int PRPSEvolution::Solve::nConfigsForProcessing = 1

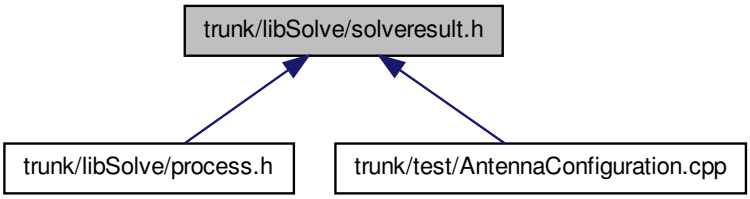
7.35.1 Detailed Description

Date

2013|Jun|25

7.36 trunk/libSolve/solveresult.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Solve::solveresult_t< T_Store1, T_Store2, T_Return >

Namespaces

- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

7.36.1 Detailed Description

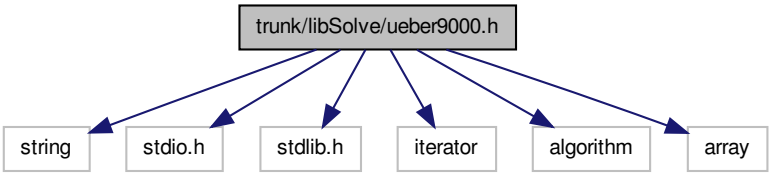
Date

2013|Jul|5

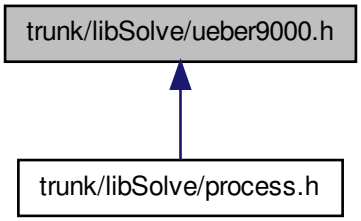
7.37 trunk/libSolve/ueber9000.cpp File Reference

7.38 trunk/libSolve/ueber9000.h File Reference

`#include <string> #include <stdio.h> #include <stdlib.h> #include <iterator> #include <algorithm> #include <array>` Include dependency graph for ueber9000.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PRPSEvolution::Solve::Ueber9000< T >

Namespaces

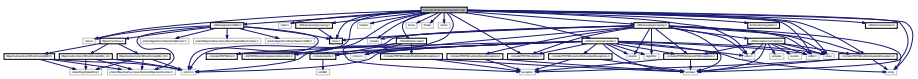
- namespace PRPSEvolution
- namespace PRPSEvolution::Solve

Variables

- std::mutex PRPSEvolution::Solve::wMutex
- int PRPSEvolution::Solve::_i_ = 0

7.39 trunk/test/AntennaConfiguration.cpp File Reference

```
#include <stdio.h>  #include <stdlib.h>  #include <math.-
h> #include <array> #include <iostream> #include <exception> ×
#include <fstream>  #include <sstream>  #include <string>
#include <chrono>  #include <future>  #include <thread> ×
#include <vector>    #include "../libSolve/processMkII.h"
#include "../include/PRPSEvolution.h" #include "../include/-
PRPSError.h"    #include "../include/PRPSEvolutionGeneral-
Exceptions.h"  #include "AntennaConfiguration.h"  #include
"../libPermutate/permutate.h" #include "../libPRPSSystem/prpsevolutionsystem.-
h"  #include "../libCalibration/calib.h"  #include "../lib-
Solve/solve.h" #include "../libSolve/solveresult.h" #include
"../libSolve/preprocessing.h" #include "../libSolve/postprocessing.-
h" Include dependency graph for AntennaConfiguration.cpp:
```



Defines

- #define _USE_SHARK_3_0_
- #define _Write_Result
- #define _DROP_BAD_

Functions

- int main (int argc, char ∗argv[])

Variables

- const int SOLUTION_AMOUNT = 1

- int VARIANT_SW
- int NO_OF_SOLUTIONS
- bool DROPBAD = false
- std::string FILENAME = ""

7.39.1 Detailed Description

This File contains the main() of the AntennaApp-Project

7.39.2 Define Documentation

7.39.2.1 #define _DROP_BAD_

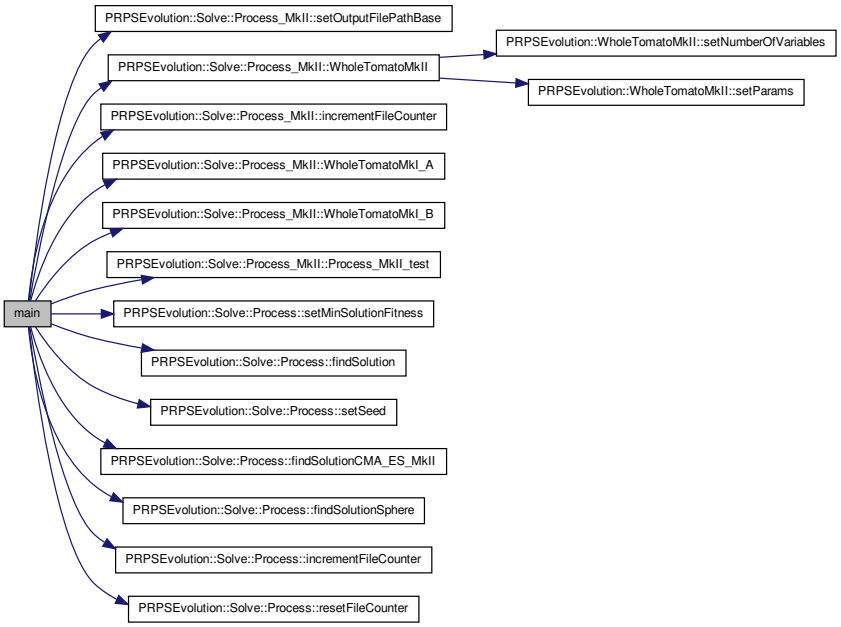
7.39.2.2 #define _USE_SHARK_3_0_

7.39.2.3 #define _Write_Result

7.39.3 Function Documentation

7.39.3.1 int main (int *argc*, char * *argv[]*)

Here is the call graph for this function:



7.39.4 Variable Documentation

7.39.4.1 bool DROPBAD = false

7.39.4.2 std::string FILENAME = ""

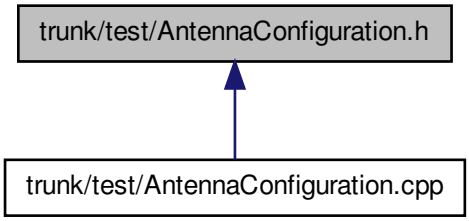
7.39.4.3 int NO_OF_SOLUTIONS

7.39.4.4 const int SOLUTION_AMOUNT = 1

7.39.4.5 int VARIANT_SW

7.40 trunk/test/AntennaConfiguration.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define VERSION_MAJOR 0
- #define VERSION_MINOR 1
- #define VERSION_SUB_MINOR 1

7.40.1 Define Documentation

7.40.1.1 #define VERSION_MAJOR 0

7.40.1.2 #define VERSION_MINOR 1

7.40.1.3 #define VERSION_SUB_MINOR 1