# AnaMod: a package for numerical Analysis Modules

Generated by Doxygen 1.5.9

Sun Oct 4 04:01:29 2009

# Contents

# 1 AnaMod : an Analysis Modules library

## 1.1 Introduction

This is a library of modules that can compute various properties of a matrix. The code heavily uses the Petsc library, and the matrix has to be stored in Petsc format.

Modules are divided into several categories, each defined in its own file. Typical categories are for structural information, normlike properties, spectral estimate. A number of modules have been supplied, but because of the modular setup, it is easy to add new modules.

This library was written for use in the Salsa project (http://icl.cs.utk.edu/salsa/), but can be used independently of it. If the Salsa NMD library (http://icl.cs.utk.edu/salsa/software/index.html) is present (see the installation instructions), its data types are used.



**Author:**

Victor Eijkhout

**Version:**

1.91

**Date:**

unreleased

Change log:

- 1.91 : spectrum modules with ";re" and ";im" have been renamed to "-re" and "-im" for MySQL sake.
- 1.9 : fixed memory leak in dummy row computation
- 1.8.a : added the stats.c file
- 1.8 : much code cleanup; unit tests to ensure proper operation

- 1.7 : handling of arrays is now completely changed. most routines have acquired an explicit array length parameter
- 1.6 : handling of the location of slepc and such. Make sure to check the Make.inc.example file
- 1.5 :
  - module-specific options
  - `-anamod_force` now has underscore for consistency
  - several spectrum imaginary parts were miscomputed
  - added SLEPC computation of spectrum
- 1.4 :
  - improved trace mode; incompatible change in the prototype of the trace function
  - removed a11 module
- 1.3 :
  - trace mode added (see Tracing the analysis modules)
  - renamed header files: module_functions.h -> anamod.h , module_types.h -> anamodtypes.h , nmdmodules.h -> anamodsalsamodules.h , module_utils.h -> anamodutils.h
- 1.2 :
  - added runtime options: forced computation of single-processor modules in parallel context, of expensive modules
  - bug fix of aux:a11.
  - compatibility with gcc4 (ignoring 'const' used to be a warning, it's now an error)
- 1.1 : correct handling of parallelism

# 2 Matrix interface to AnaMod

AnaMod uses an abstract AnaModNumericalProblem in the definition of ComputeQuantity() and such. For modules where the numerical problem is solely a Petsc matrix, MatrixComputeQuantity() is the interface.

# 3 Feature sets

AnaMod has a mechanism for querying single category/component pairs, but often a specific set of features is needed quickly, and/or multiple times. For this, the 'feature set' mechanism exists.

A feature set is a set of category/component pairs. It does not contain actual values. A feature set can be instantiated to a 'feature values' object, which contains the actual values that the feature set takes on a given problem.

This is the workflow:

- a FeatureSet object is created, once, with NewFeatureSet().
- category/component pairs are added to it with AddToFeatureSet().
- a FeatureValues object is created with NewFeatureValues().
- a call to InstantiateFeatureSet() fills in the feature set on a specified numerical problem.
- user code can retrieve values with GetFeatureValue()

Here is a piece of example code:

```
  {
    FeatureSet symmetry; int sidx,aidx;
    ierr = NewFeatureSet(&symmetry); CHKERRQ(ierr);
    ierr = AddToFeatureSet
      (symmetry,"simple","symmetry-snorm",&sidx); CHKERRQ(ierr);
    ierr = AddToFeatureSet
      (symmetry,"simple","symmetry-anorm",&aidx); CHKERRQ(ierr);
    ierr = TransformObjectSetSuitabilityFunction
      (cur,(void*)symmetry,&onlyforsymmetricproblem); CHKERRQ(ierr);
  }

PetscErrorCode onlyforsymmetricproblem
(NumericalProblem problem,void *ctx,SuitabilityValue *v)
{
  FeatureSet features = (FeatureSet)ctx; FeatureValues values;
  AnalysisItem sn,an; PetscTruth f1,f2; PetscErrorCode ierr;

  PetscFunctionBegin;
  ierr = NewFeatureValues(&values); CHKERRQ(ierr);
  ierr = InstantiateFeatureSet((void*)problem,features,values); CHKERRQ(ierr);
  ierr = GetFeatureValue(values,sidx,&sn,&f1); CHKERRQ(ierr);
  ierr = GetFeatureValue(values,aidx,&an,&f2); CHKERRQ(ierr);
  ierr = DeleteFeatureValues(values); CHKERRQ(ierr);
  if (f1 && f2 && an.r>1.e-12*sn.r) {
    printf("problem too unsymmetric\n");
  }
```

# 4 Installation and compiling programs

## 4.1 Installation

See the README file. The main thing is that you need to have Petsc installed.

## 4.2 Compilation of user programs

The easiest way to compile a program that uses the Salsa Analysis Modules is to have these includes in your makefile:

```
include $(PETSC_DIR)/bmake/common/variables
include $(SALSA_MODULES_DIR)/Make.inc
```

The following flags for your C compiler are

- line 1: Petsc option
- line 2: Analysis Modules options
- line 3: (optionally) options for using the NMD library; HAVE_NMD_DEFINE has to be "-DHAVE_NMD" if the library is present

```
CFLAGS = \
    $(PETSC_INCLUDE) $(COPTFLAGS) \
    -I$(SALSA_MODULES_DIR) \
    $(HAVE_NMD_DEFINE) -I$(LIBNMD_INCLUDE_DIR) -I$(LIBXMLSC_INCLUDE_DIR)
```

Your program needs to include the following header files:

```
#include "anamod.h"
#include "anamodsalsamodules.h"
```

The following link line brings together all the needed libraries (add libnmd if required)

```
yourprog :
$(CLINKER) -o yourprog yourprog.o \
    -L$(SALSA_MODULES_LIB_DIR) -lsalsamodules -lothermodules \
    $(PETSC_LIB)
```

# 5 Use of the analysis modules

## 5.1 Library Initialization

There are calls AnaModInitialize() and AnaModFinalize() which do global allocation.

## 5.2 Category registration

Every category needs to be registered before use with a call

Register<Category>Modules();

where the available categores are Inverse Cuthill-McKee distribution, Intelligent Preconditioner Recommendation System, Jones-Plassmann multi-colouring, Estimates for the departure from normality, Simple (normlike) quantities, Spectral properties, Structural properties, Measurements of element variance. The functions AnaModRegisterStandardModules() installs all standard available modules.

## 5.3 Quantity computation

After registring a category, its elements can be computed as

PetscErrorCode ComputeQuantity(char *cat,char *cmp,Mat A,AnalysisItem *res,PetscTruth *success);

with

1. the name of the category
2. the name of the element
3. the matrix
4. the result
5. a success indicator

For the main user functions, see the module functions file

Commandline options are discussed in section Commandline options.

## 5.4 types

The `AnalysisDataType` type is an integer. To get a printable name of the datatype, use AnaModGetTypeName() or AnaModGetTypeMySQLName().

# 6 List of the available modules

Inverse Cuthill-McKee distribution

Intelligent Preconditioner Recommendation System

Jones-Plassmann multi-colouring

# 7 Commandline options

The runtime behaviour of AnaMod can be influenced by commandline options, specified at the start of the calling application.

All options start with `"-anamod"` and they can have a sequence of comma-separated values separated from the option by a blank: `"-anamod_option value1,value2"`. Note the single dash, which is Petsc style, as opposed to the double dash of GNU style.

- `-anamod_force` : forced computation of certain computations that might otherwise be ignored. Values:
  - `"expensive"` : certain modules are very expensive to compute in certain circumstances, so AnaMod will normally refuse to compute them. (See Estimates for the departure from normality.) This option forces their computation, no matter how much time they may take.
  - `"sequential"` : certain modules are only implemented as single processor code, so they will normally not be computed in a parallel run. This option will force processor zero to gather the full matrix, and perform the computation locally. I hope I do not have to point out the potential pitfalls of this option. Sequential modules are the norms of the symmetric/anti-symmetric part in the Simple (normlike) quantities category.
- `-anamod_compute` : (not implemented yet)

The user will unlikely need access to the commandline options from the code. The programmatic interface to commandline options handling is in file Commandline Options for Runtime Control.

# 8 Customization

While this library comes with a number of categories supplied, you can write your own category, taking any of the given ones as example.

Each analysis module has to have the following prototype:

```
static PetscErrorCode MyModule(Mat A,AnalysisItem *rv,PetscTruth *flg)
```

where

- `A` is the input matrix
- `rv` is the return value
- `flg` is true if the quantity was succesfully computed, false otherwise

The module return code should be 0 for success, anything else for (catastrophic) failure. A simple failure (or refusal) to compute should be indicated through the `flg` variable.

The modules (including the return type of their computed quantities) are declared to the system by calling routine

```
PetscErrorCode RegisterMyModules()
{
  PetscErrorCode ierr;
  PetscFunctionBegin;

  ierr = RegisterModule
    ("mycategory","mymodule",THERESULTTYPE,&MyModule); CHKERRQ(ierr);
  ....
```

or you can make the invidual calls to RegisterModule(). See types and Array type handling.

# 9 Array type handling

The actual types are listed in types ; here are some semantic issues pertaining to array types.

The uniform calling structure of the modules allows only one parameter to be returned. This is a problem only for array types: we want both the array data and the length of the array. Hence we adopt the convention that every array is longer by one element than necessary, and the zero element contains the length of the array proper. That is, to store 5 elements, 6 elements are allocated, and the zero element contains the number '5'.

If, in this documentation, we refer to location array[i], we will take this to mean location 'i' after the size element. Zero-based indexing is used.

# 10 Attached quantities in Petsc

Petsc has a mechanism where quantities can be attached to objects. For instance, integers can be attached and queried with PetscObjectComposedDataSetInt() and PetscOb-

jectComposedDataGetInt(). Such data is attached together with the current state of the object, which means that it will be properly invalidated if the object is altered.

This composition mechanism is used in this analysis modules package. There are many quantities (such as the norms of the symmetric and anti-symmetric part of the matrix) that can be computed together at practically the same cost as either one alone. Thus, when one is requested by calling ComputeQuantity(), the other one will be compute as well, and attached to the matrix object. A call to ComputeQuantity() for this second quantity will then just return the attached number, and not be computed separately.

The routine HasQuantity() can query which quantities are already attached.

As a result of the use of attached quantities, benchmarking the AnaMod modules is a bit tricky. You can forced computation of the requested quantities by calling PetscObjectIncreaseState() on the matrix object after each call to ComputeQuantity().

# 11 Bulk computation

After the modules have been installed, the typical user will only use ComputeQuantity(). However, for purposes of reporting and such it may be necessary to query systematicaly the available modules. This can be done with GetCategories() and CategoryGetModules().

Related: HasComputeCategory(), HasComputeModule().

Auxiliary routines: GetCategoryIndex(), GetModuleIndex().

# 12 Module Index

## 12.1 Modules

Here is a list of all modules:



# 13 Data Structure Index

## 13.1 Data Structures

Here are the data structures with brief descriptions:

# 14 File Index

## 14.1 File List

Here is a list of all files with brief descriptions:

# 15 Module Documentation

## 15.1 Files with defined categories

### Files

- file icmk.c

  *Functions for computing the ICMK structure of a matrix.*

- file iprs.c

  *Quantities used in the UKY 'Intelligent Preconditioner Recommendation System'.*

- file jpl.c

  *Functions for computing the JPL multicolour structure of a matrix.*

- file lapack.c

*Dense routines from Lapack: eigenvalue and schur stuff.*

- file normal.c

  *Various estimates of the departure from normality.*

- file simple.c

  *Norm-like properties.*

- file spectrum.c

  *Various estimates of spectrum related quantities.*

- file stats.c

  *Statistics on the AnaMod system.*

- file structure.c

  *Structural properties of a matrix.*

- file variance.c

  *Various estimates of how 'wild' a matrix is.*

### 15.1.1 Detailed Description

## 15.2 User-accessible functions

### Files

- file anamod.h

  *Prototypes for general module functions.*

- file anamodsalsamodules.h

  *Prototypes for using the standard modules.*

- file module_functions.c

  *User functions for accessing the analysis modules.*

### 15.2.1 Detailed Description

## 15.3 Functions only of use to the implementation

**Files**

- file anamodtypes.h
- file options.c

  *Commandline options handling.*

- file utils.c

### 15.3.1 Detailed Description

# 16 Data Structure Documentation

## 16.1 AnalysisDataTypeArray_ Struct Reference

**Data Fields**

- char * name
- int alloc
- int fill
- int * has
- AnalysisDataType * data

### 16.1.1 Detailed Description

Definition at line 330 of file anamodutils.c.

### 16.1.2 Field Documentation

#### 16.1.2.1 int AnalysisDataTypeArray_::alloc

Definition at line 331 of file anamodutils.c.

Referenced by AnalysisDataTypeArrayAdd(), AnalysisDataTypeArrayGetAt(), AnalysisDataTypeArraySetAt(), AnalysisDataTypeArrayTryGetAt(), and CreateAnalysisDataTypeArray().

#### 16.1.2.2 AnalysisDataType* AnalysisDataTypeArray_::data

Definition at line 331 of file anamodutils.c.

Referenced by AnalysisDataTypeArrayAdd(), AnalysisDataTypeArrayGetAt(), AnalysisDataTypeArraySetAt(), AnalysisDataTypeArrayTryGetAt(), CreateAnalysisDataTypeArray(), and DeleteAnalysisDataTypeArray().

#### 16.1.2.3 int AnalysisDataTypeArray_::fill

Definition at line 331 of file anamodutils.c.

Referenced by AnalysisDataTypeArrayAdd(), AnalysisDataTypeArraySetAt(), and CreateAnalysisDataTypeArray().

#### 16.1.2.4 int * AnalysisDataTypeArray_::has

Definition at line 331 of file anamodutils.c.

Referenced by AnalysisDataTypeArrayAdd(), AnalysisDataTypeArrayGetAt(), AnalysisDataTypeArraySetAt(), AnalysisDataTypeArrayTryGetAt(), CreateAnalysisDataTypeArray(), and DeleteAnalysisDataTypeArray().

#### 16.1.2.5 char* AnalysisDataTypeArray_::name

Definition at line 331 of file anamodutils.c.

Referenced by CreateAnalysisDataTypeArray(), and DeleteAnalysisDataTypeArray().

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

- anamodutils.c

## 16.2 AnalysisItem Union Reference

`#include <anamodtypes.h>`

**Data Fields**

- int i
- int len
- double r
- int * ii
- double * rr
- char * c

### 16.2.1 Detailed Description

Definition at line 11 of file anamodtypes.h.

### 16.2.2 Field Documentation

#### 16.2.2.1 char* AnalysisItem::c

Definition at line 17 of file anamodtypes.h.

Referenced by QuantityAsString(), and Version().

#### 16.2.2.2 int AnalysisItem::i

Definition at line 13 of file anamodtypes.h.

Referenced by AvgDistFromDiag(), AvgNnzpRow(), BlockSize(), computennz(), DiagDefinite(), DiagonalSign(), DiagZeroStart(), DummyRowsKind(), LBandWidth(), LoBand(), MaxNNonZerosPerRow(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), NRitzValues(), nRows(), NSplits(), NUnstruct(), QuantityAsString(), RBandWidth(), RelSymm(), RetrieveQuantityByID(), Symmetry(), and UpBand().

#### 16.2.2.3 int* AnalysisItem::ii

Definition at line 15 of file anamodtypes.h.

Referenced by ColourOffsets(), Colours(), ColourSizes(), DummyRows(), LeftSkyline(), QuantityAsString(), RetrieveQuantityByID(), RightSkyline(), and Splits().

#### 16.2.2.4 int AnalysisItem::len

Definition at line 13 of file anamodtypes.h.

Referenced by QuantityAsString().

#### 16.2.2.5 double AnalysisItem::r

Definition at line 14 of file anamodtypes.h.

Referenced by AvgDiagDist(), ColVariability(), Commutator(), Departure(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), DiagonalAverage(), DiagonalDominance(), DiagonalVariance(), Kappa(), Lee95bounds(), MatCenter(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), norm1(), normF(), normInf(), PosFraction(), QuantityAsString(), RelSymm(), RetrieveQuantityByID(), RowVariability(), SigmaDiagDist(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), and TraceAbs().

#### 16.2.2.6 double* AnalysisItem::rr

Definition at line 16 of file anamodtypes.h.

Referenced by MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), QuantityAsString(), RetrieveQuantityByID(), RitzValuesC(), and RitzValuesR().

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

- anamodtypes.h

## 16.3 AnalysisItemArray_ Struct Reference

Collaboration diagram for AnalysisItemArray_:

### Data Fields

- char * name
- int alloc

- int fill
- int * has
- AnalysisItem * data

### 16.3.1 Detailed Description

Definition at line 226 of file anamodutils.c.

### 16.3.2 Field Documentation

#### 16.3.2.1 int AnalysisItemArray_::alloc

Definition at line 227 of file anamodutils.c.

Referenced by AnalysisItemArrayAdd(), AnalysisItemArrayGetAt(), AnalysisItemArraySetAt(), AnalysisItemArrayTryGetAt(), and CreateAnalysisItemArray().

#### 16.3.2.2 AnalysisItem* AnalysisItemArray_::data

Definition at line 227 of file anamodutils.c.

Referenced by AnalysisItemArrayAdd(), AnalysisItemArrayGetAt(), AnalysisItemArraySetAt(), AnalysisItemArrayTryGetAt(), CreateAnalysisItemArray(), and DeleteAnalysisItemArray().

#### 16.3.2.3 int AnalysisItemArray_::fill

Definition at line 227 of file anamodutils.c.

Referenced by AnalysisItemArrayAdd(), AnalysisItemArraySetAt(), and CreateAnalysisItemArray().

#### 16.3.2.4 int * AnalysisItemArray_::has

Definition at line 227 of file anamodutils.c.

Referenced by AnalysisItemArrayAdd(), AnalysisItemArrayGetAt(), AnalysisItemArraySetAt(), AnalysisItemArrayTryGetAt(), CreateAnalysisItemArray(), and DeleteAnalysisItemArray().

#### 16.3.2.5 char* AnalysisItemArray_::name

Definition at line 227 of file anamodutils.c.

Referenced by CreateAnalysisItemArray(), and DeleteAnalysisItemArray().

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

- anamodutils.c

## 16.4 FeatureSet_ Struct Reference

Collaboration diagram for FeatureSet_:

### Data Fields

- int cookie
- AnalysisDataType * types
- IntArray IDs
- StringArray features

### 16.4.1 Detailed Description

Definition at line 60 of file feature.c.

### 16.4.2 Field Documentation

#### 16.4.2.1 int FeatureSet_::cookie

Definition at line 61 of file feature.c.

Referenced by NewFeatureSet().

#### 16.4.2.2 StringArray FeatureSet_::features

Definition at line 62 of file feature.c.

Referenced by NewFeatureSet().

#### 16.4.2.3 IntArray FeatureSet_::IDs

Definition at line 62 of file feature.c.

Referenced by NewFeatureSet().

#### 16.4.2.4 AnalysisDataType* FeatureSet_::types

Definition at line 62 of file feature.c.

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

- feature.c

## 16.5 FeatureValues_ Struct Reference

Collaboration diagram for FeatureValues_:

### Data Fields

- int cookie
- AnalysisItemArray values
- AnalysisDataTypeArray types

### 16.5.1 Detailed Description

Definition at line 66 of file feature.c.

### 16.5.2 Field Documentation

#### 16.5.2.1 int FeatureValues_::cookie

Definition at line 67 of file feature.c.

Referenced by NewFeatureValues().

#### 16.5.2.2 AnalysisDataTypeArray FeatureValues_::types

Definition at line 68 of file feature.c.

Referenced by DeleteFeatureValues(), InstantiateFeatureSet(), and NewFeatureValues().

#### 16.5.2.3 AnalysisItemArray FeatureValues_::values

Definition at line 68 of file feature.c.

Referenced by DeleteFeatureValues(), GetFeatureValue(), InstantiateFeatureSet(), and NewFeatureValues().

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

- feature.c

## 16.6 IntArray_ Struct Reference

### Data Fields

- char * name
- int alloc
- int fill
- PetscTruth * has
- int * data

### 16.6.1 Detailed Description

Definition at line 7 of file anamodutils.c.

### 16.6.2 Field Documentation

#### 16.6.2.1 int IntArray_::alloc

Definition at line 8 of file anamodutils.c.

Referenced by CreateIntArray(), IntArrayAdd(), IntArrayGetAt(), IntArraySetAt(), and IntArrayTryGetAt().

#### 16.6.2.2 int* IntArray_::data

Definition at line 8 of file anamodutils.c.

Referenced by CreateIntArray(), DeleteIntArray(), IntArrayAdd(), IntArrayGetAt(), IntArraySetAt(), and IntArrayTryGetAt().

#### 16.6.2.3 int IntArray_::fill

Definition at line 8 of file anamodutils.c.

Referenced by CreateIntArray(), IntArrayAdd(), and IntArraySetAt().

#### 16.6.2.4 PetscTruth* IntArray_::has

Definition at line 8 of file anamodutils.c.

Referenced by CreateIntArray(), DeleteIntArray(), IntArrayAdd(), IntArrayGetAt(), IntArraySetAt(), and IntArrayTryGetAt().

#### 16.6.2.5 char* IntArray_::name

Definition at line 8 of file anamodutils.c.

Referenced by CreateIntArray(), and DeleteIntArray().

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

- anamodutils.c

## 16.7 StringArray_ Struct Reference

### Data Fields

- char * name
- int alloc
- int fill
- PetscTruth * has
- char ** data

#### 16.7.1 Detailed Description

Definition at line 111 of file anamodutils.c.

#### 16.7.2 Field Documentation

##### 16.7.2.1 int StringArray_::alloc

Definition at line 112 of file anamodutils.c.

Referenced by CreateStringArray(), StringArrayAdd(), StringArrayGetAt(), StringArraySetAt(), and StringArrayTryGetAt().

##### 16.7.2.2 char** StringArray_::data

Definition at line 112 of file anamodutils.c.

Referenced by CreateStringArray(), DeleteStringArray(), StringArrayAdd(), StringArrayGetAt(), StringArraySetAt(), and StringArrayTryGetAt().

##### 16.7.2.3 int StringArray_::fill

Definition at line 112 of file anamodutils.c.

Referenced by CreateStringArray(), DeleteStringArray(), StringArrayAdd(), StringArrayGetFill(), and StringArraySetAt().

#### 16.7.2.4 PetscTruth* StringArray_::has

Definition at line 112 of file anamodutils.c.

Referenced by CreateStringArray(), DeleteStringArray(), StringArrayAdd(), StringArrayGetAt(), StringArraySetAt(), and StringArrayTryGetAt().

#### 16.7.2.5 char* StringArray_::name

Definition at line 112 of file anamodutils.c.

Referenced by CreateStringArray(), and DeleteStringArray().

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

- anamodutils.c

# 17 File Documentation

## 17.1 anamatrix.c File Reference

```
#include <stdlib.h>
#include "anamod.h"
#include "anamatrix.h"
#include "petscmat.h"
```

Include dependency graph for anamatrix.c:

## Functions

- PetscErrorCode MatrixComputeQuantity (Mat A, const char ∗cat, const char ∗cmp, AnalysisItem ∗res, int ∗l, PetscTruth ∗success)

### 17.1.1 Function Documentation

#### 17.1.1.1 PetscErrorCode MatrixComputeQuantity (Mat *A*, const char ∗ *cat*, const char ∗ *cmp*, AnalysisItem ∗ *res*, int ∗ *l*, PetscTruth ∗ *success*)

Definition at line 15 of file anamatrix.c.

References ComputeQuantity().

Referenced by computennz(), JonesPlassmannColouring(), Lee95bounds(), MatCenter(), and MatCommutatorNormF_seq().

Here is the call graph for this function:

## 17.2 anamatrix.h File Reference

```
#include "anamod.h"
#include "petscmat.h"
```

Include dependency graph for anamatrix.h:

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

### Functions

- PetscErrorCode MatrixComputeQuantity (Mat, const char ∗, const char ∗, AnalysisItem ∗res, int ∗l, PetscTruth ∗success)

### 17.2.1 Function Documentation

#### 17.2.1.1 PetscErrorCode MatrixComputeQuantity (Mat, const char ∗, const char ∗, AnalysisItem ∗ *res*, int ∗ *l*, PetscTruth ∗ *success*)

Definition at line 15 of file anamatrix.c.

References ComputeQuantity().

Referenced by computennz(), JonesPlassmannColouring(), Lee95bounds(), MatCenter(), and MatCommutatorNormF_seq().

Here is the call graph for this function:

## 17.3 anamod.h File Reference

Prototypes for general module functions.

```
#include "petscmat.h"
#include "anamodtypes.h"
```

Include dependency graph for anamod.h:

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

### Defines

- #define ANAMOD_FORMAT_VERSION "1.0"
- #define TRUTH(x) ( (x) ? PETSC_TRUE : PETSC_FALSE )
- #define HASTOEXIST(h)
- #define ANAMODCHECKVALID(i, c, s) {if (!i) SETERRQ(1,"Null pointer"); if (i → cookie!=c) SETERRQ1(1,"Not a valid <%s>",s);}

### Typedefs

- typedef struct FeatureSet_ * FeatureSet
- typedef struct FeatureValues_ * FeatureValues

### Functions

- PetscErrorCode AnaModInitialize ()
- PetscErrorCode AnaModFinalize ()
- PetscErrorCode AnaModGetTypeName (int id, char **name)
- PetscErrorCode AnaModGetTypeMySQLName (int id, char **name)
- PetscErrorCode AnaModRegisterStandardModules ()
- PetscErrorCode RegisterModule (const char *, const char *, AnalysisDataType, PetscErrorCode(*f)(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *))
- PetscErrorCode DeRegisterCategory (char *cat)
- PetscErrorCode DeregisterModules (void)
- PetscErrorCode GetCategories (char ***, int *)
- PetscErrorCode CategoryGetModules (char *, char ***, AnalysisDataType **, int **, int *)
- PetscErrorCode HasComputeCategory (char *cat, PetscTruth *f)
- PetscErrorCode HasComputeModule (char *cat, char *cmp, PetscTruth *f)
- PetscErrorCode GetDataID (const char *, const char *, int *, PetscTruth *)
- PetscErrorCode GetDataType (const char *, const char *, AnalysisDataType *)
- PetscErrorCode ComputeQuantity (AnaModNumericalProblem, const char *, const char *, AnalysisItem *, int *, PetscTruth *)
- PetscErrorCode ComputeQuantityByID (AnaModNumericalProblem, int, int, AnalysisItem *, int *, PetscTruth *)
- PetscErrorCode HasQuantity (AnaModNumericalProblem, char *, char *, PetscTruth *)
- PetscErrorCode HasQuantityByID (AnaModNumericalProblem, int, AnalysisDataType, PetscTruth *)
- PetscErrorCode RetrieveQuantity (AnaModNumericalProblem, char *, char *, AnalysisItem *, PetscTruth *)
- PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem, int, int, AnalysisItem *, PetscTruth *)
- PetscErrorCode QuantityAsString (AnalysisItem *, AnalysisDataType, char **)
- PetscErrorCode AnaModOptionsHandling (void)
- PetscErrorCode AnaModShowOptions (MPI_Comm)
- PetscErrorCode DeclareCategoryOptionFunction (char *cat, PetscErrorCode(*f)(char *))
- PetscErrorCode GetCategoryOptionFunction (char *cat, PetscErrorCode(**f)(char *))

- PetscErrorCode AnaModHasForcedSequentialComputation (PetscTruth *)
- PetscErrorCode AnaModHasForcedExpensiveComputation (PetscTruth *)
- PetscErrorCode AnaModGetSequentialMatrix (Mat A, Mat *Awork, PetscTruth *mem, PetscTruth *local, PetscTruth *global)
- PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(*)(void *, char *, va_list))
- PetscErrorCode AnaModDeclareTraceContext (void *)
- PetscErrorCode AnaModTraceMessage (const char *fmt,...)
- PetscErrorCode AnaModHasTrace (PetscTruth *flg)
- PetscErrorCode AnaModSetTraceArrays (PetscTruth f)
- PetscErrorCode AnaModTraceArrays (PetscTruth *f)
- PetscErrorCode TabReportModules (Mat, char **, char **, int)
- PetscErrorCode PurgeAttachedArrays (Mat A)
- PetscErrorCode NewFeatureSet (FeatureSet *)
- PetscErrorCode DeleteFeatureSet (FeatureSet)
- PetscErrorCode AddToFeatureSet (FeatureSet, char *, char *, int *)
- PetscErrorCode NewFeatureValues (FeatureValues *)
- PetscErrorCode DeleteFeatureValues (FeatureValues)
- PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem, FeatureSet, FeatureValues)
- PetscErrorCode GetFeatureValue (FeatureValues, int, AnalysisItem *, PetscTruth *)
- PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode(*)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *))
- PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(**)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *), PetscTruth *)
- PetscErrorCode CreateIntArray (const char *, int, IntArray *)
- PetscErrorCode DeleteIntArray (IntArray)
- PetscErrorCode IntArrayAdd (IntArray, int, int *)
- PetscErrorCode IntArraySetAt (IntArray, int, int)
- PetscErrorCode IntArrayTryGetAt (IntArray, int, int *, PetscTruth *)
- PetscErrorCode IntArrayGetAt (IntArray, int, int *)
- PetscErrorCode IntArrayGetFill (IntArray, int *)
- PetscErrorCode CreateStringArray (const char *, int, StringArray *)
- PetscErrorCode DeleteStringArray (StringArray)
- PetscErrorCode StringArrayAdd (StringArray, char *, int *)
- PetscErrorCode StringArraySetAt (StringArray, int, char *)
- PetscErrorCode StringArrayTryGetAt (StringArray, int, char **, PetscTruth *)
- PetscErrorCode StringArrayGetAt (StringArray, int, char **)
- PetscErrorCode StringArrayGetFill (StringArray, int *)
- PetscErrorCode CreateAnalysisItemArray (char *, int, AnalysisItemArray *)
- PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray)

- PetscErrorCode AnalysisItemArrayAdd (AnalysisItemArray, AnalysisItem, int *)
- PetscErrorCode AnalysisItemArraySetAt (AnalysisItemArray, int, AnalysisItem)
- PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray, int, AnalysisItem *, PetscTruth *)
- PetscErrorCode AnalysisItemArrayGetAt (AnalysisItemArray, int, AnalysisItem *)
- PetscErrorCode CreateAnalysisDataTypeArray (char *, int, AnalysisDataTypeArray *)
- PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray)
- PetscErrorCode AnalysisDataTypeArrayAdd (AnalysisDataTypeArray, AnalysisDataType, int *)
- PetscErrorCode AnalysisDataTypeArraySetAt (AnalysisDataTypeArray, int, AnalysisDataType)
- PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray, int, AnalysisDataType *, PetscTruth *)
- PetscErrorCode AnalysisDataTypeArrayGetAt (AnalysisDataTypeArray, int, AnalysisDataType *)
- PetscErrorCode CategoryLogEventRegister (char *cat, int icat)

### 17.3.1 Detailed Description

Prototypes for general module functions.

This file defines the functions for defining and querying analysis modules.

Definition in file anamod.h.

### 17.3.2 Define Documentation

#### 17.3.2.1 #define ANAMOD_FORMAT_VERSION "1.0"

Definition at line 10 of file anamod.h.

Referenced by Version().

#### 17.3.2.2 #define ANAMODCHECKVALID(i, c, s) {if (!i) SETERRQ(1,"Null pointer"); if (i → cookie!=c) SETERRQ1(1,"Not a valid <%s>",s);}

Definition at line 21 of file anamod.h.

#### 17.3.2.3 #define HASTOEXIST(h)

**Value:**

```
if (!h) {                                       \
    printf("ERROR asking for unknown module\n"); \
    PetscFunctionReturn(1);                     \
  }
```

Definition at line 16 of file anamod.h.

Referenced by AvgDiagDist(), AvgDistFromDiag(), AvgNnzpRow(), BlockSize(), ColourOffsets(), Colours(), ColourSizes(), ColVariability(), Commutator(), compute_dd(), compute_dummy_rows(), compute_eigenvalues(), compute_ellipse_from_Ritz_values(), compute_icm_splits(), compute_nnz_structure(), compute_posdiag(), compute_singularvalues(), compute_tracea2(), ComputeDiagonal(), computennz(), computetrace(), ComputeVariability(), Departure(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), DiagDefinite(), DiagonalAverage(), DiagonalDominance(), DiagonalSign(), DiagonalVariance(), DiagZeroStart(), DummyRows(), DummyRowsKind(), eigenvaluecomp(), Kappa(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUnstruct(), PosFraction(), RBandWidth(), regularblocks(), RelSymm(), RightSkyline(), RitzValuesC(), RitzValuesR(), RowVariability(), SigmaDiagDist(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), Splits(), Symmetry(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), TraceAbs(), UpBand(), and Version().

#### 17.3.2.4 #define TRUTH(x) ( (x) ? PETSC_TRUE : PETSC_FALSE )

Definition at line 15 of file anamod.h.

Referenced by GetUpBiDiagSplits().

### 17.3.3 Typedef Documentation

#### 17.3.3.1 typedef struct FeatureSet_* FeatureSet

Definition at line 99 of file anamod.h.

#### 17.3.3.2 typedef struct FeatureValues_* FeatureValues

Definition at line 100 of file anamod.h.

### 17.3.4 Function Documentation

#### 17.3.4.1 PetscErrorCode AddToFeatureSet (FeatureSet *set*, char * *cat*, char * *cmp*, int * *idx*)

Add a requested feature to a featureset object. See Feature sets

Arguments:

- `set` : the featureset
- `cat`,cmp : the category and component name. It is an error to supply unknown names
- `idx` : the index under which the feature is known in this featureset. This index can be supplied to GetFeatureValue(). This parameter can be null.

Definition at line 114 of file feature.c.

References CHECKVALIDFSET, name, and StringArrayAdd().

Here is the call graph for this function:

#### 17.3.4.2 PetscErrorCode AnalysisDataTypeArrayAdd (AnalysisDataTypeArray *array*, AnalysisDataType *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 370 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, and AnalysisDataTypeArray_::has.

#### 17.3.4.3 PetscErrorCode AnalysisDataTypeArrayGetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType * *val*)

As AnalysisDataTypeArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 422 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, and AnalysisDataTypeArray_::has.

#### 17.3.4.4 PetscErrorCode AnalysisDataTypeArraySetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 386 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, and AnalysisDataTypeArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.3.4.5 PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the AnalysisDataTypeArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 408 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, and AnalysisDataTypeArray_::has.

#### 17.3.4.6 PetscErrorCode AnalysisItemArrayAdd (AnalysisItemArray *array*, AnalysisItem *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 266 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, and AnalysisItemArray_::has.

#### 17.3.4.7 PetscErrorCode AnalysisItemArrayGetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem * *val*)

As AnalysisItemArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 318 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, and AnalysisItemArray_::has.

#### 17.3.4.8 PetscErrorCode AnalysisItemArraySetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 282 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, and AnalysisItemArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.3.4.9 PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the AnalysisItemArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 304 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, and AnalysisItemArray_::has.

Referenced by GetFeatureValue().

#### 17.3.4.10 PetscErrorCode AnaModDeclareTraceContext (void *)

Definition at line 69 of file tracing.c.

References anamodtracectx.

#### 17.3.4.11 PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(*)(void *, char *, va_list) *fn*)

Specify a trace function.

The trace function has a prototype

```
PetscErrorCode tracefunction(void*,char*,va_list)
```

which means that it has an arbitrary number of arguments, much like `printf`. The first argument is a context, which can be set by AnaModDeclareTraceContext().

Here is an example of how you would write a trace function:

```
#include <stdarg.h>
PetscErrorCode tracefunction(void *ctx,char *fmt,va_list argp)
{
  char *prefix = (char*)ctx;
  PetscFunctionBegin;
```

```
  printf("%s ",prefix);
  vprintf(fmt, argp);
  PetscFunctionReturn(0);
}
```

Consult `string.h` (probably in `/usr/include`) to see which "v" versions of `printf` are available.

You can undeclare a trace function by passing NULL.

Definition at line 56 of file tracing.c.

References anamodtrace.

#### 17.3.4.12 PetscErrorCode AnaModFinalize ()

Definition at line 348 of file module_functions.c.

Referenced by main().

#### 17.3.4.13 PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(∗∗)(void ∗, char ∗, char ∗, AnalysisItem ∗, AnalysisDataType ∗, PetscTruth ∗), PetscTruth ∗)

Definition at line 229 of file feature.c.

References retriever.

Referenced by InstantiateFeatureSet().

#### 17.3.4.14 PetscErrorCode AnaModGetSequentialMatrix (Mat *A*, Mat ∗ *Awork*, PetscTruth ∗ *mem*, PetscTruth ∗ *local*, PetscTruth ∗ *global*)

Collect the matrix on processor zero.

There is an implicit assumption here that processor zero is the one that will do the actual work.

Argument:

- `A` : input matrix
- `Awork` : pointer to the sequential matrix, this can be a pointer to the original matrix if it was already sequential; it is NULL if no forced sequential computation is asked (see Commandline options)

- `mem` : true if `Awork` is newly allocated
- `local` : true if this processor needs to do the work
- `global` : true if any processor does work; this condition is false in the case of a distributed matrix and no forced sequential operation

Definition at line 154 of file options.c.

References AnaModHasForcedSequentialComputation().

Referenced by compute_tracea2(), MatCommutatorNormF(), and MatSymmPartNormInf().

Here is the call graph for this function:

#### 17.3.4.15 PetscErrorCode AnaModGetTypeMySQLName (int *id*, char ** *name*)

Definition at line 372 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, mysqlname, and nAnaModTypeNames.

Referenced by main().

#### 17.3.4.16 PetscErrorCode AnaModGetTypeName (int *id*, char ** *name*)

Definition at line 356 of file module_functions.c.

References anamodtypenames, and nAnaModTypeNames.

#### 17.3.4.17 PetscErrorCode AnaModHasForcedExpensiveComputation (PetscTruth * *flg*)

Query whether certain expensive operations should be done regardless the cost.

Definition at line 200 of file options.c.

References expensive.

Referenced by MatCommutatorNormF_seq().

#### 17.3.4.18 PetscErrorCode AnaModHasForcedSequentialComputation (PetscTruth * *flg*)

Query whether parallel modules should be done on processor zero.

Definition at line 127 of file options.c.

References single_proc.

Referenced by AnaModGetSequentialMatrix().

#### 17.3.4.19 PetscErrorCode AnaModHasTrace (PetscTruth * *flg*)

Test whether a trace function has been declared; see AnaModDeclareTraceFunction(). Normally you would use AnaModTraceMessage() which performs this test internally, but this function can be useful if a large amount of processing has to be performed to construct the trace message to begin with.

Definition at line 127 of file tracing.c.

References anamodtrace.

Referenced by ComputeQuantity(), and RetrieveQuantity().

#### 17.3.4.20 PetscErrorCode AnaModInitialize ()

Definition at line 319 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, and nAnaModTypeNames.

Referenced by main().

#### 17.3.4.21 PetscErrorCode AnaModOptionsHandling (void)

Process any commandline options that were given for AnaMod. These use the regular Petsc commandline options database.

This routine handles general options and module-specific options, so it has to be called after all modules have been declared.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 38 of file options.c.

References categories, expensive, GetCategories(), GetCategoryOptionFunction(), single_proc, and VALUELEN.

Here is the call graph for this function:

#### 17.3.4.22 PetscErrorCode AnaModRegisterStandardModules ()

Register all standard and nonstandard analysis modules.

Definition at line 44 of file anamodsalsa.c.

References RegisterIprsModules(), RegisterJPLModules(), RegisterLapackModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStructureModules(), and RegisterVarianceModules().

Here is the call graph for this function:

#### 17.3.4.23 PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode(∗)(void ∗, char ∗, char ∗, AnalysisItem ∗, AnalysisDataType ∗, PetscTruth ∗))

Definition at line 219 of file feature.c.

References retriever.

#### 17.3.4.24 PetscErrorCode AnaModSetTraceArrays (PetscTruth *f*)

Definition at line 82 of file tracing.c.

References anamodtracearrays.

#### 17.3.4.25 PetscErrorCode AnaModShowOptions (MPI_Comm *comm*)

Display all available options. This depends on the installed modules, so you need to do the various register calls first. See Use of the analysis modules.

For options see Commandline Options for Runtime Control.

Definition at line 99 of file options.c.

References categories, and GetCategories().

Here is the call graph for this function:

#### 17.3.4.26 PetscErrorCode AnaModTraceArrays (PetscTruth ∗ *f*)

Definition at line 95 of file tracing.c.

References anamodtracearrays.

Referenced by ComputeQuantity(), ReportAnamodContent(), and RetrieveQuantity().

#### 17.3.4.27 PetscErrorCode AnaModTraceMessage (const char * *fmt*, ...)

This function prints a trace message if a trace function has been declared; see AnaModDeclareTraceFunction().

Definition at line 107 of file tracing.c.

References anamodtrace, and anamodtracectx.

Referenced by compute_eigenvalues_petsc(), ComputeQuantity(), and RetrieveQuantity().

#### 17.3.4.28 PetscErrorCode CategoryGetModules (char * *cat*, char *** *ms*, AnalysisDataType ** *t*, int ** *id*, int * *n*)

Query the modules in a specified category.

The category name has to exist. The routine will call the Petsc error handler if the name is invalid.

Parameters:

- `cat` : the category that is being queried
- `ms` (optional) : the names of the modules in the category
- `t` (optional) : the corresponding list of datatypes
- `id` (optional) : the list of module IDs (see RetrieveQuantityByID())
- `n` (optional) : the number of modules in the category

See also GetCategories() and HasComputeCategory().

Definition at line 711 of file module_functions.c.

References categories, components, ids, ncategories, ncomponents, and types.

Referenced by analyze_matrix(), main(), and ReportAnamodContent().

#### 17.3.4.29 PetscErrorCode CategoryLogEventRegister (char * *cat*, int *icat*)

Definition at line 20 of file logging.c.

#### 17.3.4.30 PetscErrorCode ComputeQuantity (AnaModNumericalProblem *prob*, const char * *cat*, const char * *cmp*, AnalysisItem * *res*, int * *rreslen*, PetscTruth * *success*)

Compute a computational module from a certain category.

Argument:

1. the name of the category (see GetCategories())
2. the name of the module (see CategoryGetModules())
3. the matrix
4. a pointer to the result. This is given as `"(AnalysisItem*)&res"` ; see types for the definition of the AnalysisItem data type
5. the length of the result if the result is an array. This argument can be NULL.
6. a success indicator. Failure can have obvious causes, such as breakdown of an internal routine, but the routine can also refuse to compute a quantity if doing so would be too expensive (see an example in the Estimates for the departure from normality category).

A call to this routine need not involve actual computation: the requested quantity can already be attached to the matrix object (see attached quantities for details). This mechanism is used in all the standard modules that come with the AnaMod package.

The workings of this function can be traced by specifying a trace function; see Tracing the analysis modules.

Definition at line 866 of file module_functions.c.

References ANALYSISINTARRAY, AnaModHasTrace(), AnaModTraceArrays(), AnaModTraceMessage(), GetCategoryIndex(), GetModuleIndex(), hasval, modules, QuantityAsString(), TYPEii, and types.

Referenced by analyze_matrix(), DepartureRuhe75(), LoBand(), MatrixComputeQuantity(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), RelSymm(), and UpBand().

Here is the call graph for this function:

#### 17.3.4.31 PetscErrorCode ComputeQuantityByID (AnaModNumericalProblem, int, int, AnalysisItem *, int *, PetscTruth *)

#### 17.3.4.32 PetscErrorCode CreateAnalysisDataTypeArray (char *, int, AnalysisDataTypeArray *)

Definition at line 336 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, AnalysisDataTypeArray_::has, NALLOC, and AnalysisDataTypeArray_::name.

Referenced by NewFeatureValues().

#### 17.3.4.33 PetscErrorCode CreateAnalysisItemArray (char *, int, AnalysisItemArray *)

Definition at line 232 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, AnalysisItemArray_::has, NALLOC, and AnalysisItemArray_::name.

Referenced by NewFeatureValues().

#### 17.3.4.34 PetscErrorCode CreateIntArray (const char *, int, IntArray *)

Definition at line 13 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, IntArray_::has, NALLOC, and IntArray_::name.

Referenced by NewFeatureSet().

#### 17.3.4.35 PetscErrorCode CreateStringArray (const char *, int, StringArray *)

Definition at line 117 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, StringArray_::has, NALLOC, and StringArray_::name.

Referenced by NewFeatureSet().

#### 17.3.4.36 PetscErrorCode DeclareCategoryOptionFunction (char * *cat*, PetscErrorCode(*)(char *) *f*)

This function allows the module developer to give the user commandline options for control of a module.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 807 of file module_functions.c.

References GetCategoryIndex(), and optionfunctions.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.3.4.37 PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray)

Definition at line 354 of file anamodutils.c.

References AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::has, and AnalysisDataTypeArray_::name.

Referenced by DeleteFeatureValues().

#### 17.3.4.38 PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray)

Definition at line 250 of file anamodutils.c.

References AnalysisItemArray_::data, AnalysisItemArray_::has, and AnalysisItemArray_::name.

Referenced by DeleteFeatureValues().

#### 17.3.4.39 PetscErrorCode DeleteFeatureSet (FeatureSet *set*)

Delete a featureset object. See Feature sets

Definition at line 90 of file feature.c.

References CHECKVALIDFSET, DeleteIntArray(), and DeleteStringArray().

Here is the call graph for this function:

#### 17.3.4.40 PetscErrorCode DeleteFeatureValues (FeatureValues *values*)

Free a featurevalues object. See Feature sets

Definition at line 148 of file feature.c.

References CHECKVALIDFVAL, DeleteAnalysisDataTypeArray(), DeleteAnalysisItemArray(), FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.3.4.41 PetscErrorCode DeleteIntArray (IntArray)

Definition at line 31 of file anamodutils.c.

References IntArray_::data, IntArray_::has, and IntArray_::name.

Referenced by DeleteFeatureSet().

#### 17.3.4.42 PetscErrorCode DeleteStringArray (StringArray)

Definition at line 135 of file anamodutils.c.

References StringArray_::data, StringArray_::fill, StringArray_::has, and StringArray_::name.

Referenced by DeleteFeatureSet().

#### 17.3.4.43 PetscErrorCode DeRegisterCategory (char * *cat*)

Deallocate the storage for a particular category of analysis modules

Definition at line 612 of file module_functions.c.

References categories, components, hasval, ids, modules, ncategories, ncomponents, and types.

Referenced by DeregisterModules().

#### 17.3.4.44 PetscErrorCode DeregisterModules (void)

Definition at line 642 of file module_functions.c.

References categories, components, DeRegisterCategory(), GetCategories(), hasval, ids, maxcomponents, modules, ncategories, ncomponents, optionfunctions, and types.

Here is the call graph for this function:

#### 17.3.4.45 PetscErrorCode GetCategories (char *** *cats*, int * *n*)

Query the number of defined categories and their names.

Parameters:

- `cats` (optional) output: list of category names
- `n` output: number of currently installed categories

See also CategoryGetModules().

Definition at line 687 of file module_functions.c.

References categories, and ncategories.

Referenced by analyze_matrix(), AnaModOptionsHandling(), AnaModShowOptions(), DeregisterModules(), main(), and ReportAnamodContent().

#### 17.3.4.46 PetscErrorCode GetCategoryOptionFunction (char * *cat*, PetscErrorCode(**)(char *) *f*)

This function is called in AnaModOptionsHandling(). There is probably no reason for the user ever to call it.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 827 of file module_functions.c.

References GetCategoryIndex(), and optionfunctions.

Referenced by AnaModOptionsHandling().

Here is the call graph for this function:

#### 17.3.4.47 PetscErrorCode GetDataID (const char * *cat*, const char * *cmp*, int * *id*, PetscTruth * *flg*)

Get the numerical id under which a module is stored.

The flag parameter returns false if the category or cat/cmp does not exist. The flag parameter is optional, but the routine will exit if the cat/cmp is not found and the flag parameter is not provided.

This is not a user level function.

Definition at line 1164 of file module_functions.c.

References GetCategoryIndex(), GetModuleIndex(), and ids.

Referenced by AvgDiagDist(), AvgDistFromDiag(), AvgNnzpRow(), BlockSize(), ColourOffsets(), Colours(), ColourSizes(), ColVariability(), Commutator(), compute_dd(), compute_dummy_rows(), compute_eigenvalues(), compute_ellipse_from_Ritz_values(), compute_icm_splits(), compute_nnz_structure(), compute_posdiag(), compute_singularvalues(), compute_tracea2(), ComputeDiagonal(), computennz(), computetrace(), ComputeVariability(), Departure(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), DiagDefinite(), DiagonalAverage(), DiagonalDominance(), DiagonalSign(), DiagonalVariance(), DiagZeroStart(), DummyRows(), DummyRowsKind(), eigenvaluecomp(), HasQuantity(), JonesPlassmannColouring(), Kappa(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUnstruct(), PosFraction(), RBandWidth(), regularblocks(), RelSymm(), RightSkyline(), RitzValuesC(), RitzValuesR(), RowVariability(), SigmaDiagDist(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), Splits(), Symmetry(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), TraceAbs(), UpBand(), and Version().

Here is the call graph for this function:

#### 17.3.4.48 PetscErrorCode GetDataType (const char * *cat*, const char * *cmp*, AnalysisDataType * *t*)

Retrieve the `AnalysisDataType` of a computational module

Definition at line 1191 of file module_functions.c.

References GetCategoryIndex(), GetModuleIndex(), and TYPEii.

Referenced by analyze_matrix(), and HasQuantity().

Here is the call graph for this function:

#### 17.3.4.49 PetscErrorCode GetFeatureValue (FeatureValues *values*, int *index*, AnalysisItem * *val*, PetscTruth * *f*)

Extract a value from a featurevalues object. See Feature sets.

Arguments:

- `values` : the FeatureValues object.
- `index` : the index, as returned by AddToFeatureSet().
- `val` (output) : the value; this argument can be null.
- `f` (output) : indicates whether the return value was indeed preset in the featurevalues object; this argument can be null.

Definition at line 206 of file feature.c.

References AnalysisItemArrayTryGetAt(), CHECKVALIDFVAL, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.3.4.50 PetscErrorCode HasComputeCategory (char * *cat*, PetscTruth * *f*)

Query whether a specified category has been declared.

Definition at line 773 of file module_functions.c.

References GetCategoryIndex().

Here is the call graph for this function:

#### 17.3.4.51 PetscErrorCode HasComputeModule (char * *cat*, char * *cmp*, PetscTruth * *f*)

Query whether a specified module exists inside a specified category. The category need not itself have been declared.

Definition at line 787 of file module_functions.c.

References GetCategoryIndex(), and GetModuleIndex().

Referenced by LoBand(), and UpBand().

Here is the call graph for this function:

#### 17.3.4.52 PetscErrorCode HasQuantity (AnaModNumericalProblem *prob*, char * *cat*, char * *cmp*, PetscTruth * *f*)

Check if a certain quantity is precomputed, meaning that it can be retrieved (with a call to ComputeQuantity()) at no computational cost.

The category and module names have to exist. The routine will call the Petsc error handler if either name is invalid. Use HasComputeModule() to test whether a category and module is known to the system.

See the page on attached quantities for an explanation of the mechanism behind this routine.

Definition at line 1017 of file module_functions.c.

References GetDataID(), GetDataType(), and HasQuantityByID().

Referenced by computennz().

Here is the call graph for this function:

#### 17.3.4.53 PetscErrorCode HasQuantityByID (AnaModNumericalProblem *prob*, int *id*, AnalysisDataType *type*, PetscTruth * *f*)

Auxiliary routine with lookup by ID, which is much faster than by string indexing.

Definition at line 1035 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, and ANALYSISINTEGER.

Referenced by HasQuantity().

#### 17.3.4.54 PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem *prob*, FeatureSet *set*, FeatureValues *values*)

Fill in a featurevalues object. See Feature sets

Definition at line 163 of file feature.c.

References AnalysisDataTypeArraySetAt(), AnalysisItemArraySetAt(), AnaModGetRetrievalFunction(), CHECKVALIDFSET, CHECKVALIDFVAL, retriever, StringArrayGetAt(), StringArrayGetFill(), FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.3.4.55 PetscErrorCode IntArrayAdd (IntArray *array*, int *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 47 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, and IntArray_::has.

#### 17.3.4.56 PetscErrorCode IntArrayGetAt (IntArray *array*, int *idx*, int * *val*)

As IntArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 99 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, and IntArray_::has.

#### 17.3.4.57 PetscErrorCode IntArrayGetFill (IntArray, int *)

#### 17.3.4.58 PetscErrorCode IntArraySetAt (IntArray *array*, int *idx*, int *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 63 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, and IntArray_::has.

#### 17.3.4.59 PetscErrorCode IntArrayTryGetAt (IntArray *array*, int *idx*, int * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the IntArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 85 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, and IntArray_::has.

#### 17.3.4.60 PetscErrorCode NewFeatureSet (FeatureSet * *set*)

Allocate a featureset object. See Feature sets

Definition at line 74 of file feature.c.

References FeatureSet_::cookie, CreateIntArray(), CreateStringArray(), FeatureSet_-::features, FSETCOOKIE, and FeatureSet_::IDs.

Here is the call graph for this function:

#### 17.3.4.61 PetscErrorCode NewFeatureValues (FeatureValues * *values*)

Allocate a featurevalues object. See Feature sets

Definition at line 132 of file feature.c.

References FeatureValues_::cookie, CreateAnalysisDataTypeArray(), CreateAnalysisItemArray(), FVALCOOKIE, FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.3.4.62 PetscErrorCode PurgeAttachedArrays (Mat *A*)

#### 17.3.4.63 PetscErrorCode QuantityAsString (AnalysisItem * *q*, AnalysisDataType *t*, char ** *s*)

Generate a character string for a given quantity.

Definition at line 1101 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, ANALYSISSTRING, AnalysisItem::c, AnalysisItem::i, AnalysisItem::ii, AnalysisItem::len, AnalysisItem::r, and AnalysisItem::rr.

Referenced by ComputeQuantity(), ReportAnamodContent(), and RetrieveQuantity().

#### 17.3.4.64 PetscErrorCode RegisterModule (const char * *cat*, const char * *cmp*, AnalysisDataType *type*, PetscErrorCode(*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *) *f*)

Register a new computational module

This adds a computational routine (the `f` argument) into the modules database under

the given category (cat) and module (cmp) label. If the category does not exist yet, it is created.

The available types are defined in anamodtypes.h

If the routine is NULL, only the category and component are created.

Routine prototype:

- problem (input)
- item (output)
- array length (output)
- success (output)

See also HasComputeModule(), ComputeQuantity(), DeregisterModules().

Definition at line 410 of file module_functions.c.

References categories, components, hasval, id, ids, INC, maxcategories, maxcomponents, modules, MXC, ncategories, ncomponents, optionfunctions, and types.

Referenced by RegisterICMKModules(), RegisterIprsModules(), RegisterJPLModules(), RegisterLapackModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStatsModules(), RegisterStructureModules(), and RegisterVarianceModules().

#### 17.3.4.65 PetscErrorCode RetrieveQuantity (AnaModNumericalProblem *prob*, char * *cat*, char * *cmp*, AnalysisItem * *res*, PetscTruth * *success*)

Retrieve an attached quantity. Note that this does not report the length of arrays; you have to know under which name this is stored.

Definition at line 931 of file module_functions.c.

References ANALYSISINTARRAY, AnaModHasTrace(), AnaModTraceArrays(), AnaModTraceMessage(), GetCategoryIndex(), GetModuleIndex(), hasval, QuantityAsString(), RetrieveQuantityByID(), TYPEii, and types.

Here is the call graph for this function:

#### 17.3.4.66 PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem *prob*, int *icat*, int *icmp*, AnalysisItem * *result*, PetscTruth * *f*)

See also HasQuantityByID()

Definition at line 1066 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, AnalysisItem::i, ids, AnalysisItem::ii, AnalysisItem::r, AnalysisItem::rr, and TYPEii.

Referenced by RetrieveQuantity().

#### 17.3.4.67 PetscErrorCode StringArrayAdd (StringArray *array*, char * *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 153 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, and StringArray_::has.

Referenced by AddToFeatureSet().

#### 17.3.4.68 PetscErrorCode StringArrayGetAt (StringArray *array*, int *idx*, char ** *val*)

As StringArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 214 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, and StringArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.3.4.69 PetscErrorCode StringArrayGetFill (StringArray, int *)

Definition at line 166 of file anamodutils.c.

References StringArray_::fill.

Referenced by InstantiateFeatureSet().

#### 17.3.4.70 PetscErrorCode StringArraySetAt (StringArray *array*, int *idx*, char * *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 178 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, and StringArray_::has.

#### 17.3.4.71 PetscErrorCode StringArrayTryGetAt (StringArray *array*, int *idx*, char ** *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the StringArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported

- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 200 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, and StringArray_::has.

#### 17.3.4.72 PetscErrorCode TabReportModules (Mat, char **, char **, int)

## 17.4 anamodsalsa.c File Reference

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
```

Include dependency graph for anamodsalsa.c:

### Functions

- PetscErrorCode AnaModRegisterSalsaModules ()
- PetscErrorCode AnaModDeregisterSalsaModules ()
- PetscErrorCode AnaModRegisterStandardModules ()

### 17.4.1 Function Documentation

#### 17.4.1.1 PetscErrorCode AnaModDeregisterSalsaModules (void)

Definition at line 27 of file anamodsalsa.c.

References DeRegisterSimpleModules(), DeregisterSpectrumModules(), DeRegisterStructureModules(), and DeRegisterVarianceModules().

Referenced by main().

Here is the call graph for this function:

#### 17.4.1.2 PetscErrorCode AnaModRegisterSalsaModules (void)

Definition at line 7 of file anamodsalsa.c.

References CommutatorNormFAllowSqrtTimes(), RegisterIprsModules(), RegisterJPLModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStructureModules(), RegisterVarianceModules(), and SpectrumComputeUnpreconditionedSpectrum().

Referenced by main().

Here is the call graph for this function:

#### 17.4.1.3 PetscErrorCode AnaModRegisterStandardModules ()

Register all standard and nonstandard analysis modules.

Definition at line 44 of file anamodsalsa.c.

References RegisterIprsModules(), RegisterJPLModules(), RegisterLapackModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStructureModules(), and RegisterVarianceModules().

Here is the call graph for this function:

## 17.5 anamodsalsamodules.h File Reference

Prototypes for using the standard modules.

```
#include "petsc.h"
```

Include dependency graph for anamodsalsamodules.h:

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

### Defines

- #define DIAGONAL_POSITIVE 2
- #define DIAGONAL_NONNEGATIVE 1
- #define DIAGONAL_INDEFINITE 0
- #define DIAGONAL_NONPOSITIVE -1
- #define DIAGONAL_NEGATIVE -2
- #define DIAGONAL_ZERO 10
- #define DIAGONAL_INDEFINITE_WITH_ZEROS 11

### Functions

- PetscErrorCode RegisterSimpleModules (void)
- PetscErrorCode DeRegisterSimpleModules (void)
- PetscErrorCode RegisterVarianceModules (void)
- PetscErrorCode DeRegisterVarianceModules (void)
- PetscErrorCode RegisterNormalityModules (void)
- PetscErrorCode CommutatorNormFAllowSqrtTimes (int n)
- PetscErrorCode RegisterStructureModules (void)
- PetscErrorCode DeRegisterStructureModules (void)
- PetscErrorCode RegisterSpectrumModules (void)
- PetscErrorCode DeregisterSpectrumModules (void)

- PetscErrorCode SpectrumComputePreconditionedSpectrum (void)
- PetscErrorCode SpectrumComputeUnpreconditionedSpectrum (void)
- PetscErrorCode RegisterJPLModules (void)
- PetscErrorCode RegisterIprsModules (void)
- PetscErrorCode AnaModRegisterSalsaModules (void)
- PetscErrorCode AnaModDeregisterSalsaModules (void)
- PetscErrorCode RegisterLapackModules ()

### 17.5.1 Detailed Description

Prototypes for using the standard modules.

Definition in file anamodsalsamodules.h.

### 17.5.2 Define Documentation

#### 17.5.2.1 #define DIAGONAL_INDEFINITE 0

Definition at line 38 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

#### 17.5.2.2 #define DIAGONAL_INDEFINITE_WITH_ZEROS 11

Definition at line 42 of file anamodsalsamodules.h.

#### 17.5.2.3 #define DIAGONAL_NEGATIVE -2

Definition at line 40 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

#### 17.5.2.4 #define DIAGONAL_NONNEGATIVE 1

Definition at line 37 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

#### 17.5.2.5 #define DIAGONAL_NONPOSITIVE -1

Definition at line 39 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

#### 17.5.2.6 #define DIAGONAL_POSITIVE 2

Definition at line 36 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

#### 17.5.2.7 #define DIAGONAL_ZERO 10

Definition at line 41 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

### 17.5.3 Function Documentation

#### 17.5.3.1 PetscErrorCode AnaModDeregisterSalsaModules (void)

Definition at line 27 of file anamodsalsa.c.

References DeRegisterSimpleModules(), DeregisterSpectrumModules(), DeRegisterStructureModules(), and DeRegisterVarianceModules().

Referenced by main().

Here is the call graph for this function:

#### 17.5.3.2 PetscErrorCode AnaModRegisterSalsaModules (void)

Definition at line 7 of file anamodsalsa.c.

References CommutatorNormFAllowSqrtTimes(), RegisterIprsModules(), RegisterJPLModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStructureModules(), RegisterVarianceModules(), and SpectrumComputeUnpreconditionedSpectrum().

Referenced by main().

Here is the call graph for this function:

#### 17.5.3.3 PetscErrorCode CommutatorNormFAllowSqrtTimes (int *n*)

Definition at line 281 of file normal.c.

Referenced by AnaModRegisterSalsaModules().

#### 17.5.3.4 PetscErrorCode DeRegisterSimpleModules (void)

Definition at line 681 of file simple.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.5.3.5 PetscErrorCode DeregisterSpectrumModules (void)

Definition at line 1401 of file spectrum.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.5.3.6 PetscErrorCode DeRegisterStructureModules (void)

Definition at line 738 of file structure.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.5.3.7 PetscErrorCode DeRegisterVarianceModules (void)

Definition at line 333 of file variance.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.5.3.8 PetscErrorCode RegisterIprsModules (void)

Definition at line 524 of file iprs.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, AvgDiagDist(), AvgNnzpRow(), LoBand(), NDiags(), Nnz(), NnzDia(), NnzLow(), NnzUp(),

RegisterModule(), RelSymm(), SigmaDiagDist(), and UpBand().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.9 PetscErrorCode RegisterJPLModules (void)

Definition at line 523 of file jpl.c.

References ANALYSISINTARRAY, ANALYSISINTEGER, ColourOffsets(), Colours(), ColourSizes(), NColours(), and RegisterModule().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.10 PetscErrorCode RegisterLapackModules ()

Declare norm-like modules that can be performed with lapack calculations.

Definition at line 424 of file lapack.c.

References ANALYSISDOUBLE, Departure(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), and RegisterModule().

Referenced by AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.11 PetscErrorCode RegisterNormalityModules (void)

Definition at line 581 of file normal.c.

References ANALYSISDOUBLE, Commutator(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), RegisterModule(), and TraceA2().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.12 PetscErrorCode RegisterSimpleModules (void)

Declare norm-like modules that can be performed with simple calculations.

Definition at line 644 of file simple.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, DiagonalDominance(), norm1(), normF(), normInf(), NUnstruct(), RegisterModule(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), and TraceAbs().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.13 PetscErrorCode RegisterSpectrumModules (void)

Definition at line 1329 of file spectrum.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTEGER, DeclareCategoryOptionFunction(), Kappa(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), NRitzValues(), PosFraction(), RegisterModule(), RitzValuesC(), RitzValuesR(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), and SpectrumOptions().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.14 PetscErrorCode RegisterStructureModules (void)

Definition at line 689 of file structure.c.

References ANALYSISINTARRAY, ANALYSISINTEGER, BlockSize(), DiagDefinite(), DiagZeroStart(), DummyRows(), DummyRowsKind(), LBandWidth(), LeftSkyline(), MaxNNonZerosPerRow(), MinNNonZerosPerRow(), NDummyRows(), NNonZeros(), nRows(), RBandWidth(), RegisterModule(), RightSkyline(), and Symmetry().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.15 PetscErrorCode RegisterVarianceModules (void)

Definition at line 311 of file variance.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, ColVariability(), DiagonalAverage(), DiagonalSign(), DiagonalVariance(), RegisterModule(), and RowVariability().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.5.3.16 PetscErrorCode SpectrumComputePreconditionedSpectrum (void)

Definition at line 94 of file spectrum.c.

References use_prec.

#### 17.5.3.17 PetscErrorCode SpectrumComputeUnpreconditionedSpectrum (void)

Definition at line 103 of file spectrum.c.

References use_prec.

Referenced by AnaModRegisterSalsaModules().

## 17.6 anamodtypes.h File Reference

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

### Data Structures

- union AnalysisItem

### Defines

- #define ANALYSISNONE 1
- #define ANALYSISSTRING 2
- #define ANALYSISINTEGER 3
- #define ANALYSISDOUBLE 4
- #define ANALYSISFLOAT 5
- #define ANALYSISMETA 6
- #define ANALYSISVOID 7
- #define ANALYSISINTARRAY 8
- #define ANALYSISDBLARRAY 9
- #define ANALYSISFLARRAY 10

### Typedefs

- typedef void * AnaModNumericalProblem
- typedef struct IntArray_ * IntArray
- typedef struct StringArray_ * StringArray
- typedef struct AnalysisItemArray_ * AnalysisItemArray
- typedef struct AnalysisDataTypeArray_ * AnalysisDataTypeArray
- typedef int AnalysisDataType

### 17.6.1 Detailed Description

This file lists the possible return types of analysis modules. See typehandling about semantic issues.

Definition in file anamodtypes.h.

### 17.6.2 Define Documentation

#### 17.6.2.1 #define ANALYSISDBLARRAY 9

Definition at line 56 of file anamodtypes.h.

Referenced by HasQuantityByID(), QuantityAsString(), RegisterSpectrumModules(), and RetrieveQuantityByID().

#### 17.6.2.2 #define ANALYSISDOUBLE 4

Definition at line 51 of file anamodtypes.h.

Referenced by HasQuantityByID(), QuantityAsString(), RegisterIprsModules(), RegisterLapackModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterVarianceModules(), and RetrieveQuantityByID().

#### 17.6.2.3 #define ANALYSISFLARRAY 10

Definition at line 57 of file anamodtypes.h.

#### 17.6.2.4 #define ANALYSISFLOAT 5

Definition at line 52 of file anamodtypes.h.

#### 17.6.2.5 #define ANALYSISINTARRAY 8

Definition at line 55 of file anamodtypes.h.

Referenced by ComputeQuantity(), HasQuantityByID(), QuantityAsString(), RegisterICMKModules(), RegisterJPLModules(), RegisterStructureModules(), RetrieveQuantity(), and RetrieveQuantityByID().

#### 17.6.2.6 #define ANALYSISINTEGER 3

Definition at line 50 of file anamodtypes.h.

Referenced by HasQuantityByID(), QuantityAsString(), RegisterICMKModules(), RegisterIprsModules(), RegisterJPLModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStructureModules(), RegisterVarianceModules(), and RetrieveQuantityByID().

#### 17.6.2.7 #define ANALYSISMETA 6

Definition at line 53 of file anamodtypes.h.

#### 17.6.2.8 #define ANALYSISNONE 1

Definition at line 48 of file anamodtypes.h.

#### 17.6.2.9 #define ANALYSISSTRING 2

Definition at line 49 of file anamodtypes.h.

Referenced by QuantityAsString(), and RegisterStatsModules().

#### 17.6.2.10 #define ANALYSISVOID 7

Definition at line 54 of file anamodtypes.h.

### 17.6.3 Typedef Documentation

#### 17.6.3.1 typedef int AnalysisDataType

Definition at line 46 of file anamodtypes.h.

#### 17.6.3.2 typedef struct AnalysisDataTypeArray_* AnalysisDataTypeArray

Definition at line 22 of file anamodtypes.h.

#### 17.6.3.3 typedef struct AnalysisItemArray_* AnalysisItemArray

Definition at line 21 of file anamodtypes.h.

#### 17.6.3.4 typedef void* AnaModNumericalProblem

Definition at line 10 of file anamodtypes.h.

#### 17.6.3.5 typedef struct IntArray_* IntArray

Definition at line 19 of file anamodtypes.h.

#### 17.6.3.6 typedef struct StringArray_* StringArray

Definition at line 20 of file anamodtypes.h.

## 17.7 anamodutils.c File Reference

```
#include <stdlib.h>
#include "string.h"
#include "anamod.h"
```

Include dependency graph for anamodutils.c:

### Data Structures

- struct IntArray_

- struct StringArray_
- struct AnalysisItemArray_
- struct AnalysisDataTypeArray_

### Defines

- #define NALLOC 70

### Functions

- PetscErrorCode CreateIntArray (const char *name, int size, IntArray *array)
- PetscErrorCode DeleteIntArray (IntArray array)
- PetscErrorCode IntArrayAdd (IntArray array, int val, int *idx)
- PetscErrorCode IntArraySetAt (IntArray array, int idx, int val)
- PetscErrorCode IntArrayTryGetAt (IntArray array, int idx, int *val, PetscTruth *has)
- PetscErrorCode IntArrayGetAt (IntArray array, int idx, int *val)
- PetscErrorCode CreateStringArray (const char *name, int size, StringArray *array)
- PetscErrorCode DeleteStringArray (StringArray array)
- PetscErrorCode StringArrayAdd (StringArray array, char *val, int *idx)
- PetscErrorCode StringArrayGetFill (StringArray array, int *idx)
- PetscErrorCode StringArraySetAt (StringArray array, int idx, char *val)
- PetscErrorCode StringArrayTryGetAt (StringArray array, int idx, char **val, PetscTruth *has)
- PetscErrorCode StringArrayGetAt (StringArray array, int idx, char **val)
- PetscErrorCode CreateAnalysisItemArray (char *name, int size, AnalysisItemArray *array)
- PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray array)
- PetscErrorCode AnalysisItemArrayAdd (AnalysisItemArray array, AnalysisItem val, int *idx)
- PetscErrorCode AnalysisItemArraySetAt (AnalysisItemArray array, int idx, AnalysisItem val)
- PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray array, int idx, AnalysisItem *val, PetscTruth *has)
- PetscErrorCode AnalysisItemArrayGetAt (AnalysisItemArray array, int idx, AnalysisItem *val)
- PetscErrorCode CreateAnalysisDataTypeArray (char *name, int size, AnalysisDataTypeArray *array)
- PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray array)
- PetscErrorCode AnalysisDataTypeArrayAdd (AnalysisDataTypeArray array, AnalysisDataType val, int *idx)

- PetscErrorCode AnalysisDataTypeArraySetAt (AnalysisDataTypeArray array, int idx, AnalysisDataType val)
- PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray array, int idx, AnalysisDataType *val, PetscTruth *has)
- PetscErrorCode AnalysisDataTypeArrayGetAt (AnalysisDataTypeArray array, int idx, AnalysisDataType *val)

### 17.7.1 Define Documentation

#### 17.7.1.1 #define NALLOC 70

Definition at line 5 of file anamodutils.c.

Referenced by CreateAnalysisDataTypeArray(), CreateAnalysisItemArray(), CreateIntArray(), and CreateStringArray().

### 17.7.2 Function Documentation

#### 17.7.2.1 PetscErrorCode AnalysisDataTypeArrayAdd (AnalysisDataTypeArray *array*, AnalysisDataType *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 370 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, and AnalysisDataTypeArray_::has.

#### 17.7.2.2 PetscErrorCode AnalysisDataTypeArrayGetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType * *val*)

As AnalysisDataTypeArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 422 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, and AnalysisDataTypeArray_::has.

#### 17.7.2.3 PetscErrorCode AnalysisDataTypeArraySetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 386 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, and AnalysisDataTypeArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.7.2.4 PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray *array*, int *idx*, AnalysisDataType * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the AnalysisDataTypeArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 408 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, and AnalysisDataTypeArray_::has.

#### 17.7.2.5 PetscErrorCode AnalysisItemArrayAdd (AnalysisItemArray *array*, AnalysisItem *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 266 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, and AnalysisItemArray_::has.

#### 17.7.2.6 PetscErrorCode AnalysisItemArrayGetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem * *val*)

As AnalysisItemArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 318 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, and AnalysisItemArray_::has.

#### 17.7.2.7 PetscErrorCode AnalysisItemArraySetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 282 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, and AnalysisItemArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.7.2.8 PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray *array*, int *idx*, AnalysisItem * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the AnalysisItemArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 304 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, and AnalysisItemArray_::has.

Referenced by GetFeatureValue().

#### 17.7.2.9 PetscErrorCode CreateAnalysisDataTypeArray (char * *name*, int *size*, AnalysisDataTypeArray * *array*)

Definition at line 336 of file anamodutils.c.

References AnalysisDataTypeArray_::alloc, AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::fill, AnalysisDataTypeArray_::has, NALLOC, and AnalysisDataTypeArray_::name.

Referenced by NewFeatureValues().

#### 17.7.2.10 PetscErrorCode CreateAnalysisItemArray (char * *name*, int *size*, AnalysisItemArray * *array*)

Definition at line 232 of file anamodutils.c.

References AnalysisItemArray_::alloc, AnalysisItemArray_::data, AnalysisItemArray_::fill, AnalysisItemArray_::has, NALLOC, and AnalysisItemArray_::name.

Referenced by NewFeatureValues().

#### 17.7.2.11 PetscErrorCode CreateIntArray (const char * *name*, int *size*, IntArray * *array*)

Definition at line 13 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, IntArray_::has, NALLOC, and IntArray_::name.

Referenced by NewFeatureSet().

#### 17.7.2.12 PetscErrorCode CreateStringArray (const char * *name*, int *size*, StringArray * *array*)

Definition at line 117 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, StringArray_-::has, NALLOC, and StringArray_::name.

Referenced by NewFeatureSet().

#### 17.7.2.13 PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray *array*)

Definition at line 354 of file anamodutils.c.

References AnalysisDataTypeArray_::data, AnalysisDataTypeArray_::has, and AnalysisDataTypeArray_::name.

Referenced by DeleteFeatureValues().

#### 17.7.2.14 PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray *array*)

Definition at line 250 of file anamodutils.c.

References AnalysisItemArray_::data, AnalysisItemArray_::has, and AnalysisItemArray_::name.

Referenced by DeleteFeatureValues().

#### 17.7.2.15 PetscErrorCode DeleteIntArray (IntArray *array*)

Definition at line 31 of file anamodutils.c.

References IntArray_::data, IntArray_::has, and IntArray_::name.

Referenced by DeleteFeatureSet().

#### 17.7.2.16 PetscErrorCode DeleteStringArray (StringArray *array*)

Definition at line 135 of file anamodutils.c.

References StringArray_::data, StringArray_::fill, StringArray_::has, and StringArray_::name.

Referenced by DeleteFeatureSet().

#### 17.7.2.17 PetscErrorCode IntArrayAdd (IntArray *array*, int *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 47 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, and IntArray_::has.

#### 17.7.2.18 PetscErrorCode IntArrayGetAt (IntArray *array*, int *idx*, int * *val*)

As IntArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 99 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, and IntArray_::has.

#### 17.7.2.19 PetscErrorCode IntArraySetAt (IntArray *array*, int *idx*, int *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 63 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, IntArray_::fill, and IntArray_::has.

#### 17.7.2.20 PetscErrorCode IntArrayTryGetAt (IntArray *array*, int *idx*, int * *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the IntArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 85 of file anamodutils.c.

References IntArray_::alloc, IntArray_::data, and IntArray_::has.

#### 17.7.2.21 PetscErrorCode StringArrayAdd (StringArray *array*, char * *val*, int * *idx*)

Add a new value and return the index. Right now we do not yet dynamically reallocate the array if there is no more space.

Definition at line 153 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, and StringArray_::has.

Referenced by AddToFeatureSet().

#### 17.7.2.22 PetscErrorCode StringArrayGetAt (StringArray *array*, int *idx*, char ** *val*)

As StringArrayTryGetAt(), except that it is an error to ask for an index where no value has been set.

Definition at line 214 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, and StringArray_::has.

Referenced by InstantiateFeatureSet().

#### 17.7.2.23 PetscErrorCode StringArrayGetFill (StringArray *array*, int * *idx*)

Definition at line 166 of file anamodutils.c.

References StringArray_::fill.

Referenced by InstantiateFeatureSet().

#### 17.7.2.24 PetscErrorCode StringArraySetAt (StringArray *array*, int *idx*, char * *val*)

Set a value at a given index. Right now we do not yet dynamically reallocate the array if the index is out of bound.

Definition at line 178 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, StringArray_::fill, and StringArray_::has.

#### 17.7.2.25 PetscErrorCode StringArrayTryGetAt (StringArray *array*, int *idx*, char ** *val*, PetscTruth * *has*)

Retrieve data from a given index.

Arguments:

- `array` : the StringArray object
- `idx` : the index; it is an error to ask out of bounds, but not to ask beyond the highest position filled; in that case failure will be reported
- `val` (output) : the value retrieved. This argument is allowed to be null.
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 200 of file anamodutils.c.

References StringArray_::alloc, StringArray_::data, and StringArray_::has.

## 17.8 anamodutils.h File Reference

```
#include <stdlib.h>
#include "petsc.h"
```

Include dependency graph for anamodutils.h:

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

**Functions**

- int MPIAllGatherIntV (int ∗array, int n, int ∗∗Array, int ∗N, MPI_Comm comm)

### 17.8.1 Function Documentation

#### 17.8.1.1 int MPIAllGatherIntV (int ∗ *array*, int *n*, int ∗∗ *Array*, int ∗ *N*, MPI_Comm *comm*)

Definition at line 8 of file utils.c.

Referenced by GetUpBiDiagSplits(), and MatSplitPoints().

## 17.9 feature.c File Reference

```
#include <stdlib.h>
#include "string.h"
#include "anamod.h"
#include "petsc.h"
```

Include dependency graph for feature.c:

### Data Structures

- struct FeatureSet_
- struct FeatureValues_

### Defines

- #define NALLOC 25
- #define FSETCOOKIE 9876
- #define CHECKVALIDFSET(i) {ANAMODCHECKVALID(i,FSETCOOKIE,"feature set");}
- #define FVALCOOKIE 9877
- #define CHECKVALIDFVAL(i) {ANAMODCHECKVALID(i,FVALCOOKIE,"feature values");}

### Functions

- PetscErrorCode NewFeatureSet (FeatureSet *set)
- PetscErrorCode DeleteFeatureSet (FeatureSet set)
- PetscErrorCode AddToFeatureSet (FeatureSet set, char *cat, char *cmp, int *idx)
- PetscErrorCode NewFeatureValues (FeatureValues *values)
- PetscErrorCode DeleteFeatureValues (FeatureValues values)
- PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem prob, FeatureSet set, FeatureValues values)
- PetscErrorCode GetFeatureValue (FeatureValues values, int index, AnalysisItem *val, PetscTruth *f)
- PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode(*fun)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *))
- PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(**fun)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *), PetscTruth *flg)

### Variables

- static PetscErrorCode(* retriever )(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *) = NULL

### 17.9.1 Define Documentation

#### 17.9.1.1 #define CHECKVALIDFSET(i) {ANAMODCHECK-VALID(i,FSETCOOKIE,"feature set");}

Definition at line 59 of file feature.c.

Referenced by AddToFeatureSet(), DeleteFeatureSet(), and InstantiateFeatureSet().

#### 17.9.1.2 #define CHECKVALIDFVAL(i) {ANAMODCHECK-VALID(i,FVALCOOKIE,"feature values");}

Definition at line 65 of file feature.c.

Referenced by DeleteFeatureValues(), GetFeatureValue(), and InstantiateFeatureSet().

#### 17.9.1.3 #define FSETCOOKIE 9876

Definition at line 58 of file feature.c.

Referenced by NewFeatureSet().

#### 17.9.1.4 #define FVALCOOKIE 9877

Definition at line 64 of file feature.c.

Referenced by NewFeatureValues().

#### 17.9.1.5 #define NALLOC 25

Definition at line 57 of file feature.c.

### 17.9.2 Function Documentation

#### 17.9.2.1 PetscErrorCode AddToFeatureSet (FeatureSet *set*, char * *cat*, char * *cmp*, int * *idx*)

Add a requested feature to a featureset object. See Feature sets

Arguments:

- `set` : the featureset
- `cat`,cmp : the category and component name. It is an error to supply unknown names
- `idx` : the index under which the feature is known in this featureset. This index can be supplied to GetFeatureValue(). This parameter can be null.

Definition at line 114 of file feature.c.

References CHECKVALIDFSET, name, and StringArrayAdd().

Here is the call graph for this function:

#### 17.9.2.2 PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(**)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *) *fun*, PetscTruth * *flg*)

Definition at line 229 of file feature.c.

References retriever.

Referenced by InstantiateFeatureSet().

#### 17.9.2.3 PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode(*)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *) *fun*)

Definition at line 219 of file feature.c.

References retriever.

#### 17.9.2.4 PetscErrorCode DeleteFeatureSet (FeatureSet *set*)

Delete a featureset object. See Feature sets

Definition at line 90 of file feature.c.

References CHECKVALIDFSET, DeleteIntArray(), and DeleteStringArray().

Here is the call graph for this function:

#### 17.9.2.5 PetscErrorCode DeleteFeatureValues (FeatureValues *values*)

Free a featurevalues object. See Feature sets

Definition at line 148 of file feature.c.

References CHECKVALIDFVAL, DeleteAnalysisDataTypeArray(), DeleteAnalysisItemArray(), FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.9.2.6 PetscErrorCode GetFeatureValue (FeatureValues *values*, int *index*, AnalysisItem * *val*, PetscTruth * *f*)

Extract a value from a featurevalues object. See Feature sets.

Arguments:

- `values` : the FeatureValues object.
- `index` : the index, as returned by AddToFeatureSet().

- `val` (output) : the value; this argument can be null.
- `f` (output) : indicates whether the return value was indeed preset in the featurevalues object; this argument can be null.

Definition at line 206 of file feature.c.

References AnalysisItemArrayTryGetAt(), CHECKVALIDFVAL, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.9.2.7 PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem *prob*, FeatureSet *set*, FeatureValues *values*)

Fill in a featurevalues object. See Feature sets

Definition at line 163 of file feature.c.

References AnalysisDataTypeArraySetAt(), AnalysisItemArraySetAt(), AnaModGetRetrievalFunction(), CHECKVALIDFSET, CHECKVALIDFVAL, retriever, StringArrayGetAt(), StringArrayGetFill(), FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

#### 17.9.2.8 PetscErrorCode NewFeatureSet (FeatureSet * *set*)

Allocate a featureset object. See Feature sets

Definition at line 74 of file feature.c.

References FeatureSet_::cookie, CreateIntArray(), CreateStringArray(), FeatureSet_::features, FSETCOOKIE, and FeatureSet_::IDs.

Here is the call graph for this function:

#### 17.9.2.9 PetscErrorCode NewFeatureValues (FeatureValues * *values*)

Allocate a featurevalues object. See Feature sets

Definition at line 132 of file feature.c.

References FeatureValues_::cookie, CreateAnalysisDataTypeArray(), CreateAnalysisItemArray(), FVALCOOKIE, FeatureValues_::types, and FeatureValues_::values.

Here is the call graph for this function:

### 17.9.3 Variable Documentation

#### 17.9.3.1 PetscErrorCode(* retriever)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscTruth *) = NULL [static]

Referenced by AnaModGetRetrievalFunction(), AnaModSetRetrievalFunction(), and InstantiateFeatureSet().

## 17.10 icmk.c File Reference

Functions for computing the ICMK structure of a matrix.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "icmk.h"
#include "petscmat.h"
#include "petscis.h"
#include "petsc.h"
#include "anamodutils.h"
```

Include dependency graph for icmk.c:

### Defines

- #define CHDIF(a) rar[a+1]-rar[a]
- #define SPLIT_BLOCK 3
- #define VERY_FIRST_ROW (isFirst && isplit==0)
- #define VERY_LAST_ROW (isLast && isplit==nsplits-1)
- #define SET_J j = Split_array[jsplit];
- #define SET_TS ts = PetscMax(PetscMin(bs/10,i),1);
- #define SET_TSS tsr = PetscMin(ts,N-j); tsd = PetscMin(ts,M-i); tsc = PetscMin(ts,M-i);
- #define SET_LAST_COL
- #define IF_TOO_FAR_RIGHT_BREAK if (j-ts>lastcol) break;
- #define LEFTDOWN 1
- #define RIGHTDOWN 2
- #define LEFTUP 4
- #define RIGHTUP 8
- #define CORNERUP 16
- #define STATUS(isp, jsp) status[jsplits[isp]+(jsp-joffsets[isplit])]
- #define SET_STATUS(isp, jsp, s) STATUS(isp,jsp) += s
- #define OVERFLOW_DOWN i+tsd-1>=last
- #define OVERFLOW_UP i-ts<first

### Functions

- static void iSpaces (int len)
- int MatIsNull (Mat A, int *flag)
- int MatSubMatIsNull (Mat A, int i1, int i2, int j1, int j2, PetscTruth *isnull)
- static PetscErrorCode PrintArray (int *ar, int len, char *txt)
- static PetscErrorCode CondenseChain (int *chain, int len, int tar, IS *israr)
- static PetscErrorCode CanChain (int lev, int b, int N, int *splits, int nsplits, int *chain, int len, int q, IS *rar)
- int CondenseSplits (int p, IS *splits)
- static PetscErrorCode ChainSplits (int N, int *splits, int s_top, int p, IS *rar)
- static PetscErrorCode GetUpBiDiagSplits (Mat A, int first, int last, int isFirst, int isLast, int **ar, int *n_ar, int **Ar, int *N_ar)
- int MatSplitPoints (Mat A, int np, IS *splitpoints)
- int MatRedistribute (Mat *A, IS splits)
- int VecRedistribute (Vec *V, IS splits)
- static PetscErrorCode compute_icm_splits (Mat A)
- static PetscErrorCode NSplits (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode Splits (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- PetscErrorCode RegisterICMKModules (void)

### Variables

- double mq
- int nc
- int ml

#### 17.10.1 Detailed Description

Functions for computing the ICMK structure of a matrix.

#### 17.10.2 Inverse Cuthill-McKee distribution

ICMK ('Inverse Cuthill-McKee') is a heuristic (Eijkhout[2001]) that, based on the sparsity structure of a matrix, tries to find a meaningful block structure. This is done without permuting the matrix. This block structure can be used in Block Jacobi preconditioners: distributing the matrix over parallel processors according to the block structure often improves convergence.

### 17.10.3 Usage

Activate this module with:

PetscErrorCode RegisterICMKModules();

Compute these elements with

ComputeQuantity("icmk","element",A,&res,&flg);

Available elements are:

- "nsplits" : number of split points found in the matrix
- "splits" : the sequence of split points, including 0 and N

### 17.10.4 References

```
@techreport{Eijk:auto-block,
author = {Victor Eijkhout},
title = {Automatic Determination of Matrix Blocks},
institution = {Department of Computer Science, University of Tennessee},
number = {ut-cs-01-458},
note = {Lapack Working Note 151},
year = {2001}
}
```

Definition in file icmk.c.

### 17.10.5 Define Documentation

#### 17.10.5.1 #define CHDIF(a) rar[a+1]-rar[a]

Referenced by CondenseChain().

#### 17.10.5.2 #define CORNERUP 16

Referenced by MatSplitPoints().

#### 17.10.5.3 #define IF_TOO_FAR_RIGHT_BREAK if (j-ts>lastcol) break;

Referenced by MatSplitPoints().

#### 17.10.5.4 #define LEFTDOWN 1

Referenced by MatSplitPoints().

#### 17.10.5.5 #define LEFTUP 4

Referenced by MatSplitPoints().

#### 17.10.5.6 #define OVERFLOW_DOWN i+tsd-1>=last

Referenced by MatSplitPoints().

#### 17.10.5.7 #define OVERFLOW_UP i-ts<first

Referenced by MatSplitPoints().

#### 17.10.5.8 #define RIGHTDOWN 2

Referenced by MatSplitPoints().

#### 17.10.5.9 #define RIGHTUP 8

Referenced by MatSplitPoints().

#### 17.10.5.10 #define SET_J j = Split_array[jsplit];

Referenced by MatSplitPoints().

#### 17.10.5.11 #define SET_LAST_COL

**Value:**

```
ierr = MatGetRow(A,i,&ncols,&cols,PETSC_NULL); CHKERRQ(ierr); \
      lastcol = cols[ncols-1]; \
      ierr = MatRestoreRow(A,i,&ncols,&cols,PETSC_NULL); CHKERRQ(ierr);
```

Referenced by MatSplitPoints().

#### 17.10.5.12 #define SET_STATUS(isp, jsp, s) STATUS(isp,jsp) += s

Referenced by MatSplitPoints().

#### 17.10.5.13 #define SET_TS ts = PetscMax(PetscMin(bs/10,i),1);

Referenced by MatSplitPoints().

#### 17.10.5.14 #define SET_TSS tsr = PetscMin(ts,N-j); tsd = PetscMin(ts,M-i); tsc = PetscMin(ts,M-i);

Referenced by MatSplitPoints().

#### 17.10.5.15 #define SPLIT_BLOCK 3

Definition at line 145 of file icmk.c.

Referenced by CanChain(), and MatSplitPoints().

#### 17.10.5.16 #define STATUS(isp, jsp) status[jsplits[isp]+(jsp-joffsets[isplit])]

Referenced by MatSplitPoints().

#### 17.10.5.17 #define VERY_FIRST_ROW (isFirst && isplit==0)

Definition at line 253 of file icmk.c.

Referenced by MatSplitPoints().

#### 17.10.5.18 #define VERY_LAST_ROW (isLast && isplit==nsplits-1)

Definition at line 254 of file icmk.c.

Referenced by MatSplitPoints().

### 17.10.6 Function Documentation

#### 17.10.6.1 static PetscErrorCode CanChain (int *lev*, int *b*, int *N*, int * *splits*, int *nsplits*, int * *chain*, int *len*, int *q*, IS * *rar*) [static]

Definition at line 153 of file icmk.c.

References PrintArray(), and SPLIT_BLOCK.

Referenced by ChainSplits().

Here is the call graph for this function:

#### 17.10.6.2 static PetscErrorCode ChainSplits (int *N*, int * *splits*, int *s_top*, int *p*, IS * *rar*) [static]

Definition at line 235 of file icmk.c.

References CanChain(), CondenseSplits(), ml, mq, and nc.

Referenced by MatSplitPoints().

Here is the call graph for this function:

#### 17.10.6.3 static PetscErrorCode compute_icm_splits (Mat *A*) [static]

Definition at line 671 of file icmk.c.

References GetDataID(), HASTOEXIST, id, and MatSplitPoints().

Referenced by NSplits(), and Splits().

Here is the call graph for this function:

#### 17.10.6.4 static PetscErrorCode CondenseChain (int * *chain*, int *len*, int *tar*, IS * *israr*) [static]

Definition at line 114 of file icmk.c.

References CHDIF, and PrintArray().

Referenced by CondenseSplits().

Here is the call graph for this function:

#### 17.10.6.5 int CondenseSplits (int *p*, IS * *splits*)

Definition at line 216 of file icmk.c.

References CondenseChain().

Referenced by ChainSplits().

Here is the call graph for this function:

#### 17.10.6.6 static PetscErrorCode GetUpBiDiagSplits (Mat *A*, int *first*, int *last*, int *isFirst*, int *isLast*, int ** *ar*, int * *n_ar*, int ** *Ar*, int * *N_ar*) [static]

Definition at line 259 of file icmk.c.

References MPIAllGatherIntV(), and TRUTH.

Referenced by MatSplitPoints().

Here is the call graph for this function:

#### 17.10.6.7 static void iSpaces (int *len*) [static]

Definition at line 50 of file icmk.c.

#### 17.10.6.8 int MatIsNull (Mat *A*, int * *flag*)

Definition at line 57 of file icmk.c.

Referenced by MatSplitPoints().

#### 17.10.6.9 int MatRedistribute (Mat * *A*, IS *splits*)

Definition at line 582 of file icmk.c.

#### 17.10.6.10 int MatSplitPoints (Mat *A*, int *np*, IS * *splitpoints*)

Definition at line 308 of file icmk.c.

References ChainSplits(), CORNERUP, GetUpBiDiagSplits(), IF_TOO_FAR_-RIGHT_BREAK, LEFTDOWN, LEFTUP, MatIsNull(), MatSubMatIsNull(), MPIAll-GatherIntV(), OVERFLOW_DOWN, OVERFLOW_UP, RIGHTDOWN, RIGHTUP, SET_J, SET_LAST_COL, SET_STATUS, SET_TS, SET_TSS, SPLIT_BLOCK, Splits(), STATUS, VERY_FIRST_ROW, and VERY_LAST_ROW.

Referenced by compute_icm_splits().

Here is the call graph for this function:

#### 17.10.6.11 int MatSubMatIsNull (Mat *A*, int *i1*, int *i2*, int *j1*, int *j2*, PetscTruth * *isnull*)

Definition at line 78 of file icmk.c.

Referenced by MatSplitPoints().

#### 17.10.6.12 static PetscErrorCode NSplits (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 704 of file icmk.c.

References compute_icm_splits(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterICMKModules().

Here is the call graph for this function:

#### 17.10.6.13 static PetscErrorCode PrintArray (int * *ar*, int *len*, char * *txt*) [static]

Definition at line 99 of file icmk.c.

Referenced by CanChain(), and CondenseChain().

#### 17.10.6.14 PetscErrorCode RegisterICMKModules (void)

Definition at line 749 of file icmk.c.

References ANALYSISINTARRAY, ANALYSISINTEGER, NSplits(), RegisterModule(), and Splits().

Here is the call graph for this function:

#### 17.10.6.15 static PetscErrorCode Splits (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 727 of file icmk.c.

References compute_icm_splits(), GetDataID(), HASTOEXIST, id, and AnalysisItem::ii.

Referenced by MatSplitPoints(), and RegisterICMKModules().

Here is the call graph for this function:

#### 17.10.6.16 int VecRedistribute (Vec * *V*, IS *splits*)

Definition at line 623 of file icmk.c.

### 17.10.7 Variable Documentation

#### 17.10.7.1 int ml

Definition at line 144 of file icmk.c.
Referenced by ChainSplits().

#### 17.10.7.2 double mq

Definition at line 144 of file icmk.c.
Referenced by ChainSplits().

#### 17.10.7.3 int nc

Definition at line 144 of file icmk.c.

Referenced by ChainSplits().

## 17.11 icmk.h File Reference

```
#include "petscmat.h"
#include "petscis.h"
```

Include dependency graph for icmk.h:

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

### Functions

- int MatSplitPoints (Mat A, int np, IS *splitpoints)

#### 17.11.1 Function Documentation

##### 17.11.1.1 int MatSplitPoints (Mat *A*, int *np*, IS * *splitpoints*)

Definition at line 308 of file icmk.c.

References ChainSplits(), CORNERUP, GetUpBiDiagSplits(), IF_TOO_FAR_RIGHT_BREAK, LEFTDOWN, LEFTUP, MatIsNull(), MatSubMatIsNull(), MPIAllGatherIntV(), OVERFLOW_DOWN, OVERFLOW_UP, RIGHTDOWN, RIGHTUP, SET_J, SET_LAST_COL, SET_STATUS, SET_TS, SET_TSS, SPLIT_BLOCK, Splits(), STATUS, VERY_FIRST_ROW, and VERY_LAST_ROW.

Referenced by compute_icm_splits().

Here is the call graph for this function:

## 17.12 iprs.c File Reference

Quantities used in the UKY 'Intelligent Preconditioner Recommendation System'.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamatrix.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscmat.h"
```

Include dependency graph for iprs.c:

### Functions

- static PetscErrorCode computennz (Mat A)

- static PetscErrorCode NnzUp (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode NnzLow (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode NnzDia (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode Nnz (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode RelSymm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode LoBand (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode UpBand (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode AvgNnzpRow (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode AvgDistFromDiag (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode NDiags (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode AvgDiagDist (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SigmaDiagDist (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterIprsModules (void)
- PetscErrorCode DeRegisterIprsModules (void)

### 17.12.1 Detailed Description

Quantities used in the UKY 'Intelligent Preconditioner Recommendation System'.

### 17.12.2 Intelligent Preconditioner Recommendation System

The University of Kentucky 'Intelligent Preconditioner Recommendation System' (Xu[2003]) uses a number of structural properties of the matrix that go beyond the properties in our own Structural properties module.

### 17.12.3 Usage

Activate this module with

PetscErrorCode RegisterIprsModules();

Compute these elements with

ComputeQuantity("iprs",<element>,A,(AnalysisItem*)&res,&reslen,&flg);

Available elements are:

- "nnzup" : number of nonzeros in upper triangle
- "nnzlow" : number of nonzeros in lower triangle
- "nnzdia" : number of nonzeros on the diagonal
- "nnz" : total number of nonzeros
- "avgnnzprow" : average nonzeros per row
- "avgdistfromdiag" : average distance of nonzeros to the diagonal
- "relsymm" : relative symmetry, ratio of structural symmetric elements to total number of nonzeros
- "upband" : bandwidth in the upper triangle
- "loband" : bandwidth in the lower triangle
- "n-nonzero-diags" : number of diagonals that have any nonzero element
- "avg-diag-dist" : average distance of nonzero diagonal to main diagonal
- "sigma-diag-dist" : standard deviation of diag-dist

### 17.12.4 References

```
@techreport{Zhang:interim,
author = {Shu{T}ing Xu and Eun-Joo Lee and Jun Zhang},
title = {An Interim Analysis Report on Preconditioners and Matrices},
institution = {University of Kentucky, Lexington; Department of
    Computer Science},
number = {388-03},
year = {2003}
}
```

Definition in file iprs.c.

### 17.12.5 Function Documentation

#### 17.12.5.1 static PetscErrorCode AvgDiagDist (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 477 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.2 static PetscErrorCode AvgDistFromDiag (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 429 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Here is the call graph for this function:

#### 17.12.5.3 static PetscErrorCode AvgNnzpRow (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 405 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.4 static PetscErrorCode computennz (Mat *A*) [static]

This is the computational routine for all IPRS modules. Lots of pointer chasing.

Definition at line 61 of file iprs.c.

References GetDataID(), HasQuantity(), HASTOEXIST, AnalysisItem::i, id, and MatrixComputeQuantity().

Referenced by AvgDiagDist(), AvgDistFromDiag(), AvgNnzpRow(), LoBand(), NDiags(), Nnz(), NnzDia(), NnzLow(), NnzUp(), SigmaDiagDist(), and UpBand().

Here is the call graph for this function:

#### 17.12.5.5 PetscErrorCode DeRegisterIprsModules (void)

Definition at line 563 of file iprs.c.

#### 17.12.5.6 static PetscErrorCode LoBand (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 331 of file iprs.c.

References computennz(), ComputeQuantity(), GetDataID(), HasComputeModule(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.7 static PetscErrorCode NDiags (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 453 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.8 static PetscErrorCode Nnz (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 273 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.9 static PetscErrorCode NnzDia (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 249 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.10 static PetscErrorCode NnzLow (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 225 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.11 static PetscErrorCode NnzUp (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute the number of nonzeroes in the upper triangle of the matrix

Definition at line 201 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.12 PetscErrorCode RegisterIprsModules (void)

Definition at line 524 of file iprs.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, AvgDiagDist(), AvgNnzpRow(), LoBand(), NDiags(), Nnz(), NnzDia(), NnzLow(), NnzUp(), RegisterModule(), RelSymm(), SigmaDiagDist(), and UpBand().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.12.5.13 static PetscErrorCode RelSymm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 297 of file iprs.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::i, id, and AnalysisItem::r.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.14 static PetscErrorCode SigmaDiagDist (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 501 of file iprs.c.

References computennz(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

#### 17.12.5.15 static PetscErrorCode UpBand (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 368 of file iprs.c.

References computennz(), ComputeQuantity(), GetDataID(), HasComputeModule(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterIprsModules().

Here is the call graph for this function:

## 17.13 jpl.c File Reference

Functions for computing the JPL multicolour structure of a matrix.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "anamatrix.h"
#include "petscerror.h"
#include "petscmat.h"
#include "petscis.h"
#include "src/mat/impls/aij/mpi/mpiaij.h"
#include "src/mat/impls/aij/seq/aij.h"
```

Include dependency graph for jpl.c:

### Functions

- static PetscErrorCode LookAtUncolouredVar (int compare, int this_var, Mat A, PetscScalar *clr_array, PetscScalar *ran_array, PetscReal this_rand, int *neighb, int *nneigh, PetscTruth *colour_now)
- static PetscErrorCode JonesPlassmannColouring (Mat A, PetscTruth *flg)
- static PetscErrorCode NColours (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode ColourSizes (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode ColourOffsets (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode Colours (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterJPLModules (void)

#### 17.13.1 Detailed Description

Functions for computing the JPL multicolour structure of a matrix.

#### 17.13.2 Jones-Plassmann multi-colouring

#### 17.13.3 Usage

Activate this module with

PetscErrorCode RegisterJPLModules();

Compute these elements with

ComputeQuantity("jpl","element",A,(void*)&res,&flg);

Available elements are:

- "n-colours" : the number of colours computed
- "colour-set-sizes" : an array containing in location i the number of points of colour i. (See Array type handling for technicalities on arrays.)
- "colour-offsets" : locations (zero-based) of the colour sets in the colours array
- "colours" : points sorted first by colour, then increasing index

This routine works in parallel; the stored result is distributed over all participating processors.

### 17.13.4 References

```
@article{JoPl:heuristic,
author = {M.T. Jones and P.E. Plassmann},
title = {A parallel graph coloring heuristic},
journal = SJSSC,
volume = {14},
year = {1993},
keywords = {incomplete factorization, multicolouring, matrix graph}
}

@inproceedings{JoPl:parallelunstructured,
author = {M.T. Jones and P.E. Plassmann},
title = {Parallel solution of unstructed, sparse systems of linear equations},
booktitle = {Proceedings of the Sixth SIAM conference on
Parallel Processing for Scientific Computing},
editor = {R.F. Sincovec and D.E. Keyes and M.R. Leuze and L.R. Petzold
 and D.A. Reed},
publisher = {SIAM},
address = {Philadelphia},
pages = {471--475},
year = {1993}

  \section Disclaimer

  Unabashed use of British spelling of 'colour' needs no disclaimer.
}
```

Definition in file jpl.c.

### 17.13.5 Function Documentation

#### 17.13.5.1 static PetscErrorCode ColourOffsets (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute an array that has the offsets of the locations of the colours, on this processor. If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

Definition at line 451 of file jpl.c.

References GetDataID(), HASTOEXIST, id, AnalysisItem::ii, and JonesPlassmann-Colouring().

Referenced by RegisterJPLModules().

Here is the call graph for this function:

#### 17.13.5.2 static PetscErrorCode Colours (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute an array that has the colour sets on this processor. In order to know where a set starts or ends, the result of ColourOffsets() is needed.

If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

Definition at line 497 of file jpl.c.

References GetDataID(), HASTOEXIST, id, AnalysisItem::ii, and JonesPlassmann-Colouring().

Referenced by RegisterJPLModules().

Here is the call graph for this function:

#### 17.13.5.3 static PetscErrorCode ColourSizes (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute an array that has the number of points of each colour, on this processor. If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

Definition at line 408 of file jpl.c.

References GetDataID(), HASTOEXIST, id, AnalysisItem::ii, and JonesPlassmann-Colouring().

Referenced by RegisterJPLModules().

Here is the call graph for this function:

#### 17.13.5.4 static PetscErrorCode JonesPlassmannColouring (Mat *A*, PetscTruth ∗*flg*) [static]

Compute a multicolour ordering of a matrix, in parallel.

This is the main computational routine.

Definition at line 129 of file jpl.c.

References GetDataID(), id, LookAtUncolouredVar(), and MatrixComputeQuantity().

Referenced by ColourOffsets(), Colours(), ColourSizes(), and NColours().

Here is the call graph for this function:

#### 17.13.5.5 static PetscErrorCode LookAtUncolouredVar (int *compare*, int *this_var*, Mat *A*, PetscScalar ∗ *clr_array*, PetscScalar ∗ *ran_array*, PetscReal *this_rand*, int ∗ *neighb*, int ∗ *nneigh*, PetscTruth ∗ *colour_now*) [static]

Investigate the local and remote connections of a variable to see whether it needs to be coloured, and if so, with what colour.

Definition at line 77 of file jpl.c.

Referenced by JonesPlassmannColouring().

#### 17.13.5.6 static PetscErrorCode NColours (AnaModNumericalProblem *prob*, AnalysisItem ∗ *rv*, int ∗ *lv*, PetscTruth ∗ *flg*) [static]

Compute the global number of colours. Note that any given processor does not need to have all the colours.

Definition at line 377 of file jpl.c.

References GetDataID(), HASTOEXIST, AnalysisItem::i, id, and JonesPlassmann-Colouring().

Referenced by RegisterJPLModules().

Here is the call graph for this function:

#### 17.13.5.7 PetscErrorCode RegisterJPLModules (void)

Definition at line 523 of file jpl.c.

References ANALYSISINTARRAY, ANALYSISINTEGER, ColourOffsets(), Colours(), ColourSizes(), NColours(), and RegisterModule().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

## 17.14 lapack.c File Reference

Dense routines from Lapack: eigenvalue and schur stuff.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscmat.h"
```

Include dependency graph for lapack.c:

### Defines

- #define ABS(x) (x>0?x:-x)
- #define EIGENVALUE(re, im)

### Functions

- void LAPACK_DGEESX (const char *jobvs, const char *sort, char *sel, const char *sense, const int *n, double *a, const int *lda, int *sdim, double *wr, double *wi, double *vs, const int *ldvs, double *rconde, double *rcondv, double *work, const int *lwork, int *iwork, const int *liwork, int *bwork, int *info)
- static PetscErrorCode eigenvaluecomp (Mat A)
- static PetscErrorCode Departure (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MinEVbyMagIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)

- static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MaxEVbyRealIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MaxEVbyImRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- static PetscErrorCode MaxEVbyImIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *dummy, PetscTruth *flg)
- PetscErrorCode RegisterLapackModules ()

### 17.14.1 Detailed Description

Dense routines from Lapack: eigenvalue and schur stuff.

### 17.14.2 Dense calculations from Lapack

### 17.14.3 Usage

Activate this module with

PetscErrorCode RegisterLapackModules();

Compute these elements with

ComputeQuantity("lapack",<element>,A,(AnalysisItem*)&res,&flg);

Available elements are:

- something

Definition in file lapack.c.

### 17.14.4 Define Documentation

#### 17.14.4.1 #define ABS(x) (x>0?x:-x)

Referenced by eigenvaluecomp().

#### 17.14.4.2 #define EIGENVALUE(re, im)

**Value:**

```
{ \
          PetscReal mag = sqrt(re*re+im*im); \
          if (!maxbyreset) {maxbyreset = 1; maxbyrere = re; maxbyreim = im; } \
          else if (re>maxbyrere) { maxbyrere = re; maxbyreim = im; }    \
          if (!maxbyimset) {maxbyimset = 1; maxbyimre = re; maxbyimim = im; } \
          else if (im>maxbyimre) { maxbyimre = re; maxbyimim = im; }    \
          if (!maxbymagset) {maxbymagset = 1; maxmag = mag;             \
            maxbymagre = re; maxbymagim = im; }                         \
          else if (mag>maxmag) { maxbymagre = re; maxbymagim = im; }    \
          if (!minbymagset) {minbymagset = 1; minmag = mag;             \
            minbymagre = re; minbymagim = im; }                         \
          else if (mag<minmag) { minbymagre = re; minbymagim = im; }    \
      }
```

Referenced by eigenvaluecomp().

### 17.14.5 Function Documentation

#### 17.14.5.1 static PetscErrorCode Departure (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 196 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.2 static PetscErrorCode eigenvaluecomp (Mat *A*) [static]

This is the computational routine for lapack eigenvalue calculation.

Definition at line 41 of file lapack.c.

References ABS, EIGENVALUE, GetDataID(), HASTOEXIST, id, and LAPACK_-DGEESX().

Referenced by Departure(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), and

MinEVbyMagRe().

Here is the call graph for this function:

#### 17.14.5.3 void LAPACK_DGEESX (const char * *jobvs*, const char * *sort*, char * *sel*, const char * *sense*, const int * *n*, double * *a*, const int * *lda*, int * *sdim*, double * *wr*, double * *wi*, double * *vs*, const int * *ldvs*, double * *rconde*, double * *rcondv*, double * *work*, const int * *lwork*, int * *iwork*, const int * *liwork*, int * *bwork*, int * *info*)

Referenced by eigenvaluecomp().

#### 17.14.5.4 static PetscErrorCode MaxEVbyImIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 397 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.5 static PetscErrorCode MaxEVbyImRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 372 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.6 static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 247 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.7 static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 222 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.8 static PetscErrorCode MaxEVbyRealIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 347 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.9 static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 322 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.10 static PetscErrorCode MinEVbyMagIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 297 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.11 static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *dummy*, PetscTruth * *flg*) [static]

Definition at line 272 of file lapack.c.

References eigenvaluecomp(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterLapackModules().

Here is the call graph for this function:

#### 17.14.5.12 PetscErrorCode RegisterLapackModules ()

Declare norm-like modules that can be performed with lapack calculations.

Definition at line 424 of file lapack.c.

References ANALYSISDOUBLE, Departure(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), and RegisterModule().

Referenced by AnaModRegisterStandardModules().

Here is the call graph for this function:

## 17.15 logging.c File Reference

PETSc event logging in AnaMod.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include "petsc.h"
#include "anamod.h"
```

Include dependency graph for logging.c:

**Functions**

- PetscErrorCode CategoryLogEventRegister (char ∗cat, int icat)

### 17.15.1 Detailed Description

PETSc event logging in AnaMod.

### 17.15.2 Event logging

We use PETSc Log Events to report on time expended in AnaMod routines.

Definition in file logging.c.

### 17.15.3 Function Documentation

#### 17.15.3.1 PetscErrorCode CategoryLogEventRegister (char ∗ *cat*, int *icat*)

Definition at line 20 of file logging.c.

## 17.16 Make.inc File Reference

## 17.17 module_functions.c File Reference

User functions for accessing the analysis modules.

```
#include <stdlib.h>
#include <string.h>
#include "anamod.h"
```

```
#include "anamodsalsamodules.h"
#include "petscmat.h"
```

Include dependency graph for module_functions.c:

### Defines

- #define MXC 30
- #define INC 7
- #define TYPEii(icat, icmp) types[icat][icmp]

### Functions

- PetscErrorCode AnaModInitialize ()
- PetscErrorCode AnaModFinalize ()
- PetscErrorCode AnaModGetTypeName (int id, char **name)
- PetscErrorCode AnaModGetTypeMySQLName (int id, char **name)
- PetscErrorCode RegisterModule (const char *cat, const char *cmp, AnalysisDataType type, PetscErrorCode(*f)(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *))
- PetscErrorCode DeRegisterCategory (char *cat)
- PetscErrorCode DeregisterModules ()
- PetscErrorCode GetCategories (char ***cats, int *n)
- PetscErrorCode CategoryGetModules (char *cat, char ***ms, AnalysisDataType **t, int **id, int *n)
- static PetscErrorCode GetCategoryIndex (char *cat, int *idx, PetscTruth *f)
- static PetscErrorCode GetModuleIndex (int icat, char *cmp, int *idx, PetscTruth *f)
- PetscErrorCode HasComputeCategory (char *cat, PetscTruth *f)
- PetscErrorCode HasComputeModule (char *cat, char *cmp, PetscTruth *f)
- PetscErrorCode DeclareCategoryOptionFunction (char *cat, PetscErrorCode(*f)(char *))
- PetscErrorCode GetCategoryOptionFunction (char *cat, PetscErrorCode(**f)(char *))

- PetscErrorCode ComputeQuantity (AnaModNumericalProblem prob, const char *cat, const char *cmp, AnalysisItem *res, int *rreslen, PetscTruth *success)
- PetscErrorCode RetrieveQuantity (AnaModNumericalProblem prob, char *cat, char *cmp, AnalysisItem *res, PetscTruth *success)
- PetscErrorCode HasQuantity (AnaModNumericalProblem prob, char *cat, char *cmp, PetscTruth *f)
- PetscErrorCode HasQuantityByID (AnaModNumericalProblem prob, int id, AnalysisDataType type, PetscTruth *f)
- PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem prob, int icat, int icmp, AnalysisItem *result, PetscTruth *f)
- PetscErrorCode QuantityAsString (AnalysisItem *q, AnalysisDataType t, char **s)
- PetscErrorCode GetDataID (const char *cat, const char *cmp, int *id, PetscTruth *flg)
- PetscErrorCode GetDataType (const char *cat, const char *cmp, AnalysisDataType *t)

### Variables

- static int ncategories = 0
- static int maxcategories = 30
- static int * ncomponents = NULL
- static int * maxcomponents = NULL
- static PetscErrorCode(*** modules )(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *) = NULL
- static PetscErrorCode(** optionfunctions )(char *) = NULL
- static PetscTruth ** hasval = NULL
- static int ** ids = NULL
- static char ** categories = NULL
- static char *** components = NULL
- static AnalysisDataType ** types = NULL
- struct {
  int id
  const char * name
  const char * mysqlname
  } anamodtypenames [5]

- static int nAnaModTypeNames
- static int AnaModIsInitialized = 0

### 17.17.1 Detailed Description

User functions for accessing the analysis modules.

\ Analysis modules need to be defined with RegisterModule(), after which they can be invoked with ComputeQuantity(). See also HasQuantity(). The functions AnaModRegisterStandardModules() installs all standard available modules.

There are utility functions for querying the existence of modules: GetCategories(), CategoryGetModules(), HasComputeCategory(), HasComputeModule().

Utility functions: QuantityAsString(), GetDataType(), GetDataID(), GetCategoryIndex(), GetModuleIndex().

Definition in file module_functions.c.

### 17.17.2 Define Documentation

#### 17.17.2.1 #define INC 7

Definition at line 304 of file module_functions.c.

Referenced by RegisterModule().

#### 17.17.2.2 #define MXC 30

Definition at line 303 of file module_functions.c.

Referenced by RegisterModule().

#### 17.17.2.3 #define TYPEii(icat, icmp) types[icat][icmp]

Definition at line 311 of file module_functions.c.

Referenced by ComputeQuantity(), GetDataType(), RetrieveQuantity(), and RetrieveQuantityByID().

### 17.17.3 Function Documentation

#### 17.17.3.1 PetscErrorCode AnaModFinalize ()

Definition at line 348 of file module_functions.c.

Referenced by main().

#### 17.17.3.2 PetscErrorCode AnaModGetTypeMySQLName (int *id*, char ** *name*)

Definition at line 372 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, mysqlname, and nAnaModTypeNames.

Referenced by main().

#### 17.17.3.3 PetscErrorCode AnaModGetTypeName (int *id*, char ** *name*)

Definition at line 356 of file module_functions.c.

References anamodtypenames, and nAnaModTypeNames.

#### 17.17.3.4 PetscErrorCode AnaModInitialize ()

Definition at line 319 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, and nAnaModTypeNames.

Referenced by main().

#### 17.17.3.5 PetscErrorCode CategoryGetModules (char * *cat*, char *** *ms*, AnalysisDataType ** *t*, int ** *id*, int * *n*)

Query the modules in a specified category.

The category name has to exist. The routine will call the Petsc error handler if the name is invalid.

Parameters:

- `cat` : the category that is being queried
- `ms` (optional) : the names of the modules in the category

- `t` (optional) : the corresponding list of datatypes
- `id` (optional) : the list of module IDs (see RetrieveQuantityByID())
- `n` (optional) : the number of modules in the category

See also GetCategories() and HasComputeCategory().

Definition at line 711 of file module_functions.c.

References categories, components, ids, ncategories, ncomponents, and types.

Referenced by analyze_matrix(), main(), and ReportAnamodContent().

#### 17.17.3.6 PetscErrorCode ComputeQuantity (AnaModNumericalProblem *prob*, const char * *cat*, const char * *cmp*, AnalysisItem * *res*, int * *rreslen*, PetscTruth * *success*)

Compute a computational module from a certain category.

Argument:

1. the name of the category (see GetCategories())
2. the name of the module (see CategoryGetModules())
3. the matrix
4. a pointer to the result. This is given as `"(AnalysisItem*)&res"` ; see types for the definition of the AnalysisItem data type
5. the length of the result if the result is an array. This argument can be NULL.
6. a success indicator. Failure can have obvious causes, such as breakdown of an internal routine, but the routine can also refuse to compute a quantity if doing so would be too expensive (see an example in the Estimates for the departure from normality category).

A call to this routine need not involve actual computation: the requested quantity can already be attached to the matrix object (see attached quantities for details). This mechanism is used in all the standard modules that come with the AnaMod package.

The workings of this function can be traced by specifying a trace function; see Tracing the analysis modules.

Definition at line 866 of file module_functions.c.

References ANALYSISINTARRAY, AnaModHasTrace(), AnaModTraceArrays(), AnaModTraceMessage(), GetCategoryIndex(), GetModuleIndex(), hasval, modules, QuantityAsString(), TYPEii, and types.

Referenced by analyze_matrix(), DepartureRuhe75(), LoBand(), MatrixComputeQuantity(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), RelSymm(), and UpBand().

Here is the call graph for this function:

#### 17.17.3.7 PetscErrorCode DeclareCategoryOptionFunction (char * *cat*, PetscErrorCode(*)(char *) *f*)

This function allows the module developer to give the user commandline options for control of a module.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 807 of file module_functions.c.

References GetCategoryIndex(), and optionfunctions.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.17.3.8 PetscErrorCode DeRegisterCategory (char * *cat*)

Deallocate the storage for a particular category of analysis modules

Definition at line 612 of file module_functions.c.

References categories, components, hasval, ids, modules, ncategories, ncomponents, and types.

Referenced by DeregisterModules().

#### 17.17.3.9 PetscErrorCode DeregisterModules (void)

Definition at line 642 of file module_functions.c.

References categories, components, DeRegisterCategory(), GetCategories(), hasval, ids, maxcomponents, modules, ncategories, ncomponents, optionfunctions, and types.

Here is the call graph for this function:

#### 17.17.3.10 PetscErrorCode GetCategories (char *** *cats*, int * *n*)

Query the number of defined categories and their names.

Parameters:

- `cats` (optional) output: list of category names
- `n` output: number of currently installed categories

See also CategoryGetModules().

Definition at line 687 of file module_functions.c.

References categories, and ncategories.

Referenced by analyze_matrix(), AnaModOptionsHandling(), AnaModShowOptions(), DeregisterModules(), main(), and ReportAnamodContent().

#### 17.17.3.11 static PetscErrorCode GetCategoryIndex (char * *cat*, int * *idx*, PetscTruth * *f*) [static]

Get the index of a stated category. Internal routine.

Definition at line 733 of file module_functions.c.

References categories, and ncategories.

Referenced by ComputeQuantity(), DeclareCategoryOptionFunction(), GetCategoryOptionFunction(), GetDataID(), GetDataType(), HasComputeCategory(), HasComputeModule(), and RetrieveQuantity().

#### 17.17.3.12 PetscErrorCode GetCategoryOptionFunction (char * *cat*, PetscErrorCode(**)(char *) *f*)

This function is called in AnaModOptionsHandling(). There is probably no reason for the user ever to call it.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 827 of file module_functions.c.

References GetCategoryIndex(), and optionfunctions.

Referenced by AnaModOptionsHandling().

Here is the call graph for this function:

#### 17.17.3.13 PetscErrorCode GetDataID (const char * *cat*, const char * *cmp*, int * *id*, PetscTruth * *flg*)

Get the numerical id under which a module is stored.

The flag parameter returns false if the category or cat/cmp does not exist. The flag parameter is optional, but the routine will exit if the cat/cmp is not found and the flag parameter is not provided.

This is not a user level function.

Definition at line 1164 of file module_functions.c.

References GetCategoryIndex(), GetModuleIndex(), and ids.

Referenced by AvgDiagDist(), AvgDistFromDiag(), AvgNnzpRow(), BlockSize(), ColourOffsets(), Colours(), ColourSizes(), ColVariability(), Commutator(), compute_dd(), compute_dummy_rows(), compute_eigenvalues(), compute_ellipse_from_Ritz_values(), compute_icm_splits(), compute_nnz_structure(), compute_posdiag(), compute_singularvalues(), compute_tracea2(), ComputeDiagonal(), computennz(), computetrace(), ComputeVariability(), Departure(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), DiagDefinite(), DiagonalAverage(), DiagonalDominance(), DiagonalSign(), DiagonalVariance(), DiagZeroStart(), DummyRows(), DummyRowsKind(), eigenvaluecomp(), HasQuantity(), JonesPlassmannColouring(), Kappa(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUnstruct(), PosFraction(), RBandWidth(), regularblocks(), RelSymm(), RightSkyline(), RitzValuesC(), RitzValuesR(), RowVariability(), SigmaDiagDist(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), Splits(), Symmetry(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), TraceAbs(), UpBand(), and Version().

Here is the call graph for this function:

#### 17.17.3.14 PetscErrorCode GetDataType (const char * *cat*, const char * *cmp*, AnalysisDataType * *t*)

Retrieve the `AnalysisDataType` of a computational module

Definition at line 1191 of file module_functions.c.

References GetCategoryIndex(), GetModuleIndex(), and TYPEii.

Referenced by analyze_matrix(), and HasQuantity().

Here is the call graph for this function:

#### 17.17.3.15 static PetscErrorCode GetModuleIndex (int *icat*, char * *cmp*, int * *idx*, PetscTruth * *f*) [static]

Get the index of a stated module inside a category. Internal routine.

Definition at line 753 of file module_functions.c.

References components, and ncomponents.

Referenced by ComputeQuantity(), GetDataID(), GetDataType(), HasComputeModule(), and RetrieveQuantity().

#### 17.17.3.16 PetscErrorCode HasComputeCategory (char * *cat*, PetscTruth * *f*)

Query whether a specified category has been declared.

Definition at line 773 of file module_functions.c.

References GetCategoryIndex().

Here is the call graph for this function:

#### 17.17.3.17 PetscErrorCode HasComputeModule (char * *cat*, char * *cmp*, PetscTruth * *f*)

Query whether a specified module exists inside a specified category. The category need not itself have been declared.

Definition at line 787 of file module_functions.c.

References GetCategoryIndex(), and GetModuleIndex().

Referenced by LoBand(), and UpBand().

Here is the call graph for this function:

#### 17.17.3.18 PetscErrorCode HasQuantity (AnaModNumericalProblem *prob*, char * *cat*, char * *cmp*, PetscTruth * *f*)

Check if a certain quantity is precomputed, meaning that it can be retrieved (with a call to ComputeQuantity()) at no computational cost.

The category and module names have to exist. The routine will call the Petsc error handler if either name is invalid. Use HasComputeModule() to test whether a category and module is known to the system.

See the page on attached quantities for an explanation of the mechanism behind this routine.

Definition at line 1017 of file module_functions.c.

References GetDataID(), GetDataType(), and HasQuantityByID().

Referenced by computennz().

Here is the call graph for this function:

#### 17.17.3.19 PetscErrorCode HasQuantityByID (AnaModNumericalProblem *prob*, int *id*, AnalysisDataType *type*, PetscTruth * *f*)

Auxiliary routine with lookup by ID, which is much faster than by string indexing.

Definition at line 1035 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, and ANALYSISINTEGER.

Referenced by HasQuantity().

#### 17.17.3.20 PetscErrorCode QuantityAsString (AnalysisItem * *q*, AnalysisDataType *t*, char ** *s*)

Generate a character string for a given quantity.

Definition at line 1101 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, ANALYSISSTRING, AnalysisItem::c, AnalysisItem::i, AnalysisItem::ii, AnalysisItem::len, AnalysisItem::r, and AnalysisItem::rr.

Referenced by ComputeQuantity(), ReportAnamodContent(), and RetrieveQuantity().

#### 17.17.3.21 PetscErrorCode RegisterModule (const char * *cat*, const char * *cmp*, AnalysisDataType *type*, PetscErrorCode(*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *) *f*)

Register a new computational module

This adds a computational routine (the `f` argument) into the modules database under the given category (`cat`) and module (`cmp`) label. If the category does not exist yet, it is created.

The available types are defined in anamodtypes.h

If the routine is NULL, only the category and component are created.

Routine prototype:

- problem (input)
- item (output)
- array length (output)
- success (output)

See also HasComputeModule(), ComputeQuantity(), DeregisterModules().

Definition at line 410 of file module_functions.c.

References categories, components, hasval, id, ids, INC, maxcategories, maxcomponents, modules, MXC, ncategories, ncomponents, optionfunctions, and types.

Referenced by RegisterICMKModules(), RegisterIprsModules(), RegisterJPLModules(), RegisterLapackModules(), RegisterNormalityModules(), RegisterSimpleModules(), RegisterSpectrumModules(), RegisterStatsModules(), RegisterStructureModules(), and RegisterVarianceModules().

#### 17.17.3.22 PetscErrorCode RetrieveQuantity (AnaModNumericalProblem *prob*, char * *cat*, char * *cmp*, AnalysisItem * *res*, PetscTruth * *success*)

Retrieve an attached quantity. Note that this does not report the length of arrays; you have to know under which name this is stored.

Definition at line 931 of file module_functions.c.

References ANALYSISINTARRAY, AnaModHasTrace(), AnaModTraceArrays(), AnaModTraceMessage(), GetCategoryIndex(), GetModuleIndex(), hasval, QuantityAsString(), RetrieveQuantityByID(), TYPEii, and types.

Here is the call graph for this function:

#### 17.17.3.23 PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem *prob*, int *icat*, int *icmp*, AnalysisItem * *result*, PetscTruth * *f*)

See also HasQuantityByID()

Definition at line 1066 of file module_functions.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, AnalysisItem::i, ids, AnalysisItem::ii, AnalysisItem::r, AnalysisItem::rr, and TYPEii.

Referenced by RetrieveQuantity().

### 17.17.4 Variable Documentation

#### 17.17.4.1 int AnaModIsInitialized = 0 [static]

Definition at line 315 of file module_functions.c.

Referenced by AnaModGetTypeMySQLName(), and AnaModInitialize().

#### 17.17.4.2 struct { ... } anamodtypenames[5] [static]

Referenced by AnaModGetTypeMySQLName(), AnaModGetTypeName(), and AnaModInitialize().

#### 17.17.4.3 char** categories = NULL [static]

Definition at line 310 of file module_functions.c.

Referenced by analyze_matrix(), AnaModOptionsHandling(), AnaModShowOptions(), CategoryGetModules(), DeRegisterCategory(), DeregisterModules(), GetCategories(), GetCategoryIndex(), RegisterModule(), and ReportAnamodContent().

#### 17.17.4.4 char *** components = NULL [static]

Definition at line 310 of file module_functions.c.

Referenced by CategoryGetModules(), DeRegisterCategory(), DeregisterModules(), GetModuleIndex(), and RegisterModule().

#### 17.17.4.5 PetscTruth** hasval = NULL [static]

Definition at line 309 of file module_functions.c.

Referenced by ComputeQuantity(), DeRegisterCategory(), DeregisterModules(), RegisterModule(), and RetrieveQuantity().

#### 17.17.4.6 int id

Definition at line 313 of file module_functions.c.

Referenced by AvgDiagDist(), AvgDistFromDiag(), AvgNnzpRow(), BlockSize(), ColourOffsets(), Colours(), ColourSizes(), ColVariability(), Commutator(), compute_dd(), compute_dummy_rows(), compute_eigenvalues(), compute_ellipse_from_Ritz_values(), compute_icm_splits(), compute_nnz_structure(), compute_posdiag(), compute_singularvalues(), compute_tracea2(), ComputeDiagonal(), computennz(), computetrace(), ComputeVariability(), Departure(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), DiagDefinite(), DiagonalAverage(), DiagonalDominance(), DiagonalSign(), DiagonalVariance(), DiagZeroStart(), DummyRows(), DummyRowsKind(), eigenvaluecomp(), JonesPlassmannColouring(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUnstruct(), RBandWidth(), RegisterModule(), regularblocks(), RelSymm(), RightSkyline(), RitzValuesC(), RitzValuesR(), RowVariability(), SigmaDiagDist(), Splits(), Symmetry(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), TraceAbs(), UpBand(), and Version().

#### 17.17.4.7 int** ids = NULL [static]

Definition at line 309 of file module_functions.c.

Referenced by CategoryGetModules(), DeRegisterCategory(), DeregisterModules(), GetDataID(), RegisterModule(), and RetrieveQuantityByID().

#### 17.17.4.8 int maxcategories = 30 [static]

Definition at line 305 of file module_functions.c.

Referenced by RegisterModule().

#### 17.17.4.9 int * maxcomponents = NULL [static]

Definition at line 306 of file module_functions.c.

Referenced by DeregisterModules(), and RegisterModule().

#### 17.17.4.10 PetscErrorCode(*** modules)(AnaModNumericalProblem, AnalysisItem *, int *, PetscTruth *) = NULL [static]

Referenced by analyze_matrix(), ComputeQuantity(), DeRegisterCategory(), DeregisterModules(), RegisterModule(), and ReportAnamodContent().

#### 17.17.4.11 const char * mysqlname

Definition at line 313 of file module_functions.c.

Referenced by AnaModGetTypeMySQLName().

#### 17.17.4.12 const char* name

Definition at line 313 of file module_functions.c.

Referenced by AddToFeatureSet().

#### 17.17.4.13 int nAnaModTypeNames [static]

Definition at line 314 of file module_functions.c.

Referenced by AnaModGetTypeMySQLName(), AnaModGetTypeName(), and AnaModInitialize().

#### 17.17.4.14 int ncategories = 0 [static]

Definition at line 305 of file module_functions.c.

Referenced by CategoryGetModules(), DeRegisterCategory(), DeregisterModules(), GetCategories(), GetCategoryIndex(), and RegisterModule().

#### 17.17.4.15 int * ncomponents = NULL [static]

Definition at line 306 of file module_functions.c.

Referenced by CategoryGetModules(), DeRegisterCategory(), DeregisterModules(), GetModuleIndex(), and RegisterModule().

#### 17.17.4.16 PetscErrorCode(** optionfunctions)(char *) = NULL [static]

Referenced by DeclareCategoryOptionFunction(), DeregisterModules(), GetCategoryOptionFunction(), and RegisterModule().

#### 17.17.4.17 AnalysisDataType** types = NULL [static]

Definition at line 312 of file module_functions.c.

Referenced by analyze_matrix(), CategoryGetModules(), ComputeQuantity(), DeRegisterCategory(), DeregisterModules(), main(), RegisterModule(), ReportAnamodContent(), and RetrieveQuantity().

## 17.18 normal.c File Reference

Various estimates of the departure from normality.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "anamatrix.h"
#include "petscerror.h"
#include "petscksp.h"
#include "petscmat.h"
```

Include dependency graph for normal.c:

### Defines

- #define min(a, b) (a<b?a:b)
- #define max(a, b) (a>b?a:b)

### Functions

- static PetscErrorCode SparseVecProd (int na, const int *ia, const PetscScalar *va, int nb, const int *ib, const PetscScalar *vb, PetscScalar *result)
- static PetscErrorCode MatCenter (Mat A)
- static PetscErrorCode Lee95bounds (Mat A)
- static PetscErrorCode DepartureLee95 (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode compute_tracea2_seq (Mat A, PetscScalar *trace)
- static PetscErrorCode compute_tracea2 (Mat A, PetscTruth *done)
- static PetscErrorCode TraceA2 (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode CommutatorNormFAllowSqrtTimes (int n)
- static PetscErrorCode MatCommutatorNormF_seq (Mat A, PetscReal *result, PetscTruth *done)
- static PetscErrorCode MatCommutatorNormF (Mat A, PetscTruth *done)
- static PetscErrorCode Commutator (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode Lee96bounds (Mat A)
- static PetscErrorCode DepartureLee96U (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DepartureLee96L (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DepartureRuhe75 (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterNormalityModules (void)

### Variables

- static int sqrt_times = 50

### 17.18.1 Detailed Description

Various estimates of the departure from normality.

### 17.18.2 Estimates for the departure from normality

The departure from normality is a very hard to calculate quantity. This file provides several estimates.

### 17.18.3 Usage

Activate this module with:

PetscErrorCode RegisterNormalityModules();

Compute these elements with

ComputeQuantity("normal",<element>,A,&res,&reslen,&flg);

The auxiliary quantity of the Frobenius norm of the commutator can be very expensive to calculate. If the bandwidth of the matrix is more then a specified times (default: 4) the square root of the matrix size, this routine returns with failure. Set this tolerance with

PetscErrorCode CommutatorNormFAllowSqrtTimes(int n);

If this quantity is unavailable, so are the Lee96 bounds.

Available elements are:

- "trace-asquared" : an auxiliary quantity
- "commutator-normF" : the Frobenius norm of the commutator $AA^t - A^tA$
- "ruhe75-bound" : the bound from Ruhe[1975]
- "lee95-bound" : the bound from Lee[1995]
- "lee96-ubound" : the upper bound from Lee[1996]
- "lee96-lbound" : the lower bound from Lee[1996]

### 17.18.4 References

```
@article{ElPa:nonnormality,
author = {L. Elsner and M.H.C. Paardekoper},
title = {On Measures of Non-normality for Matrices},
journal = LAA,
volume = {307/308},
year = {1979},
pages = {107--124}
}

@article{Lee:upper-bound,
author = {Steven L. Lee},
title = {A Practical Upper Bound for Departure from Normality},
journal = SIMAT,
volume = {16},
issue = {2},
year = {1995},
pages = {462--468},
abstract = {Upper bound expressed in terms of Hermitian and Anti-Hermitian
    part; hence computable in $O(n^2)$ ops, as opposed to bounds by
    Henrici~\cite{Henrici:nonnormal-bounds}, which are based on $A^HA$ and
    take $O(n^3)$ ops.}
}

@article{Lee:available-bound,
author = {Steven L. Lee},
title = {Best Available Bounds for Departure from Normality},
journal = SIMAT,
volume = {17},
issue = {4},
year = {1996},
pages = {984--991}

}
```

Definition in file normal.c.

### 17.18.5 Define Documentation

#### 17.18.5.1 #define max(a, b) (a>b?a:b)

Referenced by MatCommutatorNormF_seq().

#### 17.18.5.2 #define min(a, b) (a<b?a:b)

Referenced by MatCommutatorNormF_seq().

### 17.18.6 Function Documentation

#### 17.18.6.1 static PetscErrorCode Commutator (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 415 of file normal.c.

References GetDataID(), HASTOEXIST, id, MatCommutatorNormF(), and AnalysisItem::r.

Referenced by Lee96bounds(), and RegisterNormalityModules().

Here is the call graph for this function:

#### 17.18.6.2 PetscErrorCode CommutatorNormFAllowSqrtTimes (int *n*)

Definition at line 281 of file normal.c.

Referenced by AnaModRegisterSalsaModules().

#### 17.18.6.3 static PetscErrorCode compute_tracea2 (Mat *A*, PetscTruth * *done*) [static]

Definition at line 221 of file normal.c.

References AnaModGetSequentialMatrix(), compute_tracea2_seq(), GetDataID(), HASTOEXIST, and id.

Referenced by TraceA2().

Here is the call graph for this function:

#### 17.18.6.4 static PetscErrorCode compute_tracea2_seq (Mat *A*, PetscScalar * *trace*) [static]

Definition at line 195 of file normal.c.

References SparseVecProd().

Referenced by compute_tracea2().

Here is the call graph for this function:

#### 17.18.6.5 static PetscErrorCode DepartureLee95 (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 172 of file normal.c.

References GetDataID(), HASTOEXIST, id, Lee95bounds(), and AnalysisItem::r.

Referenced by RegisterNormalityModules().

Here is the call graph for this function:

#### 17.18.6.6 static PetscErrorCode DepartureLee96L (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 506 of file normal.c.

References GetDataID(), HASTOEXIST, id, Lee96bounds(), and AnalysisItem::r.

Referenced by RegisterNormalityModules().

Here is the call graph for this function:

#### 17.18.6.7 static PetscErrorCode DepartureLee96U (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 482 of file normal.c.

References GetDataID(), HASTOEXIST, id, Lee96bounds(), and AnalysisItem::r.

Referenced by RegisterNormalityModules().

Here is the call graph for this function:

#### 17.18.6.8 static PetscErrorCode DepartureRuhe75 (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 530 of file normal.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterNormalityModules().

Here is the call graph for this function:

#### 17.18.6.9 static PetscErrorCode Lee95bounds (Mat *A*) [static]

Definition at line 139 of file normal.c.

References GetDataID(), HASTOEXIST, id, MatCenter(), MatrixComputeQuantity(), and AnalysisItem::r.

Referenced by DepartureLee95().

Here is the call graph for this function:

#### 17.18.6.10 static PetscErrorCode Lee96bounds (Mat *A*) [static]

Definition at line 440 of file normal.c.

References Commutator(), GetDataID(), HASTOEXIST, id, MatCenter(), and TraceA2().

Referenced by DepartureLee96L(), and DepartureLee96U().

Here is the call graph for this function:

#### 17.18.6.11 static PetscErrorCode MatCenter (Mat *A*) [static]

Definition at line 105 of file normal.c.

References MatrixComputeQuantity(), and AnalysisItem::r.

Referenced by Lee95bounds(), and Lee96bounds().

Here is the call graph for this function:

#### 17.18.6.12 static PetscErrorCode MatCommutatorNormF (Mat *A*, PetscTruth * *done*) [static]

Compute the Frobenius norm of the commutator $AA^t - A^t$

This computation can only be done in single-processor mode. If the commandline option (see Commandline options) "-anamod-force sequential" is specified, the matrix is gathered on processor zero to compute this quantity.

This is an expensive operation ($O(N^3)$) which can only be optimized for banded matrices. The maximum allowed bandwidth is set through CommutatorNormFAllowSqrtTimes().

See MatCommutatorNormF_seq() for the actual computation.

Definition at line 379 of file normal.c.

References AnaModGetSequentialMatrix(), GetDataID(), HASTOEXIST, id, and MatCommutatorNormF_seq().

Referenced by Commutator().

Here is the call graph for this function:

#### 17.18.6.13 static PetscErrorCode MatCommutatorNormF_seq (Mat *A*, PetscReal * *result*, PetscTruth * *done*) [static]

Compute the Frobenius norm of the commutator $AA^t - A^t$ of a sequential matrix.

This routine is accessed through MatCommutatorNormF().

Definition at line 297 of file normal.c.

References AnaModHasForcedExpensiveComputation(), MatrixComputeQuantity(), max, min, and SparseVecProd().

Referenced by MatCommutatorNormF().

Here is the call graph for this function:

#### 17.18.6.14 PetscErrorCode RegisterNormalityModules (void)

Definition at line 581 of file normal.c.

References ANALYSISDOUBLE, Commutator(), DepartureLee95(), DepartureLee96L(), DepartureLee96U(), DepartureRuhe75(), RegisterModule(), and TraceA2().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.18.6.15 static PetscErrorCode SparseVecProd (int *na*, const int * *ia*, const PetscScalar * *va*, int *nb*, const int * *ib*, const PetscScalar * *vb*, PetscScalar * *result*) [static]

Definition at line 85 of file normal.c.

Referenced by compute_tracea2_seq(), and MatCommutatorNormF_seq().

#### 17.18.6.16 static PetscErrorCode TraceA2 (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 251 of file normal.c.

References compute_tracea2(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by Lee96bounds(), and RegisterNormalityModules().

Here is the call graph for this function:

### 17.18.7 Variable Documentation

#### 17.18.7.1 int sqrt_times = 50 [static]

Definition at line 274 of file normal.c.

## 17.19 options.c File Reference

Commandline options handling.

```
#include <stdlib.h>
#include <string.h>
#include "anamod.h"
#include "petsc.h"
```

Include dependency graph for options.c:

**Defines**

- #define VALUELEN 150

### Functions

- PetscErrorCode AnaModOptionsHandling ()
- PetscErrorCode AnaModShowOptions (MPI_Comm comm)
- PetscErrorCode AnaModHasForcedSequentialComputation (PetscTruth *flg)
- PetscErrorCode AnaModGetSequentialMatrix (Mat A, Mat *Awork, PetscTruth *mem, PetscTruth *local, PetscTruth *global)
- PetscErrorCode AnaModHasForcedExpensiveComputation (PetscTruth *flg)

### Variables

- static PetscTruth single_proc = PETSC_FALSE
- static PetscTruth expensive = PETSC_FALSE

#### 17.19.1 Detailed Description

Commandline options handling.

#### 17.19.2 Commandline Options for Runtime Control

The AnaMod behaviour can be modified by commandline options.

- Certain operations can be very expensive, for instance because they need to be performed on a single processor. AnaMod will be default refuse to compute these quantities, but their computation can be forced by "`-anamod_force <option>`" where `option` is
  - `sequential`
  - `expensive`
- Each module can have options "`-anamod_<module name> <option>`". If the module has declared a function to handle such options, that function is called here. This implies that the AnaModOptionsHandling() function needs to be called after all modules have been declared.

Definition in file options.c.

#### 17.19.3 Define Documentation

##### 17.19.3.1 #define VALUELEN 150

Referenced by AnaModOptionsHandling().

### 17.19.4 Function Documentation

#### 17.19.4.1 PetscErrorCode AnaModGetSequentialMatrix (Mat *A*, Mat * *Awork*, PetscTruth * *mem*, PetscTruth * *local*, PetscTruth * *global*)

Collect the matrix on processor zero.

There is an implicit assumption here that processor zero is the one that will do the actual work.

Argument:

- `A` : input matrix
- `Awork` : pointer to the sequential matrix, this can be a pointer to the original matrix if it was already sequential; it is NULL if no forced sequential computation is asked (see Commandline options)
- `mem` : true if `Awork` is newly allocated
- `local` : true if this processor needs to do the work
- `global` : true if any processor does work; this condition is false in the case of a distributed matrix and no forced sequential operation

Definition at line 154 of file options.c.

References AnaModHasForcedSequentialComputation().

Referenced by compute_tracea2(), MatCommutatorNormF(), and MatSymmPart-NormInf().

Here is the call graph for this function:

#### 17.19.4.2 PetscErrorCode AnaModHasForcedExpensiveComputation (PetscTruth * *flg*)

Query whether certain expensive operations should be done regardless the cost.

Definition at line 200 of file options.c.

References expensive.

Referenced by MatCommutatorNormF_seq().

#### 17.19.4.3 PetscErrorCode AnaModHasForcedSequentialComputation (PetscTruth * *flg*)

Query whether parallel modules should be done on processor zero.

Definition at line 127 of file options.c.

References single_proc.

Referenced by AnaModGetSequentialMatrix().

#### 17.19.4.4 PetscErrorCode AnaModOptionsHandling (void)

Process any commandline options that were given for AnaMod. These use the regular Petsc commandline options database.

This routine handles general options and module-specific options, so it has to be called after all modules have been declared.

See DeclareCategoryOptionFunction(),GetCategoryOptionFunction(), AnaModOptionsHandling() and section Commandline Options for Runtime Control.

Definition at line 38 of file options.c.

References categories, expensive, GetCategories(), GetCategoryOptionFunction(), single_proc, and VALUELEN.

Here is the call graph for this function:

#### 17.19.4.5 PetscErrorCode AnaModShowOptions (MPI_Comm *comm*)

Display all available options. This depends on the installed modules, so you need to do the various register calls first. See Use of the analysis modules.

For options see Commandline Options for Runtime Control.

Definition at line 99 of file options.c.

References categories, and GetCategories().

Here is the call graph for this function:

### 17.19.5 Variable Documentation

#### 17.19.5.1 PetscTruth expensive = PETSC_FALSE [static]

Definition at line 24 of file options.c.

Referenced by AnaModHasForcedExpensiveComputation(), and AnaModOptionsHandling().

#### 17.19.5.2 PetscTruth single_proc = PETSC_FALSE [static]

Definition at line 24 of file options.c.

Referenced by AnaModHasForcedSequentialComputation(), and AnaModOptionsHandling().

## 17.20 petsc.c File Reference

```
#include <stdlib.h>
#include <petscoptions.h>
#include <petsc.h>
#include "nmd.h"
#include "anamod.h"
#include "anamodsalsamodules.h"
```

Include dependency graph for petsc.c:

### Functions

- static PetscErrorCode get_matrix (Mat ∗A, PetscTruth ∗success)
- PetscErrorCode analyze_matrix (Mat A, NMD_metadata nmd)
- int main (int argc, char ∗∗argv)

### 17.20.1 Function Documentation

#### 17.20.1.1 PetscErrorCode analyze_matrix (Mat *A*, NMD_metadata *nmd*)

Definition at line 47 of file petsc.c.

References categories, CategoryGetModules(), ComputeQuantity(), GetCategories(), GetDataType(), modules, and types.

Referenced by main().

Here is the call graph for this function:

#### 17.20.1.2 static PetscErrorCode get_matrix (Mat ∗ *A*, PetscTruth ∗ *success*) [static]

Definition at line 14 of file petsc.c.

Referenced by main().

#### 17.20.1.3 int main (int *argc*, char ∗∗ *argv*)

Definition at line 82 of file petsc.c.

References analyze_matrix(), AnaModDeregisterSalsaModules(), AnaModFinalize(), AnaModGetTypeMySQLName(), AnaModInitialize(), AnaModRegisterSalsaModules(), CategoryGetModules(), get_matrix(), GetCategories(), and types.

Here is the call graph for this function:

## 17.21 reporting.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include "petsc.h"
#include "anamod.h"
#include "nmd.h"
```

Include dependency graph for reporting.c:

### Functions

- static PetscErrorCode ReportAnamodContent (NMD_metadata nmd, char **rkey, char **rval, int separator)
- PetscErrorCode TabReportModules (char **rkey, int separator)
- PetscErrorCode TabReportValues (NMD_metadata nmd, char **rkey, char **rval, int separator)

#### 17.21.1 Function Documentation

##### 17.21.1.1 static PetscErrorCode ReportAnamodContent (NMD_metadata *nmd*, char ** *rkey*, char ** *rval*, int *separator*) [static]

Generate a delimited string of all module names and corresponding values.

- if the rval parameter is null, no value string is generated
- separator is a single character

Definition at line 19 of file reporting.c.

References AnaModTraceArrays(), categories, CategoryGetModules(), GetCategories(), modules, QuantityAsString(), and types.

Referenced by TabReportModules(), and TabReportValues().

Here is the call graph for this function:

#### 17.21.1.2 PetscErrorCode TabReportModules (char ** *rkey*, int *separator*)

Generate a string with `separator` delimiter listing all currently declared modules in the AnaMod system. The user needs to PetscFree() the string after use.

Definition at line 116 of file reporting.c.

References ReportAnamodContent().

Here is the call graph for this function:

#### 17.21.1.3 PetscErrorCode TabReportValues (NMD_metadata *nmd*, char ** *rkey*, char ** *rval*, int *separator*)

Generate strings with `separator` delimiter listing all currently declared modules in the AnaMod system and their values. The user needs to PetscFree() the strings after use.

Definition at line 131 of file reporting.c.

References ReportAnamodContent().

Here is the call graph for this function:

## 17.22 simple.c File Reference

Norm-like properties.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscmat.h"
#include "src/mat/impls/aij/seq/aij.h"
```

Include dependency graph for simple.c:

### Defines

- #define RC_NORM(i, j, v, w)
- #define ASSERT3(v, s, i, j) if (!(v)) SETERRQ3(1,"Violation of <%s> @ (%d,%d): %d",s,i,j);
- #define ASSERT(v, s, i, j, t) if (!(v)) SETERRQ4(1,"Violation of <%s> @ (%d,%d): %d",s,i,j,t);

### Functions

- static PetscErrorCode computetrace (Mat A)
- static PetscErrorCode Trace (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode TraceAbs (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode norm1 (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode normF (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode normInf (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode compute_dd (Mat A, PetscTruth *flg)
- static PetscErrorCode DiagonalDominance (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MatSymmPartNormInf_seq (Mat A, PetscReal *sn_r, PetscReal *an_r, PetscReal *srn_r, PetscReal *arn_r, int *nun_r, PetscTruth *done)
- static PetscErrorCode MatSymmPartNormInf (Mat A, PetscTruth *done)
- static PetscErrorCode SymmetrySNorm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SymmetryANorm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SymmetryFSNorm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SymmetryFANorm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode NUnstruct (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterSimpleModules ()
- PetscErrorCode DeRegisterSimpleModules (void)

#### 17.22.1 Detailed Description

Norm-like properties.

#### 17.22.2 Simple (normlike) quantities

The "simple" module contains simple, normlike quantities, In general, these are properties that can be calculated in time proportional to the number of nonzeros of the matrix. Note that this excludes the 2-norm.

### 17.22.3 Usage

Activate this module with

PetscErrorCode RegisterSimpleModules();

Compute these elements with

ComputeQuantity("simple",<element>,A,(AnalysisItem*)&res,&flg);

Available elements are:

- "trace" : Trace, sum of diagonal elements; see Trace().
- "trace-abs" : Trace Absolute, sum of absolute values of diagonal elements; see TraceAbs().
- "norm1" : 1-Norm, maximum column sum of absolute element sizes; see norm1().
- "normInf" : Infinity-Norm, maximum row sum of absolute element sizes; see normInf().
- "normF" : Frobenius-norm, sqrt of sum of elements squared; see normF().
- "diagonal-dominance" : least positive or most negative value of diagonal element minus sum of absolute off-diagonal elements; see diagonaldominance().
- "symmetry-snorm" : infinity norm of symmetric part; see SymmetrySNorm(). This element and all following are sequential; see Commandline options;
- "symmetry-anorm" : infinity norm of anti-symmetric part; see SymmetryANorm().
- "symmetry-fsnorm" : Frobenius norm of symmetric part; see SymmetryFSNorm().
- "symmetry-fanorm" : Frobenius norm of anti-symmetric part; see SymmetryFANorm().
- "n-struct-unsymm" : number of structurally unsymmetric elements; see NUnstruct(). (this is calculated here, but declared as an element of the Structural properties category.)

Definition in file simple.c.

### 17.22.4 Define Documentation

#### 17.22.4.1 #define ASSERT(v, s, i, j, t) if (!(v)) SETERRQ4(1,"Violation of <%s> @ (%d,%d): %d",s,i,j,t);

Referenced by MatSymmPartNormInf_seq().

#### 17.22.4.2 #define ASSERT3(v, s, i, j) if (!(v)) SETERRQ3(1,"Violation of <%s> @ (%d,%d): %d",s,i,j);

Referenced by MatSymmPartNormInf_seq().

#### 17.22.4.3 #define RC_NORM(i, j, v, w)

**Value:**

```
{                                                                    \
    PetscReal sum = PetscAbsScalar(v+w)/2, dif = PetscAbsScalar(v-w)/2; \
    ASSERT3(!(i==j && v!=w),"same elements on diagonal",i,j);          \
    sronorm[i] += sum; if (i!=j) sronorm[j] += sum;                  \
    aronorm[i] += dif; if (i!=j) aronorm[j] += dif;                  \
    if (i==j) snorm += sum*sum;                                      \
    else {snorm += 2*sum*sum; anorm += 2*dif*dif;}                   \
  }
```

Referenced by MatSymmPartNormInf_seq().

### 17.22.5 Function Documentation

#### 17.22.5.1 static PetscErrorCode compute_dd (Mat *A*, PetscTruth * *flg*) [static]

Compute the diagonal dominance of a matrix: minimum value of diagonal element (not absolute!) minus off-diagonal elements absolute.

$$\min_i a_{ii} - \sum_{j \neq i} |a_{ij}|$$

Definition at line 246 of file simple.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by DiagonalDominance().

Here is the call graph for this function:

#### 17.22.5.2 static PetscErrorCode computetrace (Mat *A*) [static]

This is the computational routine for trace calculation. It computes both the trace and the absolute trace, that is, using absolute values of matrix elements.

A Petsc vector is created and destroyed.

Definition at line 63 of file simple.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by Trace(), and TraceAbs().

Here is the call graph for this function:

#### 17.22.5.3 PetscErrorCode DeRegisterSimpleModules (void)

Definition at line 681 of file simple.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.22.5.4 static PetscErrorCode DiagonalDominance (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 282 of file simple.c.

References compute_dd(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.5 static PetscErrorCode MatSymmPartNormInf (Mat *A*, PetscTruth * *done*) [static]

This is the computational routine for all quantities relating to symmetric and anti-symmetric parts of the matrix.

This computation can only be done in single-processor mode. If the commandline option (see Commandline options) `"-anamod-force sequential"` is specified, the matrix is gathered on processor zero to compute this quantity.

See MatSymmPartNormInf_seq() for the actual computation.

Definition at line 432 of file simple.c.

References AnaModGetSequentialMatrix(), GetDataID(), HASTOEXIST, id, and MatSymmPartNormInf_seq().

Referenced by NUnstruct(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), and SymmetrySNorm().

Here is the call graph for this function:

#### 17.22.5.6 static PetscErrorCode MatSymmPartNormInf_seq (Mat *A*, PetscReal * *sn_r*, PetscReal * *an_r*, PetscReal * *srn_r*, PetscReal * *arn_r*, int * *nun_r*, PetscTruth * *done*) [static]

This is where the real work of MatSymmPartNormInf() is done

A few temporary arrays of the size of a vector are allocated and freed. Otherwise this is just a whole bunch of pointer chasing.

Definition at line 318 of file simple.c.

References ASSERT, ASSERT3, and RC_NORM.

Referenced by MatSymmPartNormInf().

#### 17.22.5.7 static PetscErrorCode norm1 (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute the 1-norm of a matrix.

Definition at line 159 of file simple.c.

References GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.8 static PetscErrorCode normF (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute the Frobenius norm of a matrix.

Definition at line 186 of file simple.c.

References GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.9 static PetscErrorCode normInf (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute the infinity-norm of a matrix.

Definition at line 213 of file simple.c.

References GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.10 static PetscErrorCode NUnstruct (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the number of structurally unsymmetric elements of a matrix. The actual computation is done in MatSymmPartNormInf(); see that routine for remarks on parallelism.

The "n-struct-unsymm" feature is declared to be in category "structure", rather than simple.

Definition at line 617 of file simple.c.

References GetDataID(), HASTOEXIST, AnalysisItem::i, id, and MatSymmPartNormInf().

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.11 PetscErrorCode RegisterSimpleModules (void)

Declare norm-like modules that can be performed with simple calculations.

Definition at line 644 of file simple.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, DiagonalDominance(), norm1(), normF(), normInf(), NUnstruct(), RegisterModule(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), and TraceAbs().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.22.5.12 static PetscErrorCode SymmetryANorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the infinity norm of the antisymmetric part of the matrix. The

actual computation is done in MatSymmPartNormInf(); see that routine for remarks on parallelism.

Definition at line 527 of file simple.c.

References GetDataID(), HASTOEXIST, id, MatSymmPartNormInf(), and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.13 static PetscErrorCode SymmetryFANorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the Frobenius norm of the antisymmetric part of the matrix. The actual computation is done in MatSymmPartNormInf(); see that routine for remarks on parallelism.

Definition at line 585 of file simple.c.

References GetDataID(), HASTOEXIST, id, MatSymmPartNormInf(), and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.14 static PetscErrorCode SymmetryFSNorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the Frobenius norm of the symmetric part of the matrix. The actual computation is done in MatSymmPartNormInf(); see that routine for remarks on parallelism.

Definition at line 556 of file simple.c.

References GetDataID(), HASTOEXIST, id, MatSymmPartNormInf(), and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.15 static PetscErrorCode SymmetrySNorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the infinity norm of the symmetric part of the matrix. The actual computation is done in MatSymmPartNormInf(); see that routine for remarks on parallelism.

Definition at line 498 of file simple.c.

References GetDataID(), HASTOEXIST, id, MatSymmPartNormInf(), and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.16 static PetscErrorCode Trace (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the trace of a matrix; the computation is done in computetrace().

Definition at line 106 of file simple.c.

References computetrace(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

#### 17.22.5.17 static PetscErrorCode TraceAbs (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Compute and store the absolute trace of a matrix; the computation is done in computetrace().

Definition at line 135 of file simple.c.

References computetrace(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterSimpleModules().

Here is the call graph for this function:

## 17.23 spectrum.c File Reference

Various estimates of spectrum related quantities.

```
#include <stdlib.h>
#include "string.h"
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscksp.h"
#include "petscmat.h"
```

Include dependency graph for spectrum.c:

### Functions

- PetscErrorCode SpectrumComputePreconditionedSpectrum ()
- PetscErrorCode SpectrumComputeUnpreconditionedSpectrum ()
- static PetscErrorCode fivestepplan (KSP solver, PetscInt it, PetscReal err, KSP-ConvergedReason ∗reason, void ∗ctx)
- static PetscErrorCode compute_eigenvalues_petsc (Mat A, PetscReal ∗∗r, PetscReal ∗∗c, int ∗neig, PetscReal ∗rx, PetscReal ∗rm, PetscTruth ∗success, char ∗∗reason)
- static PetscErrorCode compute_eigenvalues (Mat A, PetscReal ∗∗r, PetscReal ∗∗c, int ∗neig, PetscTruth ∗success, char ∗∗reason)

- static PetscErrorCode compute_singularvalues (Mat A, PetscTruth *success, char **reason)
- static PetscErrorCode compute_ellipse_from_Ritz_values (Mat A, PetscReal *r, PetscReal *c, int neig)
- static PetscErrorCode NRitzValues (AnaModNumericalProblem prob, AnalysisItem *iv, int *lv, PetscTruth *flg)
- static PetscErrorCode RitzValuesR (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode RitzValuesC (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SpectrumAX (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SpectrumAY (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SpectrumCX (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SpectrumCY (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode Kappa (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode PosFraction (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SigmaMax (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SigmaMin (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MinEVbyMagIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyRealIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyImRe (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxEVbyImIm (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode SpectrumOptions (char *opt)
- PetscErrorCode RegisterSpectrumModules ()
- PetscErrorCode DeregisterSpectrumModules ()

### Variables

- static PetscTruth trace_spectrum = PETSC_FALSE
- static PetscTruth trace_hessenberg = PETSC_FALSE
- static int use_prec = 0

### 17.23.1 Detailed Description

Various estimates of spectrum related quantities.

### 17.23.2 Spectral properties

The spectrum (eigenvalues and singular values) are both very informative about a matrix, and hard to compute. The spectrum module provides various estimates that are derived from running the system through a GMRES iteration for a few dozen iterations.

#### 17.23.2.1 Usage

Activate this module with

PetscErrorCode RegisterSpectrumModules();

Compute these elements with

ComputeQuantity("spectrum",<element>,A,(AnalysisItem*)&res,&reslen,&flg);

These elements can be computed for both a preconditioned and unpreconditioned matrix. Switch between the two with

PetscErrorCode SpectrumComputePreconditionedSpectrum(); PetscErrorCode SpectrumComputeUnpreconditionedSpectrum();

In the preconditioned case a preconditioner has to have been defined in the Petsc options database with prefix "-ana"; for instance "-ana_pc_type jacobi".

Available elements are:

- "n-ritz-values" : number of stored Ritz values
- "ritz-values-r" : real parts of stored Ritz values
- "ritz-values-c" : complex parts of stored Ritz values
- "ellipse-ax" : size of $x$-axis of the enclosing ellipse
- "ellipse-ay" : size of $y$-axis of the enclosing ellipse

- "ellipse-cx" : $x$-coordinate of the center of the enclosing ellipse
- "ellipse-cy" : $y$-coordinate of the center of the enclosing ellipse
- "kappa" : estimated condition number
- "positive-fraction" : fraction of computed eigenvalues that has positive real part
- "sigma-max" : maximum singular value
- "sigma-min" : minimum singular value
- "lambda-max-by-magnitude-re" : real part of maximum lambda by magnitude
- "lambda-max-by-magnitude-im" : imaginary part of maximum lambda by magnitude
- "lambda-min-by-magnitude-re" : real part of minimum lambda by magnitude
- "lambda-min-by-magnitude-im" : imaginary part of minimum lambda by magnitude
- "lambda-max-by-real-part-re" : real part of maximum lambda by real-part
- "lambda-max-by-real-part-im" : imaginary part of maximum lambda by real-part

#### 17.23.2.2 Theory

Jacobi methods, such as GMRES, give a reduction $AV=VH$ of $A$ to upper Hessenberg form $H$. The reduction is defined by the Krylov basis $V$. While a full reduction gives a Hessenberg matrix that has the exact same eigenvalues as $A$, using $V$ with a limited number of columns $n<N$) gives an $H$ with eigenvalues that approximate those of $A$. In particular, the outer eigenvalues are known to be fairly accurate even for modest values of $n$.

This process is used in the Petsc calls KSPSetComputeEigenvalues() and KSPSetComputeSingularValues(), which we use in the compute_eigenvalues() function. Alternatively, if the SLEPc library is available, that can be used for a similar but probably more accurate computation.

#### 17.23.2.3 Tracing

See SpectrumOptions().

Definition in file spectrum.c.

### 17.23.3 Function Documentation

#### 17.23.3.1 static PetscErrorCode compute_eigenvalues (Mat *A*, PetscReal ** *r*, PetscReal ** *c*, int * *neig*, PetscTruth * *success*, char ** *reason*) [static]

Compute a number of eigenvalues (returned as separate real and imaginary parts) of a matrix.

Parameters:

- success : PETSC_TRUE or PETSC_FALSE
- reason : explanation of the success parameter. This is a static string; does not need to be freed.
- r,c : arrays of real and imaginary part of the eigenvalues. If success is false, these are NULL. Otherwise, the calling environment needs to free them at some point.
- neig : length of the `r` and `c` arrays. This is zero if success is false.

Definition at line 457 of file spectrum.c.

References compute_eigenvalues_petsc(), GetDataID(), HASTOEXIST, and id.

Referenced by Kappa(), NRitzValues(), PosFraction(), RitzValuesC(), RitzValuesR(), SpectrumAX(), SpectrumAY(), SpectrumCX(), and SpectrumCY().

Here is the call graph for this function:

#### 17.23.3.2 static PetscErrorCode compute_eigenvalues_petsc (Mat *A*, PetscReal ** *r*, PetscReal ** *c*, int * *neig*, PetscReal * *rx*, PetscReal * *rm*, PetscTruth * *success*, char ** *reason*) [static]

Definition at line 295 of file spectrum.c.

References AnaModTraceMessage(), fivestepplan(), trace_hessenberg, trace_-spectrum, and use_prec.

Referenced by compute_eigenvalues(), and compute_singularvalues().

Here is the call graph for this function:

#### 17.23.3.3 static PetscErrorCode compute_ellipse_from_Ritz_values (Mat *A*, PetscReal * *r*, PetscReal * *c*, int *neig*) [static]

Definition at line 534 of file spectrum.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by Kappa(), NRitzValues(), PosFraction(), RitzValuesC(), RitzValuesR(), SpectrumAX(), SpectrumAY(), SpectrumCX(), and SpectrumCY().

Here is the call graph for this function:

#### 17.23.3.4 static PetscErrorCode compute_singularvalues (Mat *A*, PetscTruth * *success*, char ** *reason*) [static]

Compute a number of singular values of a matrix.

Parameters:

- success : PETSC_TRUE or PETSC_FALSE
- reason : explanation of the success parameter. This is a static string; does not need to be freed.

Definition at line 502 of file spectrum.c.

References compute_eigenvalues_petsc(), GetDataID(), HASTOEXIST, and id.

Referenced by SigmaMax(), and SigmaMin().

Here is the call graph for this function:

#### 17.23.3.5 PetscErrorCode DeregisterSpectrumModules (void)

Definition at line 1401 of file spectrum.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.23.3.6 static PetscErrorCode fivestepplan (KSP *solver*, PetscInt *it*, PetscReal *err*, KSPConvergedReason * *reason*, void * *ctx*) [static]

Definition at line 113 of file spectrum.c.

Referenced by compute_eigenvalues_petsc().

#### 17.23.3.7 static PetscErrorCode Kappa (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 882 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.8 static PetscErrorCode MaxEVbyImIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1268 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.9 static PetscErrorCode MaxEVbyImRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1230 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.10 static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1037 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.11 static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 998 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.12 static PetscErrorCode MaxEVbyRealIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1192 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.13 static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1154 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.14 static PetscErrorCode MinEVbyMagIm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1115 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.15 static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 1076 of file spectrum.c.

References ComputeQuantity(), GetDataID(), HASTOEXIST, AnalysisItem::r, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.16 static PetscErrorCode NRitzValues (AnaModNumericalProblem *prob*, AnalysisItem * *iv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 608 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.17 static PetscErrorCode PosFraction (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 914 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.18 PetscErrorCode RegisterSpectrumModules (void)

Definition at line 1329 of file spectrum.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTEGER, DeclareCategoryOptionFunction(), Kappa(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), NRitzValues(), PosFraction(), RegisterModule(), RitzValuesC(), RitzValuesR(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumCX(), SpectrumCY(), and SpectrumOptions().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.23.3.19 static PetscErrorCode RitzValuesC (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 681 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, id, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.20 static PetscErrorCode RitzValuesR (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 641 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, id, and AnalysisItem::rr.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.21 static PetscErrorCode SigmaMax (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 946 of file spectrum.c.

References compute_singularvalues(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.22 static PetscErrorCode SigmaMin (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 972 of file spectrum.c.

References compute_singularvalues(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.23 static PetscErrorCode SpectrumAX (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 753 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.24 static PetscErrorCode SpectrumAY (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 785 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.25 PetscErrorCode SpectrumComputePreconditionedSpectrum (void)

Definition at line 94 of file spectrum.c.

References use_prec.

#### 17.23.3.26 PetscErrorCode SpectrumComputeUnpreconditionedSpectrum (void)

Definition at line 103 of file spectrum.c.

References use_prec.

Referenced by AnaModRegisterSalsaModules().

#### 17.23.3.27 static PetscErrorCode SpectrumCX (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 817 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.28 static PetscErrorCode SpectrumCY (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 850 of file spectrum.c.

References compute_eigenvalues(), compute_ellipse_from_Ritz_values(), GetDataID(), HASTOEXIST, and AnalysisItem::r.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:

#### 17.23.3.29 static PetscErrorCode SpectrumOptions (char * *opt*) [static]

Spectrum options can be set with "-anamod_spectrum <someoption>". Here we process the options. The possibilities are:

- "trace_spectrum" : list the computed spectrum values on the screen
- "dump_hessenberg" : generate a Matlab file of the Hessenberg matrix

See Commandline Options for Runtime Control.

Definition at line 1313 of file spectrum.c.

References trace_hessenberg, and trace_spectrum.

Referenced by RegisterSpectrumModules().

### 17.23.4 Variable Documentation

#### 17.23.4.1 PetscTruth trace_hessenberg = PETSC_FALSE [static]

Definition at line 89 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), and SpectrumOptions().

#### 17.23.4.2 PetscTruth trace_spectrum = PETSC_FALSE [static]

Definition at line 88 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), and SpectrumOptions().

#### 17.23.4.3 int use_prec = 0 [static]

Definition at line 91 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), SpectrumComputePreconditionedSpectrum(), and SpectrumComputeUnpreconditionedSpectrum().

## 17.24 stats.c File Reference

Statistics on the AnaMod system.

```
#include <stdlib.h>
#include "anamod.h"
#include "petscerror.h"
```

Include dependency graph for stats.c:

### Functions

- static PetscErrorCode Version (AnaModNumericalProblem prob, AnalysisItem *rv, int *l, PetscTruth *flg)
- PetscErrorCode RegisterStatsModules ()

### 17.24.1 Detailed Description

Statistics on the AnaMod system.

### 17.24.2 Statistics on the AnaMod system

The stats module needs to be enabled explicitly.

Definition in file stats.c.

### 17.24.3 Function Documentation

#### 17.24.3.1 PetscErrorCode RegisterStatsModules ()

Declare statistics modules

Definition at line 34 of file stats.c.

References ANALYSISSTRING, RegisterModule(), and Version().

Here is the call graph for this function:

#### 17.24.3.2 static PetscErrorCode Version (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *l*, PetscTruth * *flg*) [static]

The AnaMod format version string

Definition at line 19 of file stats.c.

References ANAMOD_FORMAT_VERSION, AnalysisItem::c, GetDataID(), HASTOEXIST, and id.

Referenced by RegisterStatsModules().

Here is the call graph for this function:

## 17.25 structure.c File Reference

Structural properties of a matrix.

```
#include <stdlib.h>
#include "anamod.h"
```

```
#include "anamodutils.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscmat.h"
```

Include dependency graph for structure.c:

### Functions

- static PetscErrorCode nRows (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode Symmetry (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode compute_nnz_structure (AnaModNumericalProblem prob)
- static PetscErrorCode NNonZeros (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MaxNNonZerosPerRow (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode MinNNonZerosPerRow (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode LBandWidth (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode RBandWidth (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode LeftSkyline (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode RightSkyline (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode compute_dummy_rows (Mat A)
- static PetscErrorCode NDummyRows (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DummyRowsKind (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)

- static PetscErrorCode DummyRows (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode compute_posdiag (AnaModNumericalProblem prob)
- static PetscErrorCode DiagZeroStart (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DiagDefinite (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode regularblocks (AnaModNumericalProblem prob)
- static PetscErrorCode BlockSize (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterStructureModules ()
- PetscErrorCode DeRegisterStructureModules (void)

### 17.25.1 Detailed Description

Structural properties of a matrix.

This file defines the routines for computing structural properties of a matrix, as well as the routine RegisterStructureModules() that registers them.

### 17.25.2 Structural properties

This category computes matrix properties that are only a function of the nonzero structure. Thus, these properties will likely stay invariant during a nonlinear solve process, or while time-stepping a system of equations.

### 17.25.3 Usage

Activate this module with

PetscErrorCode RegisterStructureModules();

Compute these elements with

ComputeQuantity("structure",<element>,A,(void*)&res,&flg);

Available elements are:

- "nrows" : number of rows in the matrix
- "nnzeros" : number of nonzero elements in the matrix
- "max-nnzeros-per-row" : maximum number of nonzeros per row
- "min-nnzeros-per-row" : minimum number of nonzeros per row

- "left-bandwidth" : $\max_i\{i-j: a_{ij} \neq 0\}$
- "right-bandwidth" : $\max_i\{j-i: a_{ij} \neq 0\}$
- "n-dummy-rows" : number of rows with only one element
- "dummy-rows-kind" : 0 if all dummy rows have a one on the diagonal, 1 if the value is not one, 2 if not on the diagonal
- "diag-zerostart" : $\min\{i: \forall_{j>i} a_{jj} = 0\}$
- "diag-definite" : 1 if diagonal positive, 0 otherwise
- "block-size" : integer size of blocks that comprise matrix block structure, 1 in the general case
- "symmetry" : 1 for symmetric matrix, 0 otherwise
- "n-struct-unsymm" : number of structurally unsymmetric elements (this is actually calculated in the simple.c)

Definition in file structure.c.

### 17.25.4 Function Documentation

#### 17.25.4.1 static PetscErrorCode BlockSize (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 664 of file structure.c.

References GetDataID(), HASTOEXIST, AnalysisItem::i, id, and regularblocks().

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.2 static PetscErrorCode compute_dummy_rows (Mat *A*) [static]

Definition at line 371 of file structure.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by DummyRows(), DummyRowsKind(), and NDummyRows().

Here is the call graph for this function:

#### 17.25.4.3 static PetscErrorCode compute_nnz_structure (AnaModNumericalProblem *prob*) [static]

Definition at line 110 of file structure.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by LBandWidth(), LeftSkyline(), MaxNNonZerosPerRow(), MinNNonZerosPerRow(), NNonZeros(), RBandWidth(), and RightSkyline().

Here is the call graph for this function:

#### 17.25.4.4 static PetscErrorCode compute_posdiag (AnaModNumericalProblem *prob*) [static]

Definition at line 518 of file structure.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by DiagDefinite(), and DiagZeroStart().

Here is the call graph for this function:

#### 17.25.4.5 PetscErrorCode DeRegisterStructureModules (void)

Definition at line 738 of file structure.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.25.4.6 static PetscErrorCode DiagDefinite (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 586 of file structure.c.

References compute_posdiag(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.7 static PetscErrorCode DiagZeroStart (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 563 of file structure.c.

References compute_posdiag(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.8 static PetscErrorCode DummyRows (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 485 of file structure.c.

References compute_dummy_rows(), GetDataID(), HASTOEXIST, id, and AnalysisItem::ii.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.9 static PetscErrorCode DummyRowsKind (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 462 of file structure.c.

References compute_dummy_rows(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.10 static PetscErrorCode LBandWidth (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 269 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.11 static PetscErrorCode LeftSkyline (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 315 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, id, and AnalysisItem::ii.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.12 static PetscErrorCode MaxNNonZerosPerRow (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 223 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.13 static PetscErrorCode MinNNonZerosPerRow (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 246 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.14 static PetscErrorCode NDummyRows (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 439 of file structure.c.

References compute_dummy_rows(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.15 static PetscErrorCode NNonZeros (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 200 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.16 static PetscErrorCode nRows (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 54 of file structure.c.

References GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.17 static PetscErrorCode RBandWidth (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 292 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.18 PetscErrorCode RegisterStructureModules (void)

Definition at line 689 of file structure.c.

References ANALYSISINTARRAY, ANALYSISINTEGER, BlockSize(), DiagDefinite(), DiagZeroStart(), DummyRows(), DummyRowsKind(), LBandWidth(), LeftSkyline(), MaxNNonZerosPerRow(), MinNNonZerosPerRow(), NDummyRows(), NNonZeros(), nRows(), RBandWidth(), RegisterModule(), RightSkyline(), and Symmetry().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.25.4.19 static PetscErrorCode regularblocks (AnaModNumericalProblem *prob*) [static]

Definition at line 612 of file structure.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by BlockSize().

Here is the call graph for this function:

#### 17.25.4.20 static PetscErrorCode RightSkyline (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 342 of file structure.c.

References compute_nnz_structure(), GetDataID(), HASTOEXIST, id, and AnalysisItem::ii.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

#### 17.25.4.21 static PetscErrorCode Symmetry (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

Definition at line 78 of file structure.c.

References GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterStructureModules().

Here is the call graph for this function:

## 17.26 tracing.c File Reference

Trace routines for the anamod package.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include "petsc.h"
#include "anamod.h"
```

Include dependency graph for tracing.c:

### Functions

- PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(∗fn)(void ∗, char ∗, va_list))
- PetscErrorCode AnaModDeclareTraceContext (void ∗ctx)
- PetscErrorCode AnaModSetTraceArrays (PetscTruth f)
- PetscErrorCode AnaModTraceArrays (PetscTruth ∗f)
- PetscErrorCode AnaModTraceMessage (const char ∗fmt,...)
- PetscErrorCode AnaModHasTrace (PetscTruth ∗flg)

### Variables

- static PetscErrorCode(∗ anamodtrace )(void ∗, char ∗fmt, va_list) = NULL
- static size_t anamodtracectx = (size_t)NULL
- static PetscTruth anamodtracearrays = PETSC_FALSE

### 17.26.1 Detailed Description

Trace routines for the anamod package.

### 17.26.2 Tracing the analysis modules

The AnaMod package does not by default print out anything, other than severe error messages (Petsc macro SETERRQ) that accompany an abort.

However, you can specify a trace function, which can further be tuned by specifying a trace context.

See AnaModDeclareTraceFunction(), AnaModDeclareTraceContext(), AnaModTraceMessage(), AnaModSetTraceArrays(), AnaModTraceArrays().

Definition in file tracing.c.

### 17.26.3 Function Documentation

#### 17.26.3.1 PetscErrorCode AnaModDeclareTraceContext (void * *ctx*)

Definition at line 69 of file tracing.c.

References anamodtracectx.

#### 17.26.3.2 PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(*)(void *, char *, va_list) *fn*)

Specify a trace function.

The trace function has a prototype

```
PetscErrorCode tracefunction(void*,char*,va_list)
```

which means that it has an arbitrary number of arguments, much like `printf`. The first argument is a context, which can be set by AnaModDeclareTraceContext().

Here is an example of how you would write a trace function:

```
#include <stdarg.h>
PetscErrorCode tracefunction(void *ctx,char *fmt,va_list argp)
{
  char *prefix = (char*)ctx;
  PetscFunctionBegin;
  printf("%s ",prefix);
  vprintf(fmt, argp);
  PetscFunctionReturn(0);
}
```

Consult `string.h` (probably in `/usr/include`) to see which "v" versions of `printf` are available.

You can undeclare a trace function by passing NULL.

Definition at line 56 of file tracing.c.

References anamodtrace.

#### 17.26.3.3 PetscErrorCode AnaModHasTrace (PetscTruth * *flg*)

Test whether a trace function has been declared; see AnaModDeclareTraceFunction(). Normally you would use AnaModTraceMessage() which performs this test internally, but this function can be useful if a large amount of processing has to be performed to construct the trace message to begin with.

Definition at line 127 of file tracing.c.

References anamodtrace.

Referenced by ComputeQuantity(), and RetrieveQuantity().

#### 17.26.3.4 PetscErrorCode AnaModSetTraceArrays (PetscTruth *f*)

Definition at line 82 of file tracing.c.

References anamodtracearrays.

#### 17.26.3.5 PetscErrorCode AnaModTraceArrays (PetscTruth * *f*)

Definition at line 95 of file tracing.c.

References anamodtracearrays.

Referenced by ComputeQuantity(), ReportAnamodContent(), and RetrieveQuantity().

#### 17.26.3.6 PetscErrorCode AnaModTraceMessage (const char * *fmt*, ...)

This function prints a trace message if a trace function has been declared; see AnaMod-DeclareTraceFunction().

Definition at line 107 of file tracing.c.

References anamodtrace, and anamodtracectx.

Referenced by compute_eigenvalues_petsc(), ComputeQuantity(), and RetrieveQuantity().

### 17.26.4 Variable Documentation

#### 17.26.4.1 PetscErrorCode(∗ anamodtrace)(void ∗, char ∗fmt, va_list) = NULL [static]

Referenced by AnaModDeclareTraceFunction(), AnaModHasTrace(), and AnaModTraceMessage().

#### 17.26.4.2 PetscTruth anamodtracearrays = PETSC_FALSE [static]

Definition at line 25 of file tracing.c.

Referenced by AnaModSetTraceArrays(), and AnaModTraceArrays().

#### 17.26.4.3 size_t anamodtracectx = (size_t)NULL [static]

Definition at line 24 of file tracing.c.

Referenced by AnaModDeclareTraceContext(), and AnaModTraceMessage().

## 17.27 utils.c File Reference

```
#include <stdlib.h>
#include "petsc.h"
#include "anamodutils.h"
```

Include dependency graph for utils.c:

### Functions

- int MPIAllGatherIntV (int *array, int n, int **Array, int *N, MPI_Comm comm)

#### 17.27.1 Detailed Description

Definition in file utils.c.

#### 17.27.2 Function Documentation

##### 17.27.2.1 int MPIAllGatherIntV (int * *array*, int *n*, int ** *Array*, int * *N*, MPI_Comm *comm*)

Definition at line 8 of file utils.c.

Referenced by GetUpBiDiagSplits(), and MatSplitPoints().

## 17.28 variance.c File Reference

Various estimates of how 'wild' a matrix is.

```
#include <stdlib.h>
#include "anamod.h"
#include "anamodsalsamodules.h"
#include "petscerror.h"
#include "petscmat.h"
```

Include dependency graph for variance.c:

### Functions

- static PetscErrorCode ComputeVariability (Mat A)
- static PetscErrorCode RowVariability (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode ColVariability (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode ComputeDiagonal (Mat A)
- static PetscErrorCode DiagonalAverage (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DiagonalVariance (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- static PetscErrorCode DiagonalSign (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscTruth *flg)
- PetscErrorCode RegisterVarianceModules ()
- PetscErrorCode DeRegisterVarianceModules (void)

#### 17.28.1 Detailed Description

Various estimates of how 'wild' a matrix is.

#### 17.28.2 Measurements of element variance

This module contains various estimates of how different elements in a matrix are. These measures are completely heuristic, and do not relate to any known mathematical theory.

#### 17.28.3 Usage

Activate this module with

PetscErrorCode RegisterVarianceModules();

Compute these elements with

ComputeQuantity("variance",<element>,A,(AnalysisItem*)&res,&flg);

Available elements are:

- "row-variability" : $\max_i \log_{10} \frac{\max_j |a_{ij}|}{\min_j |a_{ij}|}$, computed by RowVariability()
- "col-variability" : $\max_j \log_{10} \frac{\max_i |a_{ij}|}{\min_i |a_{ij}|}$, computed by ColVariability()
- "diagonal-average" : average value of absolute diagonal elements, by DiagonalAverage()
- "diagonal-variance" : standard deviation of diagonal average, by DiagonalVariance()
- "diagonal-sign" : indicator of diagonal sign pattern, by DiagonalSign(): -10 all zero, -2 all negative, -1 nonpositive, 0 indefinite, 1 nonnegative, 2 all positive

Definition in file variance.c.

### 17.28.4 Function Documentation

#### 17.28.4.1 static PetscErrorCode ColVariability (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

This routine relies on a call to ComputeVariability()

Definition at line 146 of file variance.c.

References ComputeVariability(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterVarianceModules().

Here is the call graph for this function:

#### 17.28.4.2 static PetscErrorCode ComputeDiagonal (Mat *A*) [static]

This is a computational routine; it computes the value of several modules

Definition at line 173 of file variance.c.

References DIAGONAL_INDEFINITE, DIAGONAL_NEGATIVE, DIAGONAL_-NONNEGATIVE, DIAGONAL_NONPOSITIVE, DIAGONAL_POSITIVE, DIAGONAL_ZERO, GetDataID(), HASTOEXIST, and id.

Referenced by DiagonalAverage(), DiagonalSign(), and DiagonalVariance().

Here is the call graph for this function:

#### 17.28.4.3 static PetscErrorCode ComputeVariability (Mat *A*) [static]

Variability in rows and columns.

This is a computational routine.

Definition at line 42 of file variance.c.

References GetDataID(), HASTOEXIST, and id.

Referenced by ColVariability(), and RowVariability().

Here is the call graph for this function:

#### 17.28.4.4 PetscErrorCode DeRegisterVarianceModules (void)

Definition at line 333 of file variance.c.

Referenced by AnaModDeregisterSalsaModules().

#### 17.28.4.5 static PetscErrorCode DiagonalAverage (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

This routine relies on a call to ComputeDiagonal()

Definition at line 237 of file variance.c.

References ComputeDiagonal(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterVarianceModules().

Here is the call graph for this function:

#### 17.28.4.6 static PetscErrorCode DiagonalSign (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

This routine relies on a call to ComputeDiagonal()

Definition at line 289 of file variance.c.

References ComputeDiagonal(), GetDataID(), HASTOEXIST, AnalysisItem::i, and id.

Referenced by RegisterVarianceModules().

Here is the call graph for this function:

#### 17.28.4.7 static PetscErrorCode DiagonalVariance (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

This routine relies on a call to ComputeDiagonal()

Definition at line 263 of file variance.c.

References ComputeDiagonal(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterVarianceModules().

Here is the call graph for this function:

#### 17.28.4.8 PetscErrorCode RegisterVarianceModules (void)

Definition at line 311 of file variance.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, ColVariability(), DiagonalAverage(), DiagonalSign(), DiagonalVariance(), RegisterModule(), and RowVariability().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:

#### 17.28.4.9 static PetscErrorCode RowVariability (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscTruth * *flg*) [static]

This routine relies on a call to ComputeVariability()

Definition at line 121 of file variance.c.

References ComputeVariability(), GetDataID(), HASTOEXIST, id, and AnalysisItem::r.

Referenced by RegisterVarianceModules().

Here is the call graph for this function:

# Index