

AnaMod: a package for numerical Analysis Modules

Generated by Doxygen 1.7.6.1

Tue Jul 8 2014 09:02:40

Contents

1	AnaMod : an Analysis Modules library	1
1.1	Introduction	1
2	Matrix interface to AnaMod	2
3	Feature sets	3
4	Installation and compiling programs	4
4.1	Installation	4
4.2	Compilation of user programs	4
5	Use of the analysis modules	4
5.1	Library Initialization	4
5.2	Category registration	5
5.3	Quantity computation	5
5.4	types	5
6	List of the available modules	5
7	Customization	6
8	Array type handling	6
9	Attached quantities in Petsc	7
10	Bulk computation	7
11	Commandline options	8
12	Module Index	8
12.1	Modules	8
13	Data Structure Index	9
13.1	Data Structures	9

14 File Index	9
14.1 File List	9
15 Module Documentation	11
15.1 Files with defined categories	11
15.2 User-accessible functions	13
15.3 Functions only of use to the implementation	14
16 Data Structure Documentation	15
16.1 AnalysisDataTypeArray_ Struct Reference	15
16.1.1 Detailed Description	15
16.1.2 Field Documentation	15
16.2 AnalysisItem Union Reference	16
16.2.1 Detailed Description	16
16.2.2 Field Documentation	16
16.3 AnalysisItemArray_ Struct Reference	18
16.3.1 Detailed Description	19
16.3.2 Field Documentation	19
16.4 categoryobject_ Struct Reference	20
16.4.1 Detailed Description	21
16.4.2 Field Documentation	21
16.5 componentobject_ Struct Reference	22
16.5.1 Detailed Description	22
16.5.2 Field Documentation	22
16.6 FeatureSet_ Struct Reference	24
16.6.1 Detailed Description	24
16.6.2 Field Documentation	25
16.7 FeatureValues_ Struct Reference	26
16.7.1 Detailed Description	27
16.7.2 Field Documentation	27
16.8 IntArray_ Struct Reference	27
16.8.1 Detailed Description	28

16.8.2	Field Documentation	28
16.9	StringArray_ Struct Reference	28
16.9.1	Detailed Description	29
16.9.2	Field Documentation	29
17	File Documentation	30
17.1	anamatrix.c File Reference	30
17.1.1	Function Documentation	30
17.2	anamatrix.h File Reference	31
17.2.1	Function Documentation	32
17.3	anamod.h File Reference	33
17.3.1	Detailed Description	37
17.3.2	Define Documentation	38
17.3.3	Typedef Documentation	39
17.3.4	Enumeration Type Documentation	39
17.3.5	Function Documentation	39
17.4	anamoddriver.c File Reference	73
17.4.1	Function Documentation	74
17.5	anamodsalsa.c File Reference	76
17.5.1	Function Documentation	77
17.6	anamodsalsamodules.h File Reference	82
17.6.1	Detailed Description	83
17.6.2	Define Documentation	83
17.6.3	Function Documentation	84
17.7	anamodtypes.h File Reference	93
17.7.1	Detailed Description	94
17.7.2	Define Documentation	94
17.7.3	Typedef Documentation	95
17.8	anamodutils.c File Reference	96
17.8.1	Define Documentation	98
17.8.2	Function Documentation	98

17.9 anamodutils.h File Reference	104
17.9.1 Function Documentation	105
17.10category.c File Reference	105
17.10.1 Define Documentation	107
17.10.2 Function Documentation	107
17.10.3 Variable Documentation	115
17.11component.c File Reference	116
17.11.1 Define Documentation	117
17.11.2 Function Documentation	117
17.12feature.c File Reference	119
17.12.1 Define Documentation	120
17.12.2 Function Documentation	121
17.12.3 Variable Documentation	125
17.13icmk.c File Reference	126
17.13.1 Detailed Description	127
17.13.2 Inverse Cuthill-McKee distribution	127
17.13.3 Usage	128
17.13.4 References	128
17.13.5 Define Documentation	128
17.13.6 Function Documentation	130
17.13.7 Variable Documentation	136
17.14icmk.h File Reference	136
17.14.1 Function Documentation	137
17.15iprs.c File Reference	138
17.15.1 Detailed Description	140
17.15.2 Intelligent Preconditioner Recommendation System	140
17.15.3 Usage	140
17.15.4 References	141
17.15.5 Function Documentation	141
17.16jpl.c File Reference	150
17.16.1 Detailed Description	151

17.16.2 Jones-Plassmann multi-colouring	151
17.16.3 Usage	151
17.16.4 References	152
17.16.5 Function Documentation	152
17.17lapack.c File Reference	156
17.17.1 Detailed Description	158
17.17.2 Dense calculations from Lapack	158
17.17.3 Usage	158
17.17.4 Define Documentation	158
17.17.5 Function Documentation	159
17.18logging.c File Reference	164
17.18.1 Detailed Description	165
17.18.2 Event logging	165
17.18.3 Function Documentation	165
17.19Make.inc File Reference	165
17.20module_functions.c File Reference	165
17.20.1 Detailed Description	167
17.20.2 Define Documentation	168
17.20.3 Function Documentation	168
17.20.4 Variable Documentation	177
17.21normal.c File Reference	179
17.21.1 Detailed Description	180
17.21.2 Estimates for the departure from normality	180
17.21.3 Usage	181
17.21.4 References	181
17.21.5 Define Documentation	182
17.21.6 Function Documentation	182
17.21.7 Variable Documentation	190
17.22options.c File Reference	190
17.22.1 Detailed Description	191
17.22.2 Define Documentation	192

17.22.3 Function Documentation	192
17.22.4 Variable Documentation	194
17.23petsc.c File Reference	194
17.23.1 Function Documentation	195
17.24reporting.c File Reference	197
17.24.1 Function Documentation	198
17.25simple.c File Reference	200
17.25.1 Detailed Description	201
17.25.2 Simple (normlike) quantities	201
17.25.3 Usage	202
17.25.4 Define Documentation	202
17.25.5 Function Documentation	203
17.26spectrum.c File Reference	212
17.26.1 Detailed Description	214
17.26.2 Spectral properties	214
17.26.3 Function Documentation	216
17.26.4 Variable Documentation	233
17.27stats.c File Reference	233
17.27.1 Detailed Description	234
17.27.2 Statistics on the AnaMod system	234
17.27.3 Function Documentation	234
17.28structure.c File Reference	235
17.28.1 Detailed Description	237
17.28.2 Structural properties	237
17.28.3 Usage	237
17.28.4 Function Documentation	238
17.29tracing.c File Reference	248
17.29.1 Detailed Description	249
17.29.2 Tracing the analysis modules	249
17.29.3 Function Documentation	250
17.29.4 Variable Documentation	251

17.30utils.c File Reference	252
17.30.1 Detailed Description	252
17.30.2 Function Documentation	252
17.31variance.c File Reference	252
17.31.1 Detailed Description	253
17.31.2 Measurements of element variance	253
17.31.3 Usage	254
17.31.4 Function Documentation	254

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.

[Installation and compiling programs](#)

[Use of the analysis modules](#)

[List of the available modules](#)

[Customization](#)

[Bulk computation](#)

[Tracing the analysis modules](#)

Author

Victor Eijkhout

Version

1.91

Date

unreleased

Change log:

- 2.0 : internals revamped, bug fixes in simple:diagonal-dominance
- 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; PetscBool f1,f2; PetscErrorCode ierr;

    PetscFunctionBegin;
    ierr = NewFeatureValues(&values); CHKERRQ(ierr);
```

```

ierr = InstantiateFeatureSet((void*)problem, features, values); CHKERRQ(ierr);
ierr = GetFeatureValue(values, idx, &sn, &f1); CHKERRQ(ierr);
ierr = GetFeatureValue(values, idx, &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. - These are necessary.

5.2 Category registration

Every category needs to be registered before use with a call

```
Register<Category>Modules();
```

where the available categories 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 [AnaMod-RegisterStandardModules\(\)](#) installs all standard available modules.

5.3 Quantity computation

After registering a category, its elements can be computed as

```
PetscErrorCode ComputeQuantity(char *cat,char *cmp,Mat A,AnalysisItem *res,Petsc-Bool *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 [AnaModGetTypeNames\(\)](#) or [AnaModGetTypeMySQLName\(\)](#).

6 List of the available modules

[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](#)

7 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, PetscBool *flg)
```

where

- `A` is the input matrix
- `rv` is the return value
- `flg` is true if the quantity was successfully 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 individual calls to [RegisterModule\(\)](#). See [types](#) and [Array type handling](#).

8 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.

9 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 `PetscObjectComposedDataGetInt()`. 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()`.

10 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 systematically the available modules. This can be done with `GetCategories()` and `CategoryGetModules()`.

Related: `HasComputeCategory()`, `HasComputeModule()`.

Auxiliary routines: `GetCategoryIndex()`, `GetModuleIndex()`.

11 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` : Certain operations can be very expensive, for instance because they need to be performed on a single processor. AnaMod will by default refuse to compute these quantities, but their computation can be forced by `"-anamod_force <option>"` where `option` is
 - `"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.
- 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.
- `"-anamod_use_only <mod>"` : prevent other categories from being computed
- `-anamod_compute` : (not implemented yet)

12 Module Index

12.1 Modules

Here is a list of all modules:

Files with defined categories	11
User-accessible functions	13

Functions only of use to the implementation

14

13 Data Structure Index

13.1 Data Structures

Here are the data structures with brief descriptions:

AnalysisDataTypeArray_	15
AnalysisItem	16
AnalysisItemArray_	18
categoryobject_	20
componentobject_	22
FeatureSet_	24
FeatureValues_	26
IntArray_	27
StringArray_	28

14 File Index

14.1 File List

Here is a list of all files with brief descriptions:

anamatrix.c	30
anamatrix.h	31
anamod.h	
Prototypes for general module functions	33
anamoddriver.c	73
anamodsalsa.c	76

anamodsalsamodules.h	
Prototypes for using the standard modules	82
anamodtypes.h	93
anamodutils.c	96
anamodutils.h	104
category.c	105
component.c	116
feature.c	119
icmk.c	
Functions for computing the ICMK structure of a matrix	126
icmk.h	136
iprs.c	
Quantities used in the UKY 'Intelligent Preconditioner Recommendation System'	138
jpl.c	
Functions for computing the JPL multicolour structure of a matrix	150
lapack.c	
Dense routines from Lapack: eigenvalue and schur stuff	156
logging.c	
PETSc event logging in AnaMod	164
Make.inc	165
module_functions.c	
User functions for accessing the analysis modules	165
normal.c	
Various estimates of the departure from normality	179
options.c	
Commandline options handling	190
petsc.c	194
reporting.c	197

simple.c	
Norm-like properties	200
spectrum.c	
Various estimates of spectrum related quantities	212
stats.c	
Statistics on the AnaMod system	233
structure.c	
Structural properties of a matrix	235
tracing.c	
Trace routines for the anamod package	248
utils.c	252
variance.c	
Various estimates of how 'wild' a matrix is	252

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.

15.2 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.

15.3 Functions only of use to the implementation

Files

- file [anamotypes.h](#)
- file [options.c](#)
Commandline options handling.
- file [utils.c](#)

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 331 of file anamodutils.c.

16.1.2 Field Documentation

16.1.2.1 int AnalysisDataTypeArray_::alloc

Definition at line 332 of file anamodutils.c.

Referenced by [AnalysisDataTypeArrayAdd\(\)](#), [AnalysisDataTypeArrayGetAt\(\)](#), [AnalysisDataTypeArraySetAt\(\)](#), [AnalysisDataTypeArrayTryGetAt\(\)](#), and [CreateAnalysisDataTypeArray\(\)](#).

16.1.2.2 AnalysisDataType* AnalysisDataTypeArray_::data

Definition at line 332 of file anamodutils.c.

Referenced by [AnalysisDataTypeArrayAdd\(\)](#), [AnalysisDataTypeArrayGetAt\(\)](#), [AnalysisDataTypeArraySetAt\(\)](#), [AnalysisDataTypeArrayTryGetAt\(\)](#), [CreateAnalysisDataTypeArray\(\)](#), and [DeleteAnalysisDataTypeArray\(\)](#).

16.1.2.3 int AnalysisDataTypeArray_::fill

Definition at line 332 of file anamodutils.c.

Referenced by [AnalysisDataTypeArrayAdd\(\)](#), [AnalysisDataTypeArraySetAt\(\)](#), and [CreateAnalysisDataTypeArray\(\)](#).

16.1.2.4 int * AnalysisDataTypeArray_::has

Definition at line 332 of file anamodutils.c.

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

16.1.2.5 char* AnalysisDataTypeArray_::name

Definition at line 332 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](#)
- const char * [c](#)

16.2.1 Detailed Description

Definition at line 11 of file anamodtypes.h.

16.2.2 Field Documentation

16.2.2.1 const 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 AvgNnzpRow(), BlockSize(), ComponentRetrieve(), 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(), ComponentRetrieve(), - 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(), ComponentRetrieve(), 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.

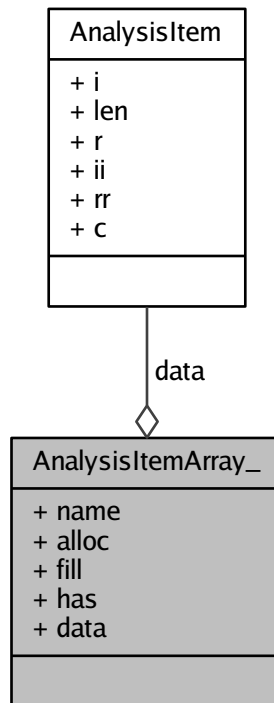
Referenced by ComponentRetrieve(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyRealIm(), MaxEVbyRealRe(), MinEVbyMagIm(), MinEVbyMagRe(), QuantityAsString(), RetrieveQuantityByID(), RitzValuesC(), and - RitzValuesR().

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.

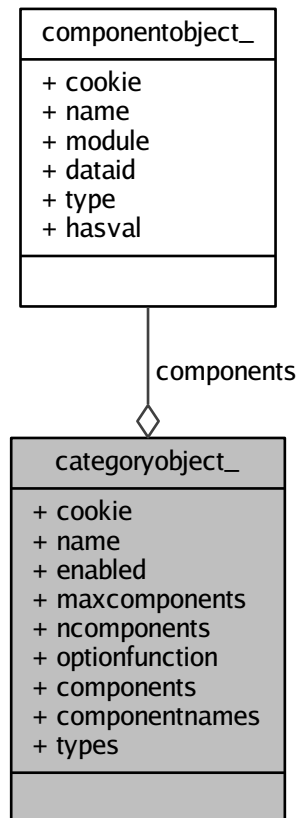
Referenced by CreateAnalysisItemArray(), and DeleteAnalysisItemArray().

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

- [anamodutils.c](#)

16.4 categoryobject_ Struct Reference

Collaboration diagram for categoryobject_:



Data Fields

- int `cookie`
- char * `name`
- int `enabled`
- int `maxcomponents`

- int [ncomponents](#)
- PetscErrorCode(* [optionfunction](#))(char *)
- [componentobject](#) * [components](#)
- const char ** [componentnames](#)
- [AnalysisDataType](#) * [types](#)

16.4.1 Detailed Description

Definition at line 9 of file category.c.

16.4.2 Field Documentation

16.4.2.1 const char** categoryobject_::componentnames

Definition at line 15 of file category.c.

Referenced by [CategoryGetComponentIndex\(\)](#), [CategoryGetModules\(\)](#), [CategoryGetOrCreateComponent\(\)](#), [CreateCategoryObject\(\)](#), and [DestroyCategoryObject\(\)](#).

16.4.2.2 componentobject* categoryobject_::components

Definition at line 15 of file category.c.

Referenced by [CategoryGetComponent\(\)](#), [CategoryGetOrCreateComponent\(\)](#), [CreateCategoryObject\(\)](#), and [DestroyCategoryObject\(\)](#).

16.4.2.3 int categoryobject_::cookie

Definition at line 10 of file category.c.

Referenced by [CreateCategoryObject\(\)](#).

16.4.2.4 int categoryobject_::enabled

Definition at line 12 of file category.c.

Referenced by [CategoryEnableByName\(\)](#), and [CreateCategoryObject\(\)](#).

16.4.2.5 int categoryobject_::maxcomponents

Definition at line 13 of file category.c.

Referenced by [CategoryGetOrCreateComponent\(\)](#), and [CreateCategoryObject\(\)](#).

16.4.2.6 char* categoryobject_::name

Definition at line 11 of file category.c.

Referenced by `CreateCategoryObject()`, `DestroyCategoryObject()`, `GetFirstCategory()`, `GetNextCategory()`, and `GetOrCreateCategory()`.

16.4.2.7 `int categoryobject_::ncomponents`

Definition at line 13 of file `category.c`.

Referenced by `CategoryGetComponentIndex()`, `CategoryGetModules()`, `CategoryGetOrCreateComponent()`, `CreateCategoryObject()`, and `DestroyCategoryObject()`.

16.4.2.8 `PetscErrorCode(* categoryobject_::optionfunction)(char *)`

Definition at line 14 of file `category.c`.

Referenced by `CategoryGetOptionFunction()`, and `DeclareCategoryOptionFunction()`.

16.4.2.9 `AnalysisDataType* categoryobject_::types`

Definition at line 16 of file `category.c`.

Referenced by `CategoryComponentSetModule()`, `CategoryGetModules()`, `CreateCategoryObject()`, and `DestroyCategoryObject()`.

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

- [category.c](#)

16.5 componentobject_ Struct Reference

Data Fields

- `int` [cookie](#)
- `char *` [name](#)
- `PetscErrorCode(* module)(AnaModNumericalProblem, AnalysisItem *, int *, - PetscBool *)`
- `int` [dataid](#)
- [AnalysisDataType](#) `type`
- `PetscBool` [hasval](#)

16.5.1 Detailed Description

Definition at line 8 of file `component.c`.

16.5.2 Field Documentation

16.5.2.1 int componentobject_::cookie

Definition at line 9 of file component.c.

Referenced by CreateComponentObject().

16.5.2.2 int componentobject_::dataid

Definition at line 12 of file component.c.

Referenced by ComponentGetId(), ComponentRetrieve(), and ComponentSetModule().

16.5.2.3 PetscBool componentobject_::hasval

Definition at line 12 of file component.c.

Referenced by ComponentCompute().

16.5.2.4 PetscErrorCode(* componentobject_::module)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *)

Definition at line 11 of file component.c.

Referenced by ComponentCompute(), and ComponentSetModule().

16.5.2.5 char* componentobject_::name

Definition at line 10 of file component.c.

Referenced by ComponentGetName(), CreateComponentObject(), and DestroyComponentObject().

16.5.2.6 AnalysisDataType componentobject_::type

Definition at line 12 of file component.c.

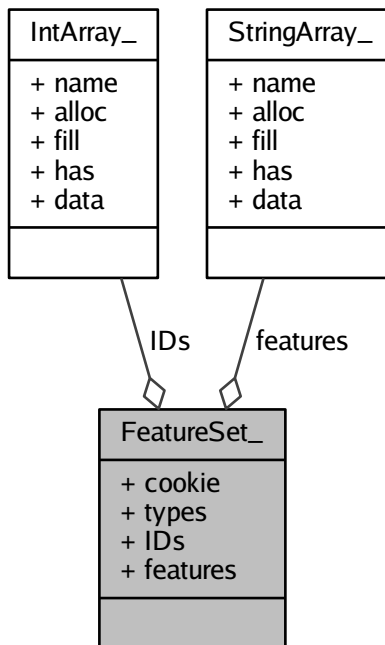
Referenced by ComponentGetType(), ComponentRetrieve(), and ComponentSetModule().

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

- [component.c](#)

16.6 FeatureSet_ Struct Reference

Collaboration diagram for FeatureSet_:



Data Fields

- int [cookie](#)
- [AnalysisDataType](#) * [types](#)
- [IntArray](#) [IDs](#)
- [StringArray](#) [features](#)

16.6.1 Detailed Description

Definition at line 60 of file `feature.c`.

16.6.2 Field Documentation

16.6.2.1 int FeatureSet_::cookie

Definition at line 61 of file feature.c.

Referenced by NewFeatureSet().

16.6.2.2 StringArray FeatureSet_::features

Definition at line 62 of file feature.c.

Referenced by NewFeatureSet().

16.6.2.3 IntArray FeatureSet_::IDs

Definition at line 62 of file feature.c.

Referenced by NewFeatureSet().

16.6.2.4 AnalysisDataType* FeatureSet_::types

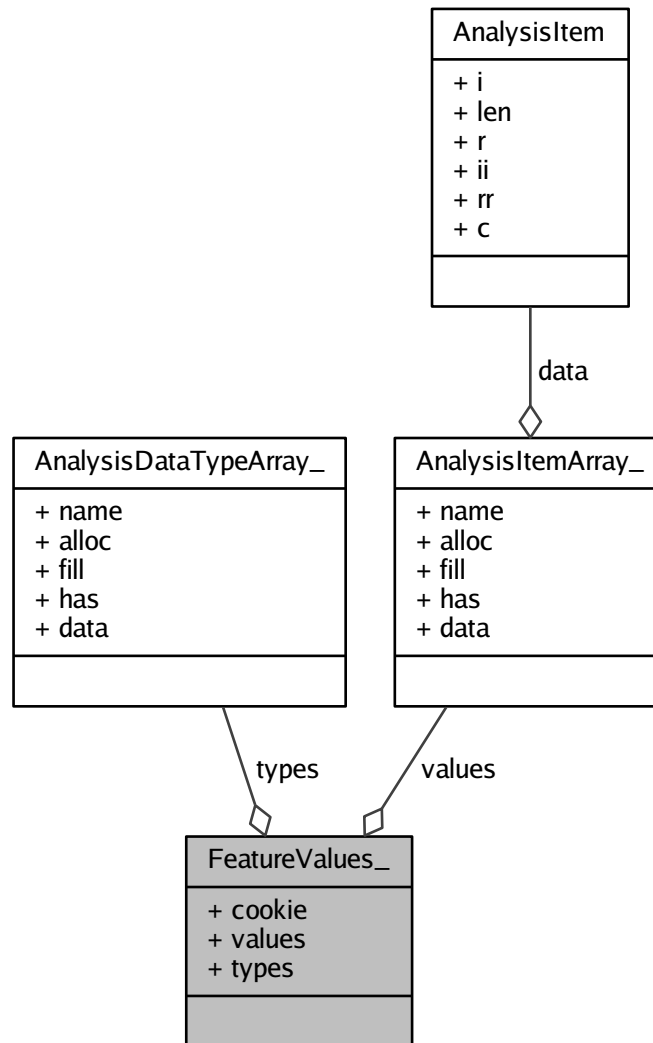
Definition at line 62 of file feature.c.

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

- [feature.c](#)

16.7 FeatureValues_ Struct Reference

Collaboration diagram for FeatureValues_:



Data Fields

- int [cookie](#)
- [AnalysisItemArray](#) values
- [AnalysisDataTypeArray](#) types

16.7.1 Detailed Description

Definition at line 66 of file feature.c.

16.7.2 Field Documentation

16.7.2.1 int FeatureValues_::cookie

Definition at line 67 of file feature.c.

Referenced by NewFeatureValues().

16.7.2.2 AnalysisDataTypeArray FeatureValues_::types

Definition at line 68 of file feature.c.

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

16.7.2.3 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](#)

16.8 IntArray_ Struct Reference

Data Fields

- char * [name](#)
- int [alloc](#)
- int [fill](#)
- PetscBool * [has](#)
- int * [data](#)

16.8.1 Detailed Description

Definition at line 7 of file anamodutils.c.

16.8.2 Field Documentation

16.8.2.1 int IntArray_::alloc

Definition at line 8 of file anamodutils.c.

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

16.8.2.2 int* IntArray_::data

Definition at line 8 of file anamodutils.c.

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

16.8.2.3 int IntArray_::fill

Definition at line 8 of file anamodutils.c.

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

16.8.2.4 PetscBool* IntArray_::has

Definition at line 8 of file anamodutils.c.

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

16.8.2.5 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](#)

16.9 StringArray_ Struct Reference

Data Fields

- char * [name](#)

- int [alloc](#)
- int [fill](#)
- PetscBool * [has](#)
- char ** [data](#)

16.9.1 Detailed Description

Definition at line 111 of file `anamodutils.c`.

16.9.2 Field Documentation

16.9.2.1 int StringArray_::alloc

Definition at line 112 of file `anamodutils.c`.

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

16.9.2.2 char** StringArray_::data

Definition at line 112 of file `anamodutils.c`.

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

16.9.2.3 int StringArray_::fill

Definition at line 112 of file `anamodutils.c`.

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

16.9.2.4 PetscBool* StringArray_::has

Definition at line 112 of file `anamodutils.c`.

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

16.9.2.5 char* StringArray_::name

Definition at line 112 of file `anamodutils.c`.

Referenced by `CreateStringArray()`, and `DeleteStringArray()`.

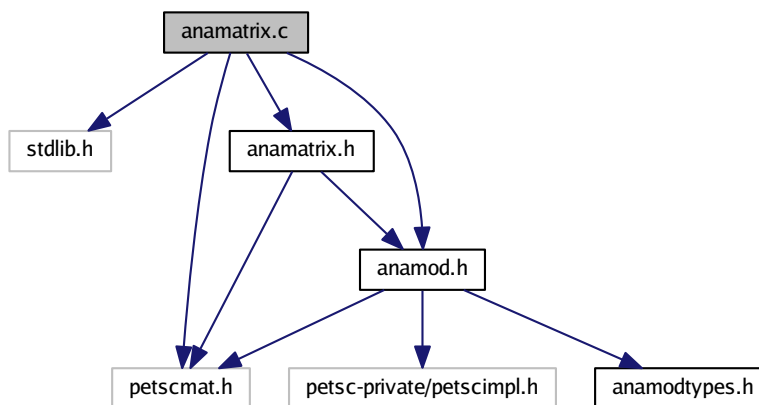
The documentation for this struct was generated from the following file:

- [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, PetscBool *success)

17.1.1 Function Documentation

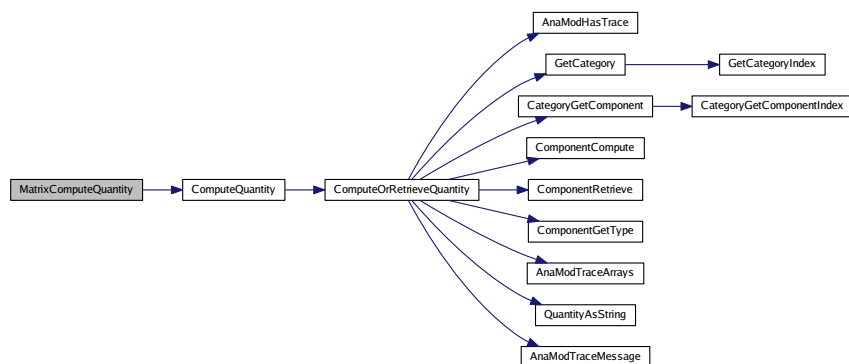
17.1.1.1 PetscErrorCode MatrixComputeQuantity (Mat A, const char * cat, const char * cmp, AnalysisItem * res, int * l, PetscBool * success)

Definition at line 15 of file anamatrix.c.

References [ComputeQuantity\(\)](#).

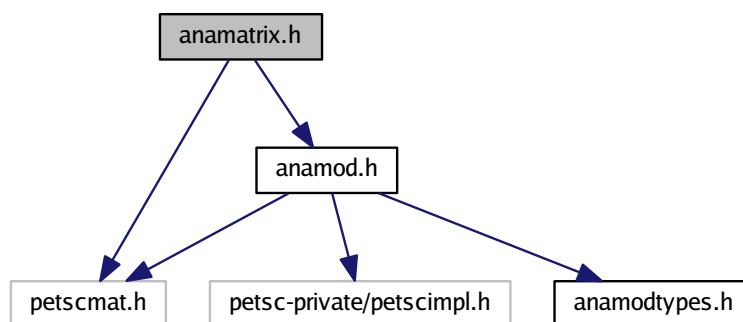
Referenced by [computennz\(\)](#), [JonesPlassmannColouring\(\)](#), [Lee95bounds\(\)](#), [MatCenter\(\)](#), and [MatCommutatorNormF_seq\(\)](#).

Here is the call graph for this function:

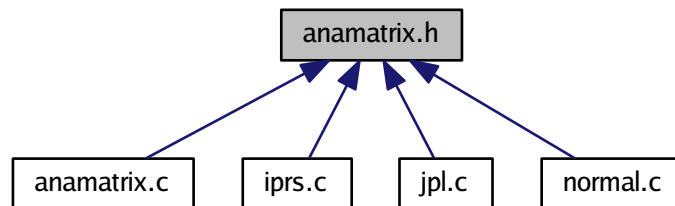


17.2 anamatrix.h File Reference

`#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 *I, PetscBool *success)

17.2.1 Function Documentation

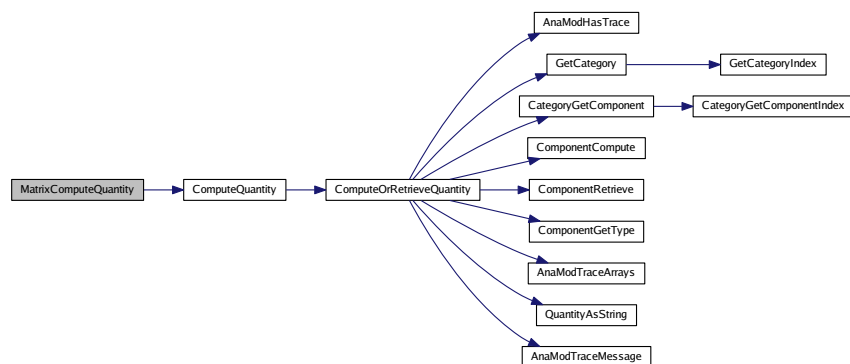
17.2.1.1 PetscErrorCode **MatrixComputeQuantity** (Mat , const char * , const char * , **AnalysisItem** * res, int * I, PetscBool * success)

Definition at line 15 of file anamatrix.c.

References [ComputeQuantity\(\)](#).

Referenced by [computennz\(\)](#), [JonesPlassmannColouring\(\)](#), [Lee95bounds\(\)](#), [MatCenter\(\)](#), and [MatCommutatorNormF_seq\(\)](#).

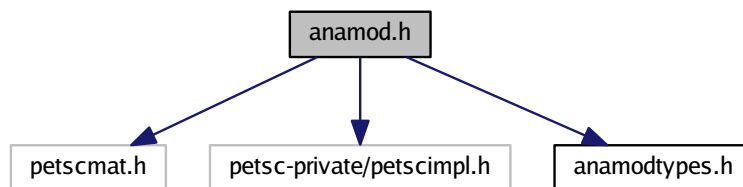
Here is the call graph for this function:



17.3 anamod.h File Reference

Prototypes for general module functions.

```
#include "petscmat.h" #include <petsc-private/petscsimpl.h>
#include "anamodtypes.h" Include dependency graph for anamod.h:
```



The diagram illustrates the hierarchical structure of the dataset. At the top is the root node 'dataset'. It branches down into ten intermediate nodes labeled 'dataset_1' through 'dataset_10'. Each of these intermediate nodes then branches into a common set of feature nodes. The features include 'intermediate_1', 'intermediate_2', 'category', 'component', 'attribute_1', 'point', 'feature', 'value', 'link', 'tagging', 'module_function', 'option', 'reporting', 'output', 'structure', 'testing', and 'status'. The connections are shown as lines from the parent nodes to the child nodes, indicating a hierarchical relationship.

- #define ANAMOD_FORMAT_VERSION "1.0"
- #define MXC 30
- #define INC 7
- #define TRUTH(x) ((x) ? PETSC_TRUE : PETSC_FALSE)
- #define HASTOEXIST(h)
- #define ANAMODCHECKVALID(i, c, s) {if (!(i)) SETERRQ(MPI_COMM_WORLD,1,"Null pointer"); if ((i->cookie!=c) SETERRQ1(MPI_COMM_WORLD,1,"Not a valid <%s>",s);}

- typedef struct **categoryobject_** * **categoryobject**
- typedef struct **componentobject_** * **componentobject**
- typedef struct **FeatureSet_** * **FeatureSet**
- typedef struct **FeatureValues** * **FeatureValues**

- enum `CatCmpEnableMode` { `CATCMP_ENABLE`, `CATCMP_SKIP_FROM_LOOPS`, `CATCMP_DISABLE` }

- PetscErrorCode [AnaModInitialize](#) ()
- PetscErrorCode [AnaModFinalize](#) ()
- PetscErrorCode [AnaModGetTypeNames](#) (int [id](#), const char **[name](#))
- PetscErrorCode [AnaModGetTypeMySQLName](#) (int [id](#), const char **[name](#))
- PetscErrorCode [AllocCategoryObjects](#) ()
- PetscErrorCode [FreeCategoryObjects](#) ()
- PetscErrorCode [CreateCategoryObject](#) (const char *, [categoryobject](#) *)
- PetscErrorCode [DestroyCategoryObject](#) ([categoryobject](#))
- PetscErrorCode [GetCategory](#) (const char *, [categoryobject](#) *, PetscBool *)

- PetscErrorCode [GetOrCreateCategory](#) (const char *, [categoryobject](#) *)
- PetscErrorCode [CategoryGetComponentIndex](#) ([categoryobject](#), const char *, int *, PetscBool *)
- PetscErrorCode [CategoryComponentSetModule](#) (const char *, const char *, - [AnalysisDataType](#), int, PetscErrorCode (*)([AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *))
- PetscErrorCode [AnaModRegisterStandardModules](#) ()
- PetscErrorCode [RegisterModule](#) (const char *, const char *, [AnalysisDataType](#), PetscErrorCode (*)([AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *))
- PetscErrorCode [DeRegisterCategory](#) (const char *cat)
- PetscErrorCode [DeregisterModules](#) (void)
- PetscErrorCode [GetCategories](#) (int *, const char ***)
- PetscErrorCode [CategoryGetModules](#) (const char *, const char ***, [AnalysisDataType](#) **, int **, int *)
- PetscErrorCode [HasComputeCategory](#) (const char *, PetscBool *)
- PetscErrorCode [HasComputeModule](#) (const char *, const char *, PetscBool *)
- PetscErrorCode [CategoryEnableByName](#) (const char *, int)
- PetscErrorCode [GetFirstCategory](#) (const char **, PetscBool *)
- PetscErrorCode [GetNextCategory](#) (const char **, PetscBool *)
- PetscErrorCode [CreateComponentObject](#) (const char *, [componentobject](#) *)
- PetscErrorCode [DestroyComponentObject](#) ([componentobject](#))
- PetscErrorCode [CategoryGetOrCreateComponent](#) ([categoryobject](#), const char *, [componentobject](#) *)
- PetscErrorCode [CategoryGetComponent](#) ([categoryobject](#), const char *, [componentobject](#) *, PetscBool *)
- PetscErrorCode [ComponentSetModule](#) ([componentobject](#), [AnalysisDataType](#), int, PetscErrorCode (*)([AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *))
- PetscErrorCode [ComponentGetType](#) ([componentobject](#), [AnalysisDataType](#) *)
- PetscErrorCode [ComponentCompute](#) ([componentobject](#), [AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *)
- PetscErrorCode [ComponentRetrieve](#) ([componentobject](#), [AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *)
- PetscErrorCode [ComponentGetName](#) ([componentobject](#), const char **)
- PetscErrorCode [ComponentGetId](#) ([componentobject](#), int *)
- PetscErrorCode [GetDataID](#) (const char *, const char *, int *, PetscBool *)
- PetscErrorCode [GetDataType](#) (const char *, const char *, [AnalysisDataType](#) *, PetscBool *)
- PetscErrorCode [ComputeQuantity](#) ([AnaModNumericalProblem](#), const char *, const char *, [AnalysisItem](#) *, int *, PetscBool *)
- PetscErrorCode [ComputeQuantityByID](#) ([AnaModNumericalProblem](#), int, int, - [AnalysisItem](#) *, int *, PetscBool *)

- PetscErrorCode [HasQuantity](#) ([AnaModNumericalProblem](#), const char *, const char *, PetscBool *)
- PetscErrorCode [HasQuantityByID](#) ([AnaModNumericalProblem](#), int, [AnalysisDataType](#), PetscBool *)
- PetscErrorCode [RetrieveQuantity](#) ([AnaModNumericalProblem](#), const char *, const char *, [AnalysisItem](#) *, int *, PetscBool *)
- PetscErrorCode [RetrieveQuantityByID](#) ([AnaModNumericalProblem](#), int, [AnalysisDataType](#), [AnalysisItem](#) *, PetscBool *)
- PetscErrorCode [QuantityAsString](#) ([AnalysisItem](#) *, [AnalysisDataType](#), char **)
- PetscErrorCode [AnaModOptionsHandling](#) (void)
- PetscErrorCode [AnaModShowOptions](#) (MPI_Comm)
- PetscErrorCode [DeclareCategoryOptionFunction](#) (const char *cat, PetscErrorCode(*)(char *))
- PetscErrorCode [CategoryGetOptionFunction](#) (const char *cat, PetscErrorCode(**f)(char *))
- PetscErrorCode [AnaModHasForcedSequentialComputation](#) (PetscBool *)
- PetscErrorCode [AnaModHasForcedExpensiveComputation](#) (PetscBool *)
- PetscErrorCode [AnaModGetSequentialMatrix](#) (Mat A, Mat *Awork, PetscBool *mem, PetscBool *local, PetscBool *global)
- PetscErrorCode [AnaModDeclareTraceFunction](#) (PetscErrorCode*)(void *, const char *, va_list)
- PetscErrorCode [AnaModDeclareTraceContext](#) (void *)
- PetscErrorCode [AnaModTraceMessage](#) (const char *fmt,...)
- PetscErrorCode [AnaModHasTrace](#) (PetscBool *flag)
- PetscErrorCode [AnaModSetTraceArrays](#) (PetscBool f)
- PetscErrorCode [AnaModTraceArrays](#) (PetscBool *f)
- PetscErrorCode [TabReportModules](#) (Mat, char **, char **, int)
- PetscErrorCode [PurgeAttachedArrays](#) (Mat A)
- PetscErrorCode [NewFeatureSet](#) ([FeatureSet](#) *)
- PetscErrorCode [DeleteFeatureSet](#) ([FeatureSet](#))
- PetscErrorCode [AddToFeatureSet](#) ([FeatureSet](#), const char *, const char *, int *)
- PetscErrorCode [NewFeatureValues](#) ([FeatureValues](#) *)
- PetscErrorCode [DeleteFeatureValues](#) ([FeatureValues](#))
- PetscErrorCode [InstantiateFeatureSet](#) ([AnaModNumericalProblem](#), [FeatureSet](#), [FeatureValues](#))
- PetscErrorCode [GetFeatureValue](#) ([FeatureValues](#), int, [AnalysisItem](#) *, PetscBool *)
- PetscErrorCode [AnaModSetRetrievalFunction](#) (PetscErrorCode*)(void *, char *, char *, [AnalysisItem](#) *, [AnalysisDataType](#) *, PetscBool *)
- PetscErrorCode [AnaModGetRetrievalFunction](#) (PetscErrorCode(**f)(void *, char *, char *, [AnalysisItem](#) *, [AnalysisDataType](#) *, PetscBool *), PetscBool *)
- PetscErrorCode [AnaModCheckValidFeatureSet](#) (void *)
- PetscErrorCode [CreateIntArray](#) (const char *, int, [IntArray](#) *)

- PetscErrorCode [DeleteIntArray](#) ([IntArray](#))
- PetscErrorCode [IntArrayAdd](#) ([IntArray](#), int, int *)
- PetscErrorCode [IntArraySetAt](#) ([IntArray](#), int, int)
- PetscErrorCode [IntArrayTryGetAt](#) ([IntArray](#), int, int *, PetscBool *)
- PetscErrorCode [IntArrayGetAt](#) ([IntArray](#), int, int *)
- PetscErrorCode [IntArrayGetFill](#) ([IntArray](#), int *)
- PetscErrorCode [CreateStringArray](#) (const char *, int, [StringArray](#) *)
- PetscErrorCode [DeleteStringArray](#) ([StringArray](#))
- PetscErrorCode [StringArrayAdd](#) ([StringArray](#), const char *, int *)
- PetscErrorCode [StringArraySetAt](#) ([StringArray](#), int, const char *)
- PetscErrorCode [StringArrayTryGetAt](#) ([StringArray](#), int, char **, PetscBool *)
- PetscErrorCode [StringArrayGetAt](#) ([StringArray](#), int, char **)
- PetscErrorCode [StringArrayGetFill](#) ([StringArray](#), int *)
- PetscErrorCode [CreateAnalysisItemArray](#) (const char *, int, [AnalysisItemArray](#) *)
- PetscErrorCode [DeleteAnalysisItemArray](#) ([AnalysisItemArray](#))
- PetscErrorCode [AnalysisItemArrayAdd](#) ([AnalysisItemArray](#), [AnalysisItem](#), int *)
- PetscErrorCode [AnalysisItemArraySetAt](#) ([AnalysisItemArray](#), int, [AnalysisItem](#))
- PetscErrorCode [AnalysisItemArrayTryGetAt](#) ([AnalysisItemArray](#), int, [AnalysisItem](#) *, PetscBool *)
- PetscErrorCode [AnalysisItemArrayGetAt](#) ([AnalysisItemArray](#), int, [AnalysisItem](#) *)
- PetscErrorCode [CreateAnalysisDataTypeArray](#) (const char *, int, [AnalysisDataTypeArray](#) *)
- PetscErrorCode [DeleteAnalysisDataTypeArray](#) ([AnalysisDataTypeArray](#))
- PetscErrorCode [AnalysisDataTypeArrayAdd](#) ([AnalysisDataTypeArray](#), [AnalysisDataType](#), int *)
- PetscErrorCode [AnalysisDataTypeArraySetAt](#) ([AnalysisDataTypeArray](#), int, - [AnalysisDataType](#))
- PetscErrorCode [AnalysisDataTypeArrayTryGetAt](#) ([AnalysisDataTypeArray](#), int, - [AnalysisDataType](#) *, PetscBool *)
- 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(MPI_COMM_WORLD,1,"Null pointer"); if ((i)->cookie!=c)
      SETERRQ1(MPI_COMM_WORLD,1,"Not a valid <%s>",s);}
```

Definition at line 24 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 19 of file anamod.h.

Referenced by AvgDiagDist(), 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(), MaxEVbyReallm(), 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 INC 7`

Definition at line 12 of file anamod.h.

17.3.2.5 `#define MXC 30`

Definition at line 11 of file anamod.h.

Referenced by `AllocCategoryObjects()`, and `CreateCategoryObject()`.

17.3.2.6 `#define TRUTH(x) ((x) ? PETSC_TRUE : PETSC_FALSE)`

Definition at line 18 of file anamod.h.

Referenced by `GetUpBiDiagSplits()`.

17.3.3 Typedef Documentation

17.3.3.1 `typedef struct categoryobject_* categoryobject`

Definition at line 26 of file anamod.h.

17.3.3.2 `typedef struct componentobject_* componentobject`

Definition at line 27 of file anamod.h.

17.3.3.3 `typedef struct FeatureSet_* FeatureSet`

Definition at line 144 of file anamod.h.

17.3.3.4 `typedef struct FeatureValues_* FeatureValues`

Definition at line 145 of file anamod.h.

17.3.4 Enumeration Type Documentation

17.3.4.1 `enum CatCmpEnableMode`

Enumerator:

CATCMP_ENABLE
CATCMP_SKIP_FROM_LOOPS
CATCMP_DISABLE

Definition at line 63 of file anamod.h.

17.3.5 Function Documentation

17.3.5.1 PetscErrorCode AddToFeatureSet (FeatureSet set, const char * cat, const 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 115 of file `feature.c`.

References `CHECKVALIDFSET`, `name`, and `StringArrayAdd()`.

Here is the call graph for this function:



17.3.5.2 PetscErrorCode AllocCategoryObjects ()

Definition at line 26 of file `category.c`.

References `categorynames`, `maxcategories`, `MXC`, and `ncategories`.

Referenced by `AnaModInitialize()`.

17.3.5.3 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 372 of file `anamodutils.c`.

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

17.3.5.4 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 424 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), and [AnalysisDataTypeArray_::has](#).

17.3.5.5 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 388 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), [AnalysisDataTypeArray_::fill](#), and [AnalysisDataTypeArray_::has](#).

Referenced by [InstantiateFeatureSet\(\)](#).

17.3.5.6 PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray array, int idx, AnalysisDataType * val, PetscBool * 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 410 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), and [AnalysisDataTypeArray_::has](#).

17.3.5.7 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 267 of file anamodutils.c.

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

17.3.5.8 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 319 of file anamodutils.c.

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

17.3.5.9 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 283 of file anamodutils.c.

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

Referenced by `InstantiateFeatureSet()`.

17.3.5.10 PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray array, int idx, AnalysisItem * val, PetscBool * 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 305 of file anamodutils.c.

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

Referenced by `GetFeatureValue()`.

17.3.5.11 PetscErrorCode AnaModCheckValidFeatureSet (void *)

Definition at line 242 of file feature.c.

References CHECKVALIDDFSET.

17.3.5.12 PetscErrorCode AnaModDeclareTraceContext (void *)

Definition at line 69 of file tracing.c.

References anamodtracectx.

17.3.5.13 PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(*)(void *, const 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.

17.3.5.14 PetscErrorCode AnaModFinalize ()

Finalization for AnaMod. See also [AnaModInitialize\(\)](#)

Definition at line 318 of file module_functions.c.

References FreeCategoryObjects().

Referenced by main().

Here is the call graph for this function:



17.3.5.15 `PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(**)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscBool *), PetscBool *`
`)`

Definition at line 230 of file feature.c.

References retriever.

Referenced by InstantiateFeatureSet().

17.3.5.16 `PetscErrorCode AnaModGetSequentialMatrix (Mat A, Mat * Awork, PetscBool * mem, PetscBool * local, PetscBool * 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 192 of file options.c.

References AnaModHasForcedSequentialComputation().

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

Here is the call graph for this function:



17.3.5.17 **PetscErrorCode AnaModGetTypeMySQLName** (int *id*, const char ** *name*)

Definition at line 344 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, mysqlname, and nAnaModTypeNames.

Referenced by main().

17.3.5.18 **PetscErrorCode AnaModGetTypeNames** (int *id*, const char ** *name*)

Definition at line 328 of file module_functions.c.

References anamodtypenames, and nAnaModTypeNames.

17.3.5.19 **PetscErrorCode AnaModHasForcedExpensiveComputation** (PetscBool * *flag*)

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

Definition at line 238 of file options.c.

References expensive.

Referenced by MatCommutatorNormF_seq().

17.3.5.20 **PetscErrorCode AnaModHasForcedSequentialComputation** (PetscBool * *flag*)

Query whether parallel modules should be done on processor zero.

Definition at line 165 of file options.c.

References single_proc.

Referenced by AnaModGetSequentialMatrix().

17.3.5.21 PetscErrorCode AnaModHasTrace (PetscBool * *flag*)

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 [ComputeOrRetrieveQuantity\(\)](#).

17.3.5.22 PetscErrorCode AnaModInitialize ()

Initializin for AnaMod. See also [AnaModFinalize\(\)](#)

Definition at line 284 of file module_functions.c.

References [AllocCategoryObjects\(\)](#), [AnaModIsInitialized](#), [anamodtypenames](#), and [n-AnaModTypeNames](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:

**17.3.5.23 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.

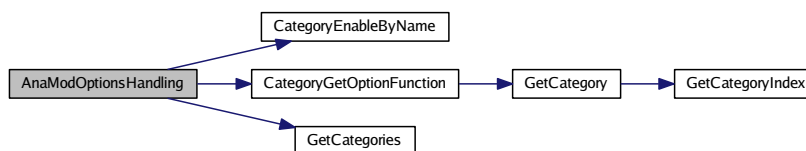
See [DeclareCategoryOptionFunction\(\)](#), [CategoryGetOptionFunction\(\)](#), [AnaModOptionsHandling\(\)](#) and section [Commandline options](#).

Definition at line 60 of file options.c.

References [CATCMP_SKIP_FROM_LOOPS](#), [CategoryEnableByName\(\)](#), [CategoryGetOptionFunction\(\)](#), [expensive](#), [GetCategories\(\)](#), [single_proc](#), and [VALUELEN](#).

Referenced by main().

Here is the call graph for this function:



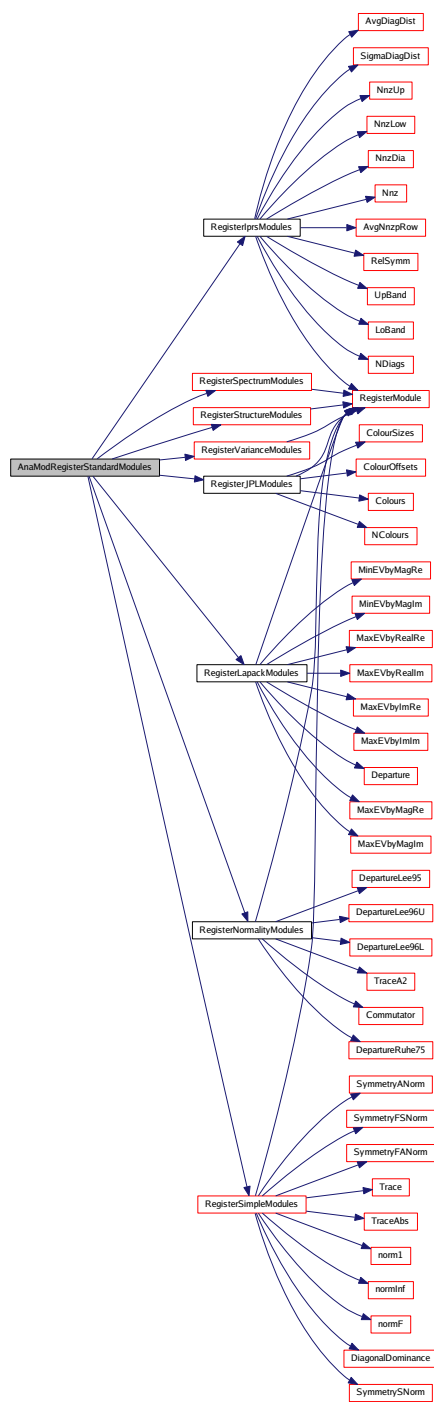
17.3.5.24 PetscErrorCode AnaModRegisterStandardModules ()

Register all standard and nonstandard analysis modules.

Definition at line 47 of file `anamodsalsa.c`.

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

Here is the call graph for this function:



17.3.5.25 `PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode*)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscBool *))`

Definition at line 220 of file feature.c.

References retriever.

17.3.5.26 `PetscErrorCode AnaModSetTraceArrays (PetscBool f)`

Definition at line 82 of file tracing.c.

References anamodtracearrays.

17.3.5.27 `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](#).

Definition at line 136 of file options.c.

References GetCategories().

Here is the call graph for this function:



17.3.5.28 `PetscErrorCode AnaModTraceArrays (PetscBool * f)`

Definition at line 95 of file tracing.c.

References anamodtracearrays.

Referenced by `ComputeOrRetrieveQuantity()`, and `ReportAnamodContent()`.

17.3.5.29 `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(), and ComputeOrRetrieveQuantity().

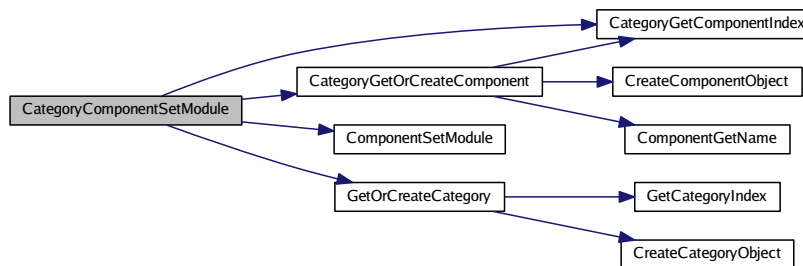
17.3.5.30 PetscErrorCode CategoryComponentSetModule (const char * , const char * , AnalysisDataType , int , PetscErrorCode*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *))

Definition at line 226 of file category.c.

References CategoryGetComponentIndex(), CategoryGetOrCreateComponent(), ComponentSetModule(), GetOrCreateCategory(), and categoryobject_::types.

Referenced by RegisterModule().

Here is the call graph for this function:



17.3.5.31 PetscErrorCode CategoryEnableByName (const char * cat, int mode)

Mark a category as enabled/disabled. Values (CatCmpEnableMode):

- 0 : enable
- 1 : skip from loops
- 2 : skip altogether

Definition at line 317 of file category.c.

References categoryobject_::enabled, name, and ncategories.

Referenced by AnaModOptionsHandling().

17.3.5.32 `PetscErrorCode CategoryGetComponent (categoryobject catg, const char * cmp, componentobject * cmpt, PetscBool * success)`

Retrieve a component.

See also [CategoryGetOrCreateComponent\(\)](#), [CategoryGetComponentIndex\(\)](#).

Definition at line 209 of file category.c.

References [CategoryGetComponentIndex\(\)](#), [CHECKVALIDCATEGORY](#), and [categoryobject_::components](#).

Referenced by [ComputeOrRetrieveQuantity\(\)](#), [GetDataID\(\)](#), [GetDataType\(\)](#), [HasComputeModule\(\)](#), and [HasQuantity\(\)](#).

Here is the call graph for this function:



17.3.5.33 `PetscErrorCode CategoryGetComponentIndex (categoryobject catg, const char * cmp, int * icmp, PetscBool * flag)`

Test for presence of a component in a category, and return its index if present. The index parameter can be null.

Definition at line 164 of file category.c.

References [CHECKVALIDCATEGORY](#), [categoryobject_::componentnames](#), and [categoryobject_::ncomponents](#).

Referenced by [CategoryComponentSetModule\(\)](#), [CategoryGetComponent\(\)](#), and [CategoryGetOrCreateComponent\(\)](#).

17.3.5.34 `PetscErrorCode CategoryGetModules (const char * cat, const 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

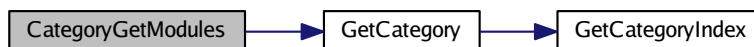
See also [GetCategories\(\)](#) and [HasComputeCategory\(\)](#).

Definition at line 256 of file `category.c`.

References `categoryobject_::componentnames`, `GetCategory()`, `categoryobject_::ncomponents`, and `categoryobject_::types`.

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

Here is the call graph for this function:



17.3.5.35 `PetscErrorCode CategoryGetOptionFunction (const char * cat, PetscErrorCode (*)(char *) f)`

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

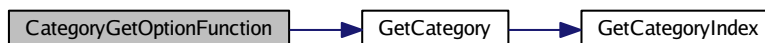
See [DeclareCategoryOptionFunction\(\)](#), `GetCategoryOptionFunction()`, [AnaModOptionsHandling\(\)](#) and section `optionsfile`.

Definition at line 358 of file `category.c`.

References `GetCategory()`, and `categoryobject_::optionfunction`.

Referenced by `AnaModOptionsHandling()`.

Here is the call graph for this function:



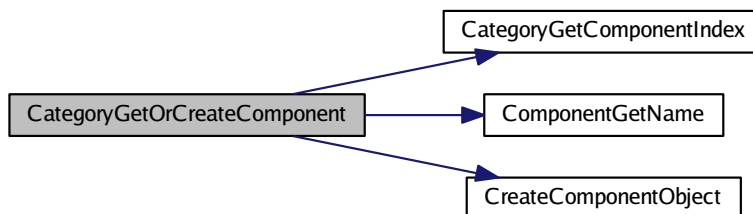
17.3.5.36 `PetscErrorCode CategoryGetOrCreateComponent (categoryobject , const char * , componentobject *)`

Definition at line 184 of file category.c.

References `CategoryGetComponentIndex()`, `CHECKVALIDCATEGORY`, `ComponentGetName()`, `categoryobject_::componentnames`, `categoryobject_::components`, `-CreateComponentObject()`, `categoryobject_::maxcomponents`, and `categoryobject_::ncomponents`.

Referenced by `CategoryComponentSetModule()`.

Here is the call graph for this function:



17.3.5.37 `PetscErrorCode CategoryLogEventRegister (char * cat, int icat)`

Definition at line 20 of file logging.c.

17.3.5.38 `PetscErrorCode ComponentCompute (componentobject , AnaModNumericalProblem , AnalysisItem * , int * , PetscBool *)`

Definition at line 57 of file component.c.

References CHECKVALIDCOMPONENT, componentobject_::hasval, and componentobject_::module.

Referenced by ComputeOrRetrieveQuantity().

17.3.5.39 PetscErrorCode ComponentGetId (componentobject , int *)

Definition at line 139 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::dataid.

Referenced by GetDataID(), and HasQuantity().

17.3.5.40 PetscErrorCode ComponentGetName (componentobject , const char **)

Definition at line 129 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::name.

Referenced by CategoryGetOrCreateComponent().

17.3.5.41 PetscErrorCode ComponentGetType (componentobject , AnalysisDataType *)

Definition at line 119 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::type.

Referenced by ComputeOrRetrieveQuantity(), GetDataType(), and HasQuantity().

17.3.5.42 PetscErrorCode ComponentRetrieve (componentobject , AnaModNumericalProblem , AnalysisItem * , int * , PetscBool *)

Definition at line 75 of file component.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, CHECKVALIDCOMPONENT, componentobject_::dataid, AnalysisItem::i, AnalysisItem::ii, AnalysisItem::r, AnalysisItem::rr, and componentobject_::type.

Referenced by ComputeOrRetrieveQuantity().

17.3.5.43 PetscErrorCode ComponentSetModule (componentobject , AnalysisDataType , int , PetscErrorCode (*)(AnaModNumericalProblem, AnalysisItem * , int * , PetscBool *))

Definition at line 43 of file component.c.

References CHECKVALIDCOMPONENT, componentobject_::dataid, id, componentobject_::module, and componentobject_::type.

Referenced by CategoryComponentSetModule().

17.3.5.44 **PetscErrorCode ComputeQuantity** (*AnaModNumericalProblem prob*, const char * *cat*, const char * *cmp*, AnalysisItem * *res*, int * *rreslen*, PetscBool * *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.

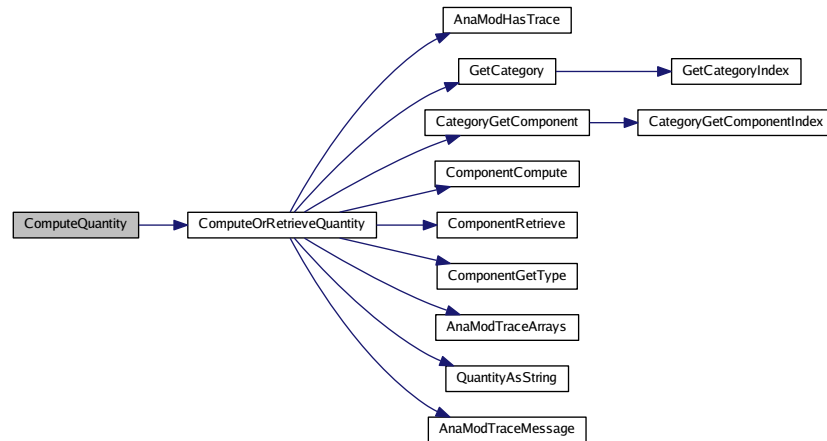
The workings of this function can be traced by specifying a trace function; see [Tracing the analysis modules](#).

Definition at line 558 of file module_functions.c.

References [AnaModIsInitialized](#), [ComputeOrRetrieveQuantity\(\)](#), and [MODE_COMPUTE](#).

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:



17.3.5.45 `PetscErrorCode ComputeQuantityByID (AnaModNumericalProblem , int , int , AnalysisItem * , int * , PetscBool *)`

17.3.5.46 `PetscErrorCode CreateAnalysisDataTypeArray (const char * , int , AnalysisDataTypeArray *)`

Definition at line 338 of file `anamodutils.c`.

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

Referenced by `NewFeatureValues()`.

17.3.5.47 `PetscErrorCode CreateAnalysisItemArray (const char * , int , AnalysisItemArray *)`

Definition at line 233 of file `anamodutils.c`.

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

Referenced by `NewFeatureValues()`.

17.3.5.48 `PetscErrorCode CreateCategoryObject (const char * cat, categoryobject * obj)`

Create a category object with a given name. See also [DestroyCategoryObject\(\)](#).

Definition at line 57 of file category.c.

References CATCMP_ENABLE, CATEGORYCOOKIE, categoryobject_::componentnames, categoryobject_::components, categoryobject_::cookie, categoryobject_::enabled, categoryobject_::maxcomponents, MXC, categoryobject_::name, categoryobject_::ncomponents, and categoryobject_::types.

Referenced by GetOrCreateCategory().

17.3.5.49 `PetscErrorCode CreateComponentObject (const char *, componentobject *)`

Definition at line 17 of file component.c.

References COMPONENTCOOKIE, componentobject_::cookie, and componentobject_::name.

Referenced by CategoryGetOrCreateComponent().

17.3.5.50 `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().

17.3.5.51 `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().

17.3.5.52 `PetscErrorCode DeclareCategoryOptionFunction (const 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\(\)](#), [AnaMod-OptionsHandling\(\)](#) and section optionsfile.

Definition at line 338 of file category.c.

References `GetCategory()`, and `categoryobject_::optionfunction`.

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



17.3.5.53 `PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray)`

Definition at line 356 of file `anamodutils.c`.

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

Referenced by `DeleteFeatureValues()`.

17.3.5.54 `PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray)`

Definition at line 251 of file `anamodutils.c`.

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

Referenced by `DeleteFeatureValues()`.

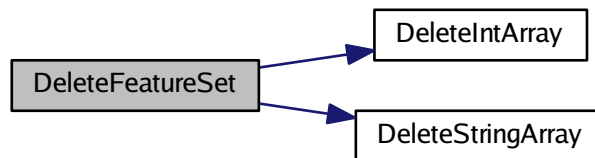
17.3.5.55 `PetscErrorCode DeleteFeatureSet (FeatureSet set)`

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:



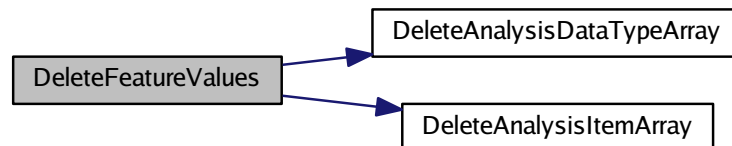
17.3.5.56 `PetscErrorCode DeleteFeatureValues (FeatureValues values)`

Free a featurevalues object. See [Feature sets](#)

Definition at line 149 of file `feature.c`.

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

Here is the call graph for this function:



17.3.5.57 `PetscErrorCode DeleteIntArray (IntArray)`

Definition at line 31 of file `anamodutils.c`.

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

Referenced by `DeleteFeatureSet()`.

17.3.5.58 PetscErrorCode DeleteStringArray (StringArray)

Definition at line 135 of file anamodutils.c.

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

Referenced by `DeleteFeatureSet()`.

17.3.5.59 PetscErrorCode DeRegisterCategory (const char * cat)

Deallocate the storage for a particular category of analysis modules. No longer needed.

Definition at line 409 of file module_functions.c.

17.3.5.60 PetscErrorCode DeregisterModules (void)

This function is no longer needed

Definition at line 425 of file module_functions.c.

17.3.5.61 PetscErrorCode DestroyCategoryObject (categoryobject obj)

Deallocate a category object. See also [CreateCategoryObject\(\)](#).

Definition at line 79 of file category.c.

References `CHECKVALIDCATEGORY`, `categoryobject_::componentnames`, `categoryobject_::components`, `DestroyComponentObject()`, `categoryobject_::name`, `categoryobject_::ncomponents`, and `categoryobject_::types`.

Referenced by `FreeCategoryObjects()`.

Here is the call graph for this function:

**17.3.5.62 PetscErrorCode DestroyComponentObject (componentobject)**

Definition at line 30 of file component.c.

References `CHECKVALIDCOMPONENT`, and `componentobject_::name`.

Referenced by DestroyCategoryObject().

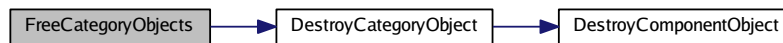
17.3.5.63 PetscErrorCode FreeCategoryObjects ()

Definition at line 40 of file category.c.

References categorynames, DestroyCategoryObject(), and ncategories.

Referenced by AnaModFinalize().

Here is the call graph for this function:



17.3.5.64 PetscErrorCode GetCategories (int *, const char ***)

Definition at line 150 of file category.c.

References categorynames, and ncategories.

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

17.3.5.65 PetscErrorCode GetCategory (const char * cat, categoryobject * catg, PetscBool * f)

Return a named category

Definition at line 138 of file category.c.

References GetCategoryIndex().

Referenced by CategoryGetModules(), CategoryGetOptionFunction(), ComputeOrRetrieveQuantity(), DeclareCategoryOptionFunction(), GetDataID(), GetDataType(), HasComputeCategory(), HasComputeModule(), and HasQuantity().

Here is the call graph for this function:



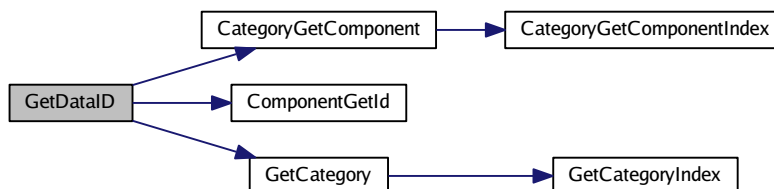
17.3.5.66 PetscErrorCode GetDataID (const char *, const char *, int *, PetscBool *)

Definition at line 589 of file module_functions.c.

References CategoryGetComponent(), ComponentGetId(), and GetCategory().

Referenced by AvgDiagDist(), 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(), Kappa(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyReallm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUstruct(), 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:



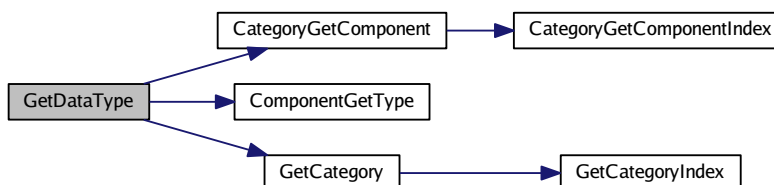
17.3.5.67 `PetscErrorCode GetDataType (const char *, const char *, AnalysisDataType *, PetscBool *)`

Definition at line 612 of file `module_functions.c`.

References `CategoryGetComponent()`, `ComponentGetType()`, and `GetCategory()`.

Referenced by `analyze_matrix()`.

Here is the call graph for this function:



17.3.5.68 `PetscErrorCode GetFeatureValue (FeatureValues values, int index, AnalysisItem * val, PetscBool * 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 207 of file `feature.c`.

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

Here is the call graph for this function:



17.3.5.69 `PetscErrorCode GetFirstCategory (const char **, PetscBool *)`

Definition at line 276 of file `category.c`.

References `CATCMP_ENABLE`, `categoryreadout`, `categoryobject_::name`, and `ncategories`.

Referenced by `analyze_matrix()`.

17.3.5.70 `PetscErrorCode GetNextCategory (const char **, PetscBool *)`

Definition at line 294 of file `category.c`.

References `CATCMP_ENABLE`, `categoryreadout`, `categoryobject_::name`, and `ncategories`.

Referenced by `analyze_matrix()`.

17.3.5.71 `PetscErrorCode GetOrCreateCategory (const char * cat, categoryobject * catg)`

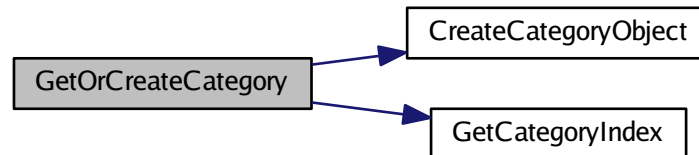
Return a named category, creating it if necessary

Definition at line 119 of file `category.c`.

References `categorynames`, `CreateCategoryObject()`, `GetCategoryIndex()`, `maxcategories`, `categoryobject_::name`, and `ncategories`.

Referenced by CategoryComponentSetModule().

Here is the call graph for this function:



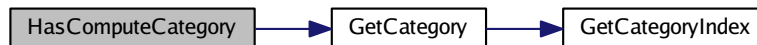
17.3.5.72 `PetscErrorCode HasComputeCategory (const char * cat, PetscBool * f)`

Query whether a specified category has been declared.

Definition at line 451 of file `module_functions.c`.

References `GetCategory()`.

Here is the call graph for this function:



17.3.5.73 `PetscErrorCode HasComputeModule (const char * cat, const char * cmp, PetscBool * f)`

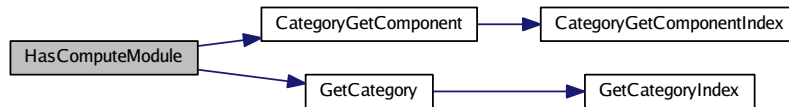
Query whether a specified module exists inside a specified category. The category need not itself have been declared.

Definition at line 465 of file `module_functions.c`.

References `CategoryGetComponent()`, and `GetCategory()`.

Referenced by `LoBand()`, and `UpBand()`.

Here is the call graph for this function:



17.3.5.74 PetscErrorCode HasQuantity (AnaModNumericalProblem *prob*, const char * *cat*, const char * *cmp*, PetscBool * *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. Use [HasComputeModule\(\)](#) to test whether a category and module is known to the system.

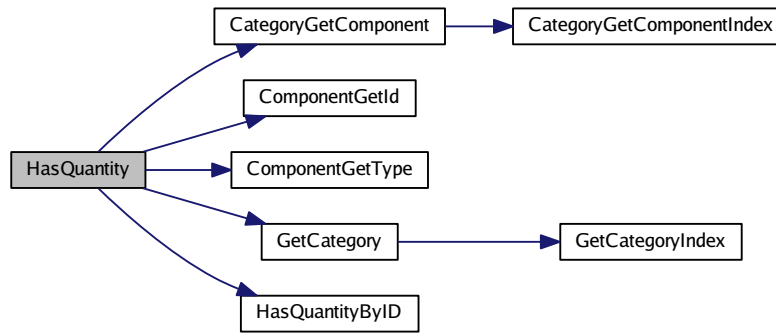
See [the page on attached quantities](#) for an explanation of the mechanism behind this routine.

Definition at line 646 of file `module_functions.c`.

References `CategoryGetComponent()`, `ComponentGetId()`, `ComponentGetType()`, `GetCategory()`, `HasQuantityById()`, and `id`.

Referenced by `computennz()`.

Here is the call graph for this function:



17.3.5.75 `PetscErrorCode HasQuantityById (AnaModNumericalProblem prob, int id, AnalysisDataType type, PetscBool * f)`

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

Definition at line 667 of file `module_functions.c`.

References `ANALYSISDBLARRAY`, `ANALYSISDOUBLE`, `ANALYSISINTARRAY`, and `ANALYSISINTEGER`.

Referenced by `HasQuantity()`.

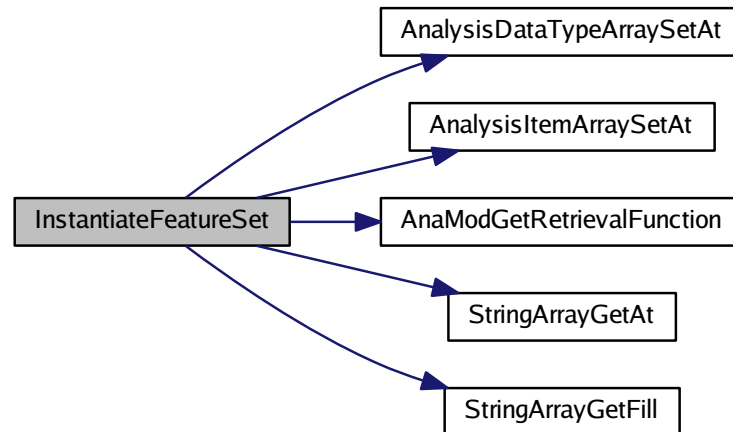
17.3.5.76 `PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem prob, FeatureSet set, FeatureValues values)`

Fill in a featurevalues object. See [Feature sets](#)

Definition at line 164 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:



17.3.5.77 `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`.

17.3.5.78 `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`.

17.3.5.79 `PetscErrorCode IntArrayGetFill (IntArray , int *)`

17.3.5.80 `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`.

17.3.5.81 `PetscErrorCode IntArrayTryGetAt (IntArray array, int idx, int * val, PetscBool * 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`.

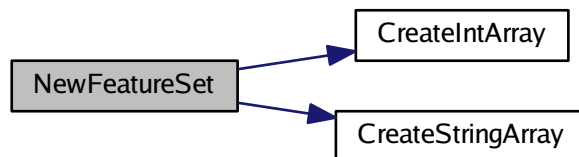
17.3.5.82 `PetscErrorCode NewFeatureSet (FeatureSet * set)`

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:



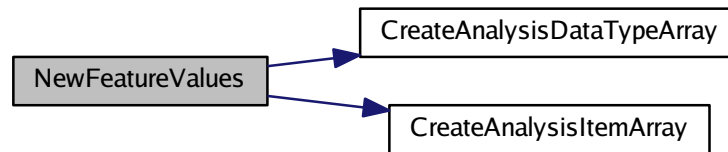
17.3.5.83 PetscErrorCode NewFeatureValues (FeatureValues * values)

Allocate a featurevalues object. See [Feature sets](#)

Definition at line 133 of file feature.c.

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

Here is the call graph for this function:

**17.3.5.84 PetscErrorCode PurgeAttachedArrays (Mat A)****17.3.5.85 PetscErrorCode QuantityAsString (AnalysisItem * q, AnalysisDataType t, char ** s)**

Generate a character string for a given quantity.

Definition at line 731 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 `ComputeOrRetrieveQuantity()`, and `ReportAnamodContent()`.

17.3.5.86 PetscErrorCode RegisterModule (const char * cat, const char * cmp, AnalysisDataType type, PetscErrorCode(*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *) 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)

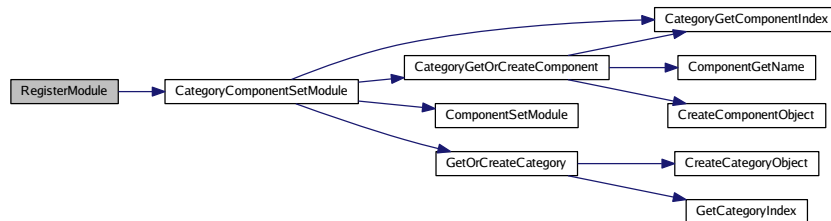
See also [HasComputeModule\(\)](#), [ComputeQuantity\(\)](#).

Definition at line 382 of file `module_functions.c`.

References `AnaModIsInitialized`, `CategoryComponentSetModule()`, and `id`.

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

Here is the call graph for this function:



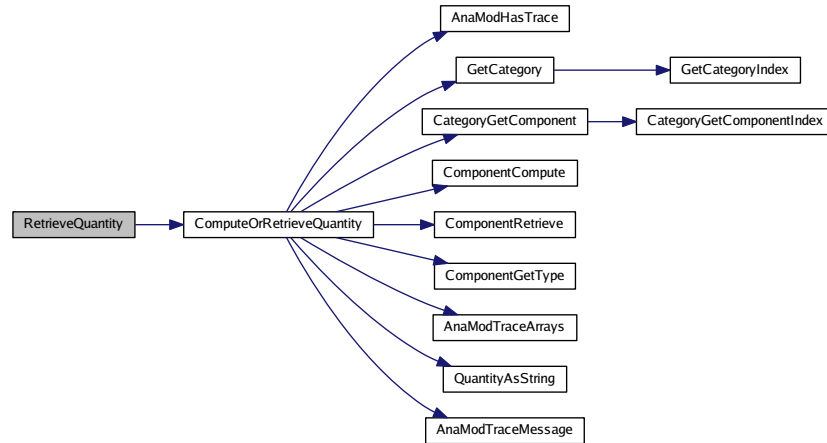
17.3.5.87 `PetscErrorCode RetrieveQuantity (AnaModNumericalProblem prob, const char * cat, const char * cmp, AnalysisItem * res, int * rreslen, PetscBool * success)`

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 577 of file `module_functions.c`.

References `ComputeOrRetrieveQuantity()`, and `MODE_RETRIEVE`.

Here is the call graph for this function:



17.3.5.88 `PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem prob, int id, AnalysisDataType type, AnalysisItem * result, PetscBool * f)`

See also [HasQuantityByID\(\)](#)

Definition at line 698 of file `module_functions.c`.

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

17.3.5.89 `PetscErrorCode StringArrayAdd (StringArray array, const 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()`.

17.3.5.90 `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()`.

17.3.5.91 `PetscErrorCode StringArrayGetFill (StringArray , int *)`

Definition at line 166 of file anamodutils.c.

References `StringArray_::fill`.

Referenced by `InstantiateFeatureSet()`.

17.3.5.92 `PetscErrorCode StringArraySetAt (StringArray array, int idx, const 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`.

17.3.5.93 `PetscErrorCode StringArrayTryGetAt (StringArray array, int idx, char ** val, PetscBool * 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.

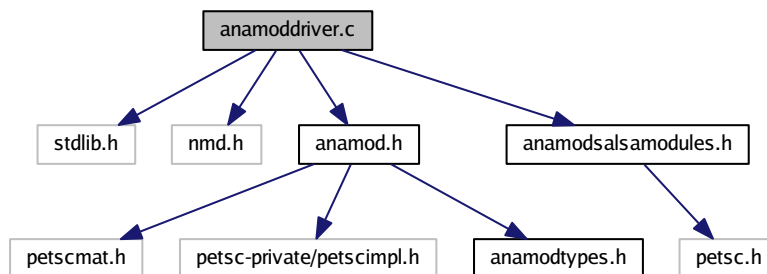
References `StringArray_::alloc`, `StringArray_::data`, and `StringArray_::has`.

17.3.5.94 `PetscErrorCode TabReportModules (Mat, char **, char **, int)`

17.4 anamoddriver.c File Reference

```
#include <stdlib.h> #include "nmd.h" #include "anamod.-
h" #include "anamodsalsamodules.h" Include dependency graph for
```


anamoddriver.c:



Functions

- static PetscErrorCode [get_matrix](#) (Mat *A, PetscBool *success)
- PetscErrorCode [analyze_matrix](#) (Mat A, NMD_metadata nmd)
- static PetscErrorCode [usage](#) (MPI_Comm comm)
- int [main](#) (int argc, char **argv)

17.4.1 Function Documentation

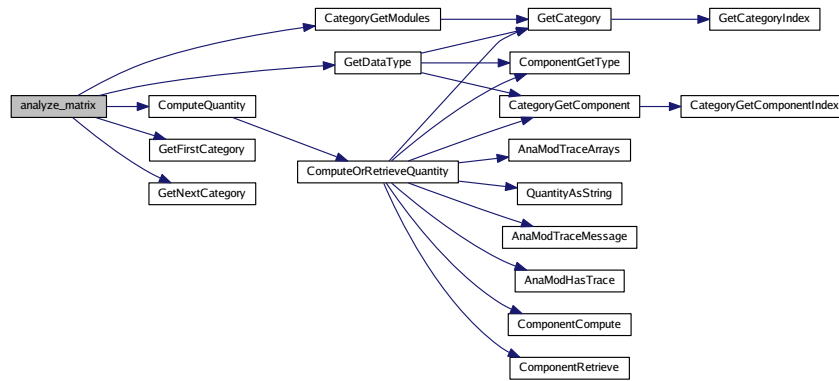
17.4.1.1 PetscErrorCode analyze_matrix (Mat A, NMD_metadata nmd)

Definition at line 48 of file `anamoddriver.c`.

References `CategoryGetModules()`, `ComputeQuantity()`, `GetDataType()`, `GetFirstCategory()`, `GetNextCategory()`, and `FeatureValues_::types`.

Referenced by `main()`.

Here is the call graph for this function:



17.4.1.2 static PetscErrorCode get_matrix (Mat * A, PetscBool * success) [static]

Definition at line 12 of file anamoddriver.c.

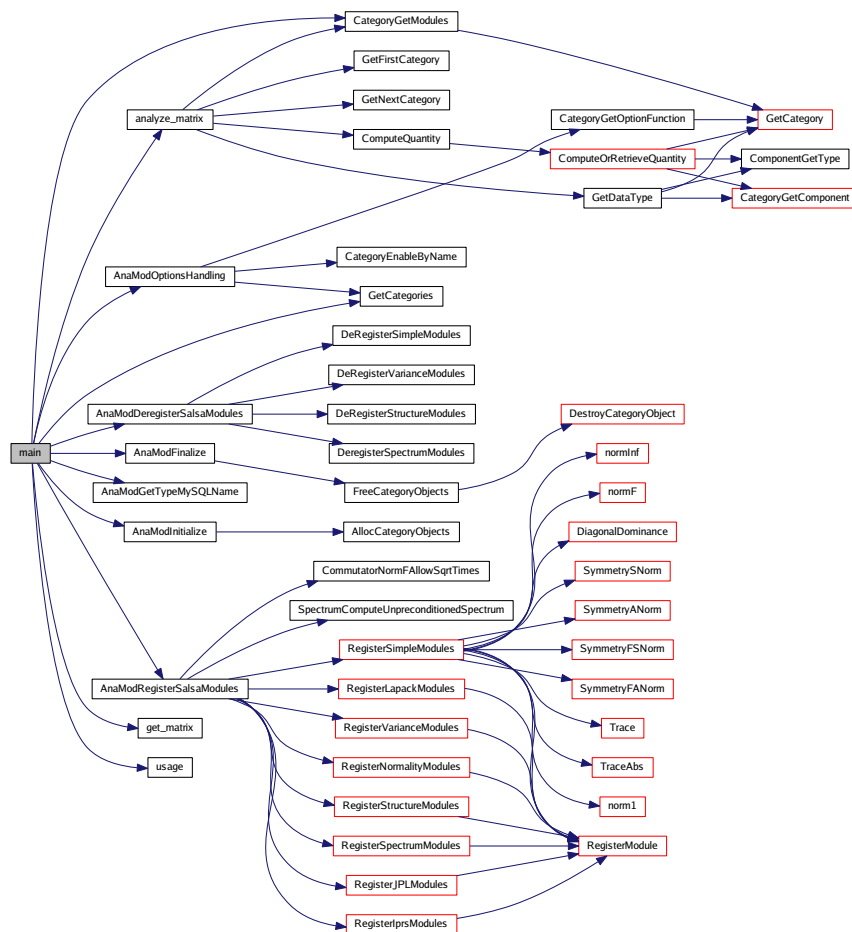
Referenced by main().

17.4.1.3 int main (int argc, char ** argv)

Definition at line 91 of file anamoddriver.c.

References analyze_matrix(), AnaModDeregisterSalsaModules(), AnaModFinalize(), - AnaModGetTypeMySQLName(), AnaModInitialize(), AnaModOptionsHandling(), AnaModRegisterSalsaModules(), CategoryGetModules(), get_matrix(), GetCategories(), - FeatureValues_::types, and usage().

Here is the call graph for this function:



17.4.1.4 static PetscErrorCode usage (MPI_Comm comm) [static]

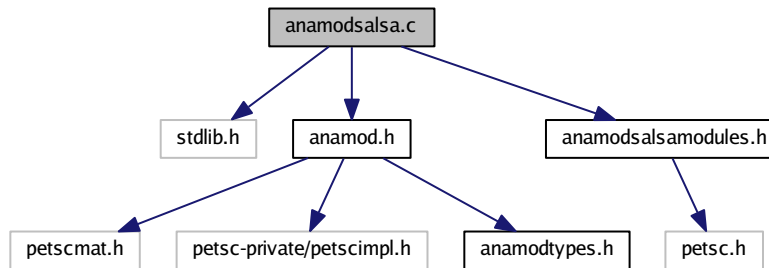
Definition at line 76 of file anamoddriver.c.

Referenced by main().

17.5 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](#) ()

17.5.1 Function Documentation

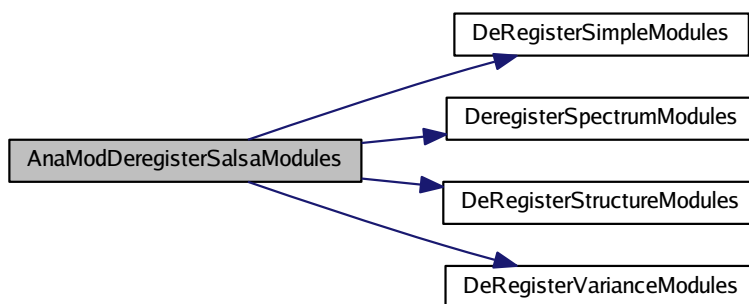
17.5.1.1 PetscErrorCode AnaModDeregisterSalsaModules (void)

Definition at line 30 of file anamodsalsa.c.

References [DeRegisterSimpleModules\(\)](#), [DeregisterSpectrumModules\(\)](#), [DeRegisterStructureModules\(\)](#), and [DeRegisterVarianceModules\(\)](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



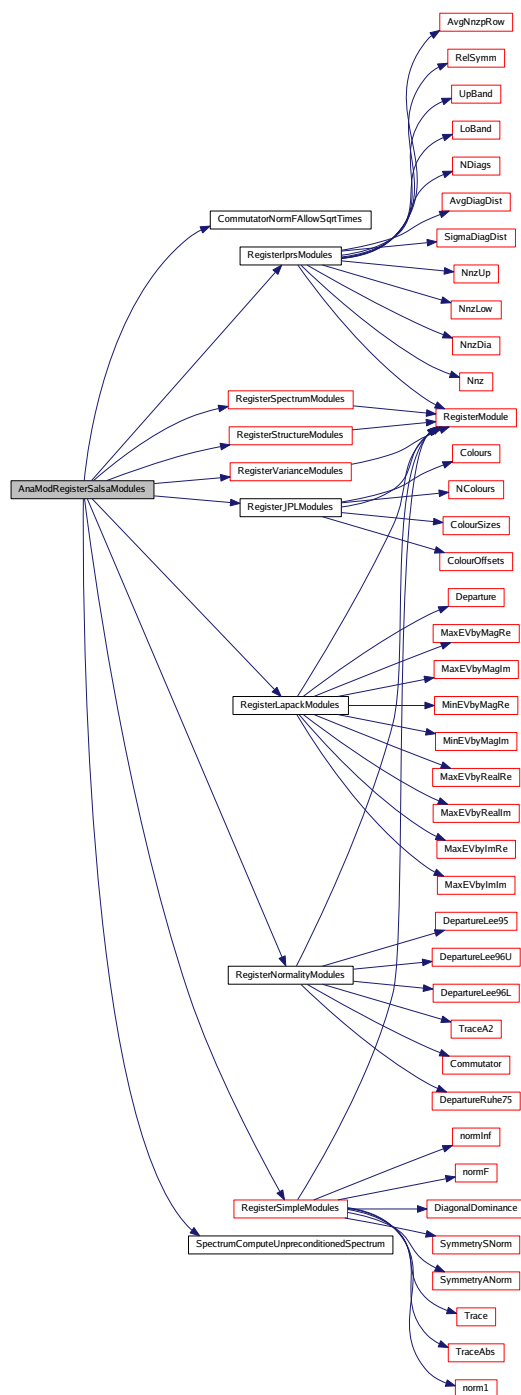
17.5.1.2 PetscErrorCode AnaModRegisterSalsaModules (void)

Definition at line 7 of file `anamodsalsa.c`.

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

Referenced by `main()`.

Here is the call graph for this function:



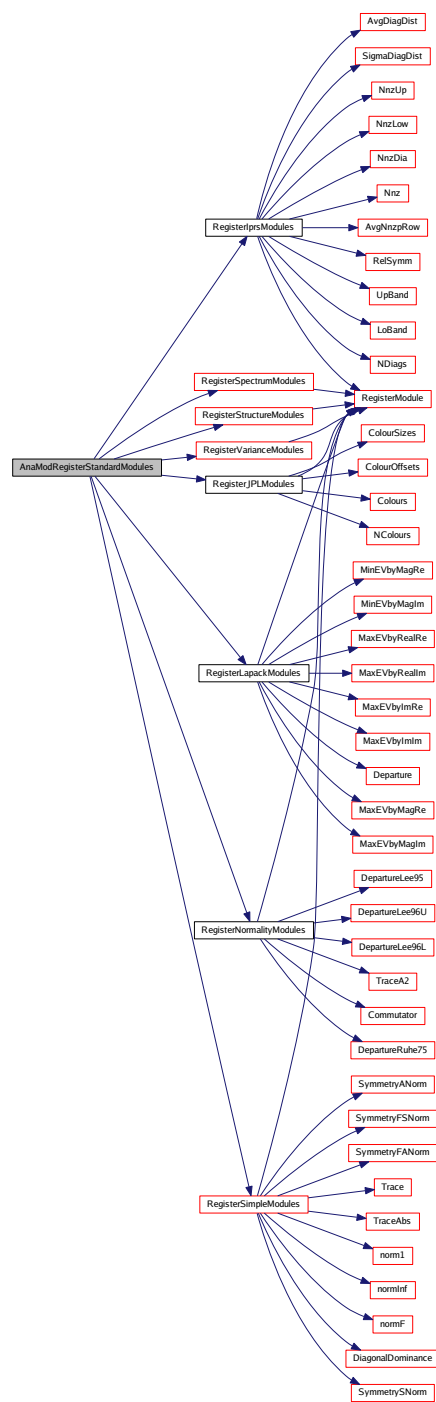
17.5.1.3 PetscErrorCode AnaModRegisterStandardModules ()

Register all standard and nonstandard analysis modules.

Definition at line 47 of file anamodsalsa.c.

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

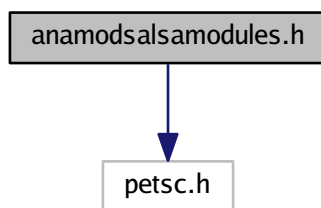
Here is the call graph for this function:



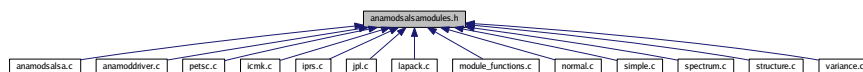
17.6 anamodsalsamodules.h File Reference

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)

17.6.1 Detailed Description

Prototypes for using the standard modules.

Definition in file [anamodsalsamodules.h](#).

17.6.2 Define Documentation

17.6.2.1 `#define DIAGONAL_INDEFINITE 0`

Definition at line 41 of file [anamodsalsamodules.h](#).

Referenced by [ComputeDiagonal\(\)](#).

17.6.2.2 `#define DIAGONAL_INDEFINITE_WITH_ZEROS 11`

Definition at line 45 of file [anamodsalsamodules.h](#).

17.6.2.3 `#define DIAGONAL_NEGATIVE -2`

Definition at line 43 of file [anamodsalsamodules.h](#).

Referenced by [ComputeDiagonal\(\)](#).

17.6.2.4 `#define DIAGONAL_NONNEGATIVE 1`

Definition at line 40 of file [anamodsalsamodules.h](#).

Referenced by [ComputeDiagonal\(\)](#).

17.6.2.5 #define DIAGONAL_NONPOSITIVE -1

Definition at line 42 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

17.6.2.6 #define DIAGONAL_POSITIVE 2

Definition at line 39 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

17.6.2.7 #define DIAGONAL_ZERO 10

Definition at line 44 of file anamodsalsamodules.h.

Referenced by ComputeDiagonal().

17.6.3 Function Documentation

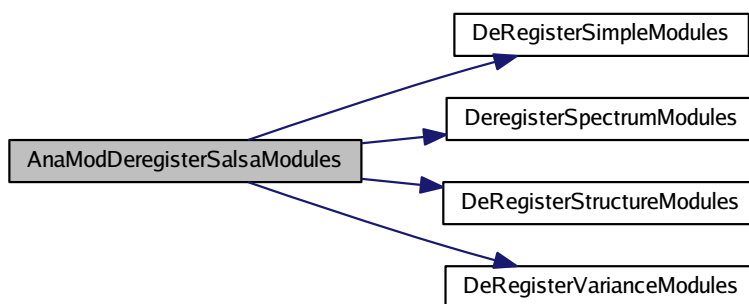
17.6.3.1 PetscErrorCode AnaModDeregisterSalsaModules (void)

Definition at line 30 of file anamodsalsa.c.

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

Referenced by main().

Here is the call graph for this function:



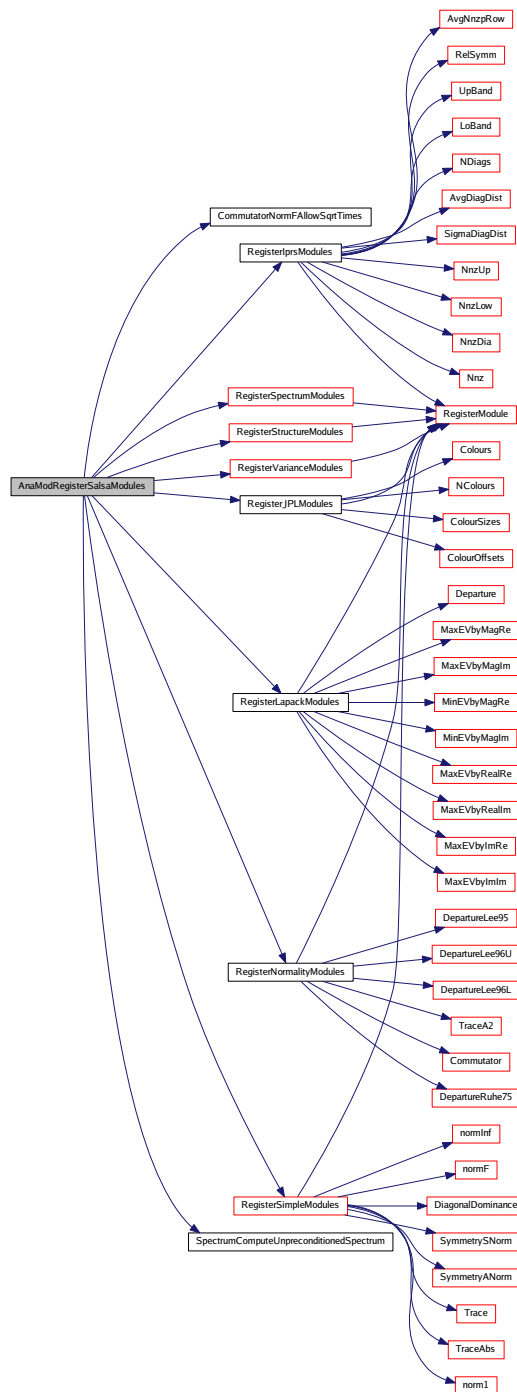
17.6.3.2 PetscErrorCode AnaModRegisterSalsaModules (void)

Definition at line 7 of file anamodsalsa.c.

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

Referenced by main().

Here is the call graph for this function:



17.6.3.3 PetscErrorCode CommutatorNormFAllowSqrtTimes (int *n*)

Definition at line 283 of file normal.c.

Referenced by AnaModRegisterSalsaModules().

17.6.3.4 PetscErrorCode DeRegisterSimpleModules (void)

Definition at line 694 of file simple.c.

Referenced by AnaModDeregisterSalsaModules().

17.6.3.5 PetscErrorCode DeregisterSpectrumModules (void)

Definition at line 1414 of file spectrum.c.

Referenced by AnaModDeregisterSalsaModules().

17.6.3.6 PetscErrorCode DeRegisterStructureModules (void)

Definition at line 720 of file structure.c.

Referenced by AnaModDeregisterSalsaModules().

17.6.3.7 PetscErrorCode DeRegisterVarianceModules (void)

Definition at line 333 of file variance.c.

Referenced by AnaModDeregisterSalsaModules().

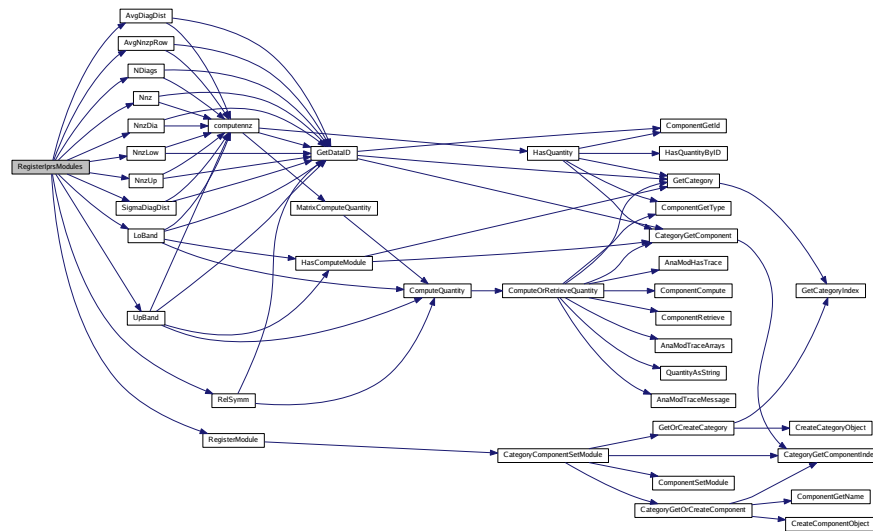
17.6.3.8 PetscErrorCode RegisterIprsModules (void)

Definition at line 535 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:



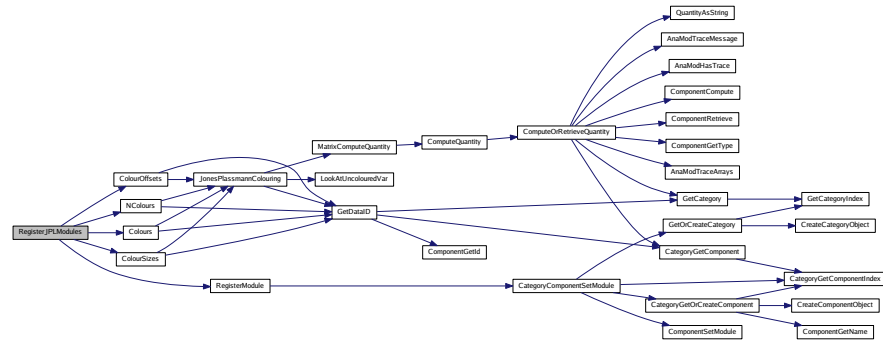
17.6.3.9 PetscErrorCode RegisterJPLModules (void)

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:



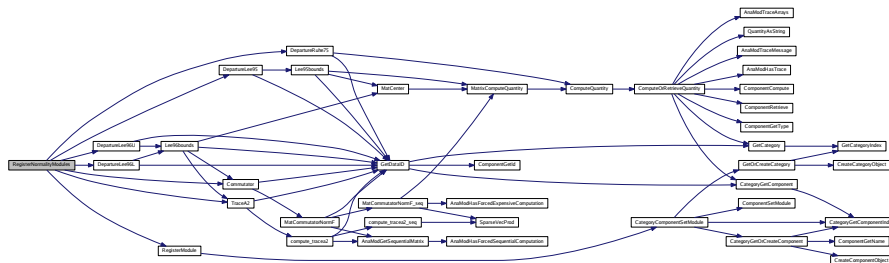
17.6.3.10 PetscErrorCode RegisterNormalityModules (void)

Definition at line 583 of file normal.c.

References ANALYSISDOUBLE, Commutator(), DepartureLee95(), DepartureLee96-L(), DepartureLee96U(), DepartureRuhe75(), RegisterModule(), and TraceA2().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:



17.6.3.11 PetscErrorCode RegisterSimpleModules (void)

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

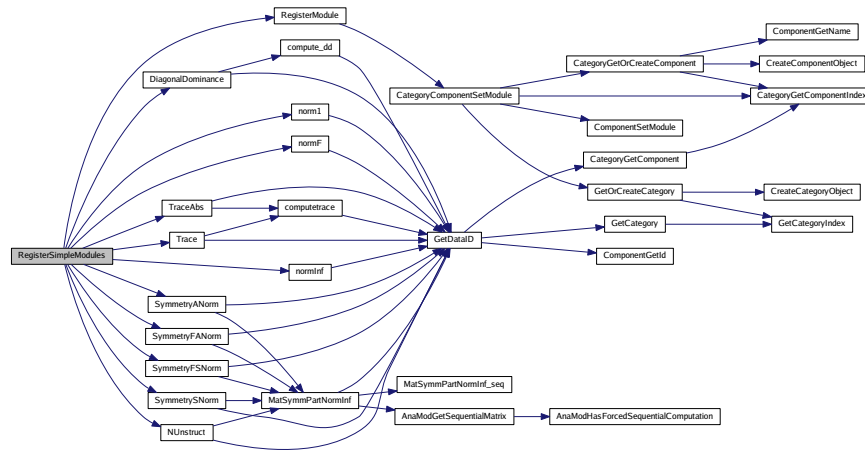
Definition at line 657 of file simple.c.

References ANALYSISDOUBLE, ANALYSISINTEGER, and 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:



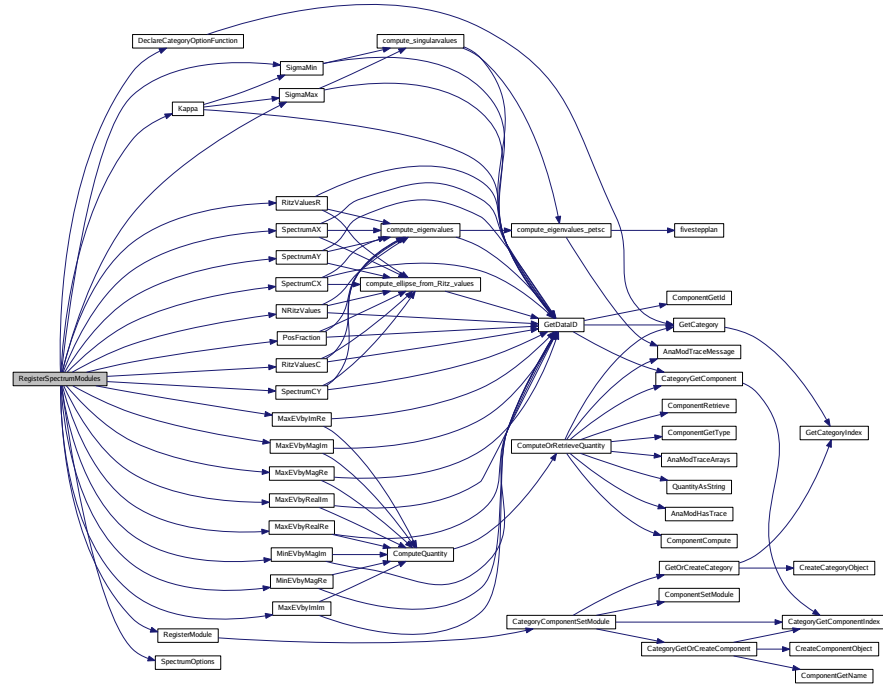
17.6.3.12 PetscErrorCode RegisterSpectrumModules (void)

Definition at line 1343 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:



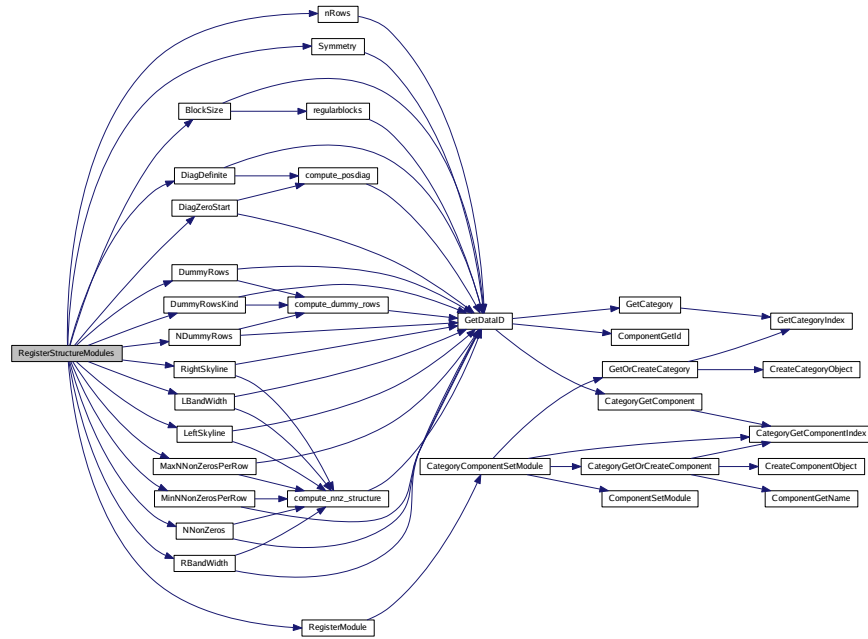
17.6.3.13 PetscErrorCode RegisterStructureModules (void)

Definition at line 671 of file structure.c.

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

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:



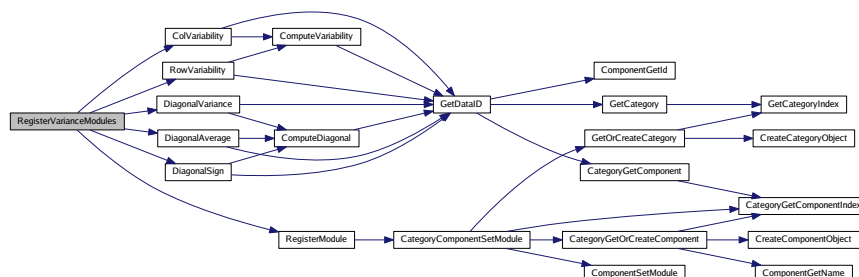
17.6.3.14 PetscErrorCode RegisterVarianceModules (void)

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:



17.6.3.15 PetscErrorCode SpectrumComputePreconditionedSpectrum (void)

Definition at line 95 of file spectrum.c.

References use_prec.

17.6.3.16 PetscErrorCode SpectrumComputeUnpreconditionedSpectrum (void)

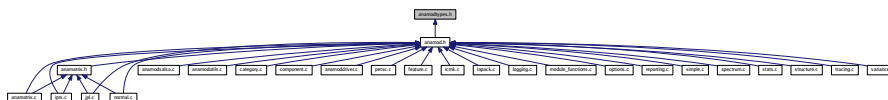
Definition at line 104 of file spectrum.c.

References use_prec.

Referenced by AnaModRegisterSalsaModules().

17.7 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`

17.7.1 Detailed Description

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

Definition in file [anamodtypes.h](#).

17.7.2 Define Documentation

17.7.2.1 `#define ANALYSISDBLARRAY` 9

Definition at line 56 of file [anamodtypes.h](#).

Referenced by [ComponentRetrieve\(\)](#), [HasQuantityByID\(\)](#), [QuantityAsString\(\)](#), [RegisterSpectrumModules\(\)](#), and [RetrieveQuantityByID\(\)](#).

17.7.2.2 `#define ANALYSISDOUBLE` 4

Definition at line 51 of file [anamodtypes.h](#).

Referenced by [ComponentRetrieve\(\)](#), [HasQuantityByID\(\)](#), [QuantityAsString\(\)](#), [RegisterIprsModules\(\)](#), [RegisterLapackModules\(\)](#), [RegisterNormalityModules\(\)](#), [RegisterSimpleModules\(\)](#), [RegisterSpectrumModules\(\)](#), [RegisterVarianceModules\(\)](#), and [RetrieveQuantityByID\(\)](#).

17.7.2.3 #define ANALYSISFLARRAY 10

Definition at line 57 of file anamodtypes.h.

17.7.2.4 #define ANALYSISFLOAT 5

Definition at line 52 of file anamodtypes.h.

17.7.2.5 #define ANALYSISINTARRAY 8

Definition at line 55 of file anamodtypes.h.

Referenced by ComponentRetrieve(), ComputeOrRetrieveQuantity(), HasQuantityByID(), QuantityAsString(), RegisterICMKModules(), RegisterJPLModules(), RegisterStructureModules(), and RetrieveQuantityByID().

17.7.2.6 #define ANALYSISINTEGER 3

Definition at line 50 of file anamodtypes.h.

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

17.7.2.7 #define ANALYSISMETA 6

Definition at line 53 of file anamodtypes.h.

17.7.2.8 #define ANALYSISNONE 1

Definition at line 48 of file anamodtypes.h.

17.7.2.9 #define ANALYSISSTRING 2

Definition at line 49 of file anamodtypes.h.

Referenced by QuantityAsString(), and RegisterStatsModules().

17.7.2.10 #define ANALYSISVOID 7

Definition at line 54 of file anamodtypes.h.

17.7.3 Typedef Documentation

17.7.3.1 typedef int AnalysisDataType

Definition at line 46 of file anamodtypes.h.

17.7.3.2 typedef struct AnalysisDataTypeArray_* AnalysisDataTypeArray

Definition at line 22 of file anamodtypes.h.

17.7.3.3 typedef struct AnalysisItemArray_* AnalysisItemArray

Definition at line 21 of file anamodtypes.h.

17.7.3.4 typedef void* AnaModNumericalProblem

Definition at line 10 of file anamodtypes.h.

17.7.3.5 typedef struct IntArray_* IntArray

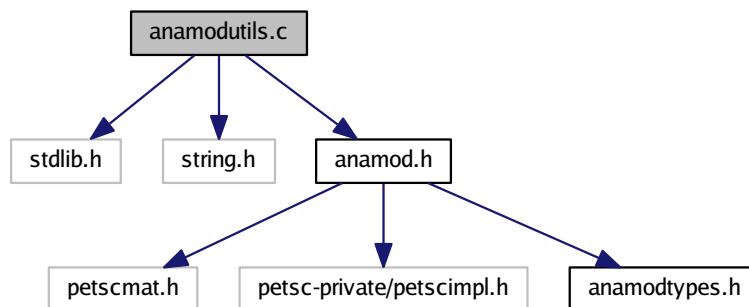
Definition at line 19 of file anamodtypes.h.

17.7.3.6 typedef struct StringArray_* StringArray

Definition at line 20 of file anamodtypes.h.

17.8 anamodutils.c File Reference

```
#include <stdlib.h> #include "string.h" #include "anamod.h"
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, PetscBool *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, const char *val, int *idx)
- PetscErrorCode [StringArrayGetFill](#) ([StringArray](#) array, int *idx)
- PetscErrorCode [StringArraySetAt](#) ([StringArray](#) array, int idx, const char *val)
- PetscErrorCode [StringArrayTryGetAt](#) ([StringArray](#) array, int idx, char **val, PetscBool *has)
- PetscErrorCode [StringArrayGetAt](#) ([StringArray](#) array, int idx, char **val)
- PetscErrorCode [CreateAnalysisItemArray](#) (const 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, PetscBool *has)
- PetscErrorCode [AnalysisItemArrayGetAt](#) ([AnalysisItemArray](#) array, int idx, - [AnalysisItem](#) *val)
- PetscErrorCode [CreateAnalysisDataTypeArray](#) (const 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, PetscBool *has)
- PetscErrorCode [AnalysisDataTypeArrayGetAt](#) ([AnalysisDataTypeArray](#) array, int idx, [AnalysisDataType](#) *val)

17.8.1 Define Documentation

17.8.1.1 #define NALLOC 70

Definition at line 5 of file anamodutils.c.

Referenced by [CreateAnalysisDataTypeArray\(\)](#), [CreateAnalysisItemArray\(\)](#), [CreateIntArray\(\)](#), and [CreateStringArray\(\)](#).

17.8.2 Function Documentation

17.8.2.1 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 372 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), [AnalysisDataTypeArray_::fill](#), and [AnalysisDataTypeArray_::has](#).

17.8.2.2 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 424 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), and [AnalysisDataTypeArray_::has](#).

17.8.2.3 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 388 of file anamodutils.c.

References [AnalysisDataTypeArray_::alloc](#), [AnalysisDataTypeArray_::data](#), [AnalysisDataTypeArray_::fill](#), and [AnalysisDataTypeArray_::has](#).

Referenced by `InstantiateFeatureSet()`.

17.8.2.4 `PetscErrorCode AnalysisDataTypeArrayTryGetAt (AnalysisDataTypeArray array, int idx, AnalysisDataType * val, PetscBool * 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 410 of file `anamodutils.c`.

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

17.8.2.5 `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 267 of file `anamodutils.c`.

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

17.8.2.6 `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 319 of file `anamodutils.c`.

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

17.8.2.7 `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 283 of file anamodutils.c.

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

Referenced by InstantiateFeatureSet().

17.8.2.8 PetscErrorCode AnalysisItemArrayTryGetAt (AnalysisItemArray array, int idx, AnalysisItem * val, PetscBool * 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 305 of file anamodutils.c.

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

Referenced by GetFeatureValue().

17.8.2.9 PetscErrorCode CreateAnalysisDataTypeArray (const char * name, int size, AnalysisDataTypeArray * array)

Definition at line 338 of file anamodutils.c.

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

Referenced by NewFeatureValues().

17.8.2.10 PetscErrorCode CreateAnalysisItemArray (const char * name, int size, AnalysisItemArray * array)

Definition at line 233 of file anamodutils.c.

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

Referenced by NewFeatureValues().

17.8.2.11 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().

17.8.2.12 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().

17.8.2.13 PetscErrorCode DeleteAnalysisDataTypeArray (AnalysisDataTypeArray *array*)

Definition at line 356 of file anamodutils.c.

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

Referenced by DeleteFeatureValues().

17.8.2.14 PetscErrorCode DeleteAnalysisItemArray (AnalysisItemArray *array*)

Definition at line 251 of file anamodutils.c.

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

Referenced by DeleteFeatureValues().

17.8.2.15 PetscErrorCode DeleteIntArray (IntArray *array*)

Definition at line 31 of file anamodutils.c.

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

Referenced by DeleteFeatureSet().

17.8.2.16 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().

17.8.2.17 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.

17.8.2.18 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.

17.8.2.19 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.

17.8.2.20 PetscErrorCode IntArrayTryGetAt (IntArray array, int idx, int * val, PetscBool * 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.

17.8.2.21 PetscErrorCode StringArrayAdd (StringArray array, const 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()`.

17.8.2.22 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()`.

17.8.2.23 PetscErrorCode StringArrayGetFill (StringArray array, int * idx)

Definition at line 166 of file anamodutils.c.

References `StringArray_::fill`.

Referenced by `InstantiateFeatureSet()`.

17.8.2.24 PetscErrorCode StringArraySetAt (StringArray array, int idx, const 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`.

17.8.2.25 PetscErrorCode StringArrayTryGetAt (StringArray array, int idx, char ** val, PetscBool * 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.

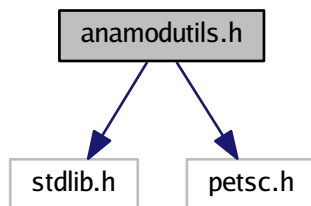
- `has` (output) : true if a value was stored at this index. This argument is allowed to be null.

Definition at line 200 of file `anamodutils.c`.

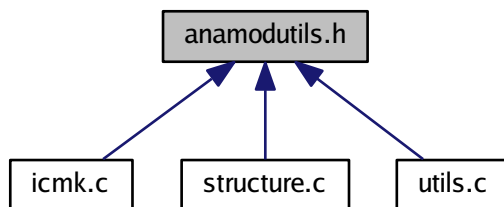
References `StringArray_::alloc`, `StringArray_::data`, and `StringArray_::has`.

17.9 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)

17.9.1 Function Documentation

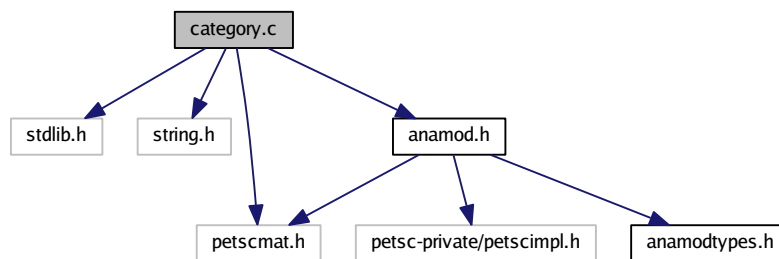
17.9.1.1 int [MPIAllGatherIntV](#) (int * *array*, int *n*, int ** *Array*, int * *N*, MPI_Comm *comm*)

Definition at line 8 of file `utils.c`.

Referenced by `GetUpBiDiagSplits()`, and `MatSplitPoints()`.

17.10 category.c File Reference

```
#include <stdlib.h> #include <string.h> #include "anamod.-
h" #include "petscmat.h" Include dependency graph for category.c:
```



Data Structures

- struct [categoryobject_](#)

Defines

- #define [CATEGORYCOOKIE](#) 983429
- #define [CHECKVALIDCATEGORY](#)(x) [ANAMODCHECKVALID](#)(x, [CATEGORYCOOKIE](#), "category")

Functions

- PetscErrorCode [AllocCategoryObjects](#) ()
- PetscErrorCode [FreeCategoryObjects](#) ()
- PetscErrorCode [CreateCategoryObject](#) (const char *cat, [categoryobject](#) *obj)
- PetscErrorCode [DestroyCategoryObject](#) ([categoryobject](#) obj)
- static PetscErrorCode [GetCategoryIndex](#) (const char *cat, int *icat, PetscBool *flag)
- PetscErrorCode [GetOrCreateCategory](#) (const char *cat, [categoryobject](#) *catg)
- PetscErrorCode [GetCategory](#) (const char *cat, [categoryobject](#) *catg, PetscBool *f)
- PetscErrorCode [GetCategories](#) (int *ncat, const char ***cats)
- PetscErrorCode [CategoryGetComponentIndex](#) ([categoryobject](#) catg, const char *cmp, int *icmp, PetscBool *flag)
- PetscErrorCode [CategoryGetOrCreateComponent](#) ([categoryobject](#) catg, const char *cmp, [componentobject](#) *cmpt)
- PetscErrorCode [CategoryGetComponent](#) ([categoryobject](#) catg, const char *cmp, [componentobject](#) *cmpt, PetscBool *success)
- PetscErrorCode [CategoryComponentSetModule](#) (const char *cat, const char *cmp, [AnalysisDataType](#) type, int id, PetscErrorCode(*)([AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, PetscBool *))
- PetscErrorCode [CategoryGetModules](#) (const char *cat, const char ***ms, - [AnalysisDataType](#) **t, int **id, int *n)
- PetscErrorCode [GetFirstCategory](#) (const char **rname, PetscBool *rfound)
- PetscErrorCode [GetNextCategory](#) (const char **rname, PetscBool *rfound)
- PetscErrorCode [CategoryEnableByName](#) (const char *cat, int mode)
- PetscErrorCode [DeclareCategoryOptionFunction](#) (const char *cat, PetscErrorCode(*)(char *))
- PetscErrorCode [CategoryGetOptionFunction](#) (const char *cat, PetscErrorCode(**)(char *))

Variables

- int [ncategories](#)
- int [maxcategories](#)
- [categoryobject](#) * [categoryobjects](#)
- const char ** [categorynames](#)
- static int [categoryreadout](#) = -1

17.10.1 Define Documentation

17.10.1.1 `#define CATEGORYCOOKIE 983429`

Definition at line 6 of file category.c.

Referenced by `CreateCategoryObject()`.

17.10.1.2 `#define CHECKVALIDCATEGORY(x) ANAMODCHECKVALID(x,CATEGORYCOOKIE,"category")`

Definition at line 7 of file category.c.

Referenced by `CategoryGetComponent()`, `CategoryGetComponentIndex()`, `CategoryGetOrCreateComponent()`, and `DestroyCategoryObject()`.

17.10.2 Function Documentation

17.10.2.1 `PetscErrorCode AllocCategoryObjects ()`

Definition at line 26 of file category.c.

References `categorynames`, `maxcategories`, `MXC`, and `ncategories`.

Referenced by `AnaModInitialize()`.

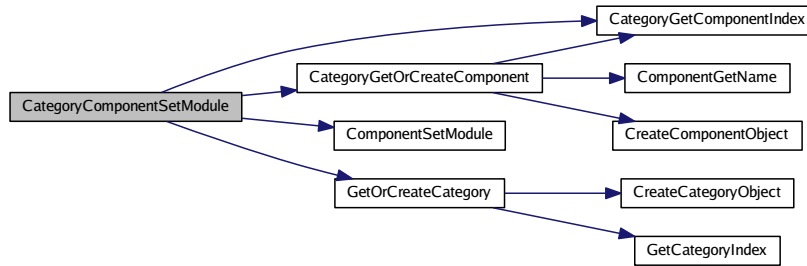
17.10.2.2 `PetscErrorCode CategoryComponentSetModule (const char * cat, const char * cmp, AnalysisDataType type, int id, PetscErrorCode (*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *) f)`

Definition at line 226 of file category.c.

References `CategoryGetComponentIndex()`, `CategoryGetOrCreateComponent()`, `ComponentSetModule()`, `GetOrCreateCategory()`, and `categoryobject_::types`.

Referenced by `RegisterModule()`.

Here is the call graph for this function:



17.10.2.3 PetscErrorCode CategoryEnableByName (const char * cat, int mode)

Mark a category as enabled/disabled. Values (CatCmpEnableMode):

- 0 : enable
- 1 : skip from loops
- 2 : skip altogether

Definition at line 317 of file category.c.

References categoryobject_::enabled, name, and ncategories.

Referenced by AnaModOptionsHandling().

17.10.2.4 PetscErrorCode CategoryGetComponent (categoryobject catg, const char * cmp, componentobject * cmpt, PetscBool * success)

Retrieve a component.

See also [CategoryGetOrCreateComponent\(\)](#), [CategoryGetComponentIndex\(\)](#).

Definition at line 209 of file category.c.

References CategoryGetComponentIndex(), CHECKVALIDCATEGORY, and categoryobject_::components.

Referenced by ComputeOrRetrieveQuantity(), GetDataID(), GetDataType(), HasComputeModule(), and HasQuantity().

Here is the call graph for this function:



17.10.2.5 PetscErrorCode CategoryGetComponentIndex (categoryobject *catg*, const char * *cmp*, int * *icmp*, PetscBool * *flag*)

Test for presence of a component in a category, and return its index if present. The index parameter can be null.

Definition at line 164 of file category.c.

References CHECKVALIDCATEGORY, categoryobject_::componentnames, and categoryobject_::ncomponents.

Referenced by CategoryComponentSetModule(), CategoryGetComponent(), and - CategoryGetOrCreateComponent().

17.10.2.6 PetscErrorCode CategoryGetModules (const char * *cat*, const 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

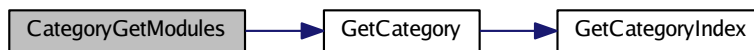
See also [GetCategories\(\)](#) and [HasComputeCategory\(\)](#).

Definition at line 256 of file category.c.

References `categoryobject_::componentnames`, `GetCategory()`, `categoryobject_::ncomponents`, and `categoryobject_::types`.

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

Here is the call graph for this function:



17.10.2.7 `PetscErrorCode CategoryGetOptionFunction (const char * cat, PetscErrorCode(**)(char *) f)`

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

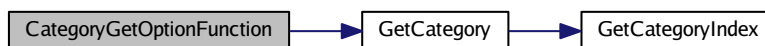
See [DeclareCategoryOptionFunction\(\)](#), `GetCategoryOptionFunction()`, [AnaModOptionsHandling\(\)](#) and section `optionsfile`.

Definition at line 358 of file `category.c`.

References `GetCategory()`, and `categoryobject_::optionfunction`.

Referenced by `AnaModOptionsHandling()`.

Here is the call graph for this function:



17.10.2.8 `PetscErrorCode CategoryGetOrCreateComponent (categoryobject catg, const char * cmp, componentobject * cmpt)`

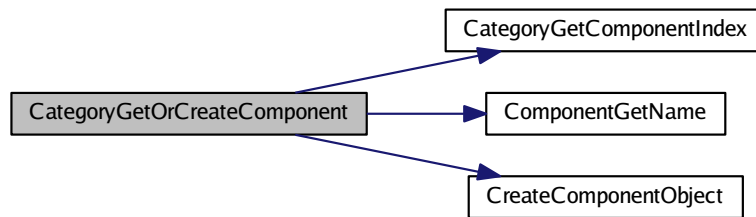
Definition at line 184 of file `category.c`.

References `CategoryGetComponentIndex()`, `CHECKVALIDCATEGORY`, `ComponentGetName()`, `categoryobject_::componentnames`, `categoryobject_::components`, -

CreateComponentObject(), categoryobject_::maxcomponents, and categoryobject_::ncomponents.

Referenced by CategoryComponentSetModule().

Here is the call graph for this function:



17.10.2.9 PetscErrorCode CreateCategoryObject (const char * cat, categoryobject * obj)

Create a category object with a given name. See also [DestroyCategoryObject\(\)](#).

Definition at line 57 of file category.c.

References CATCMP_ENABLE, CATEGORYCOOKIE, categoryobject_::componentnames, categoryobject_::components, categoryobject_::cookie, categoryobject_::enabled, categoryobject_::maxcomponents, MXC, categoryobject_::name, categoryobject_::ncomponents, and categoryobject_::types.

Referenced by GetOrCreateCategory().

17.10.2.10 PetscErrorCode DeclareCategoryOptionFunction (const 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 optionsfile.

Definition at line 338 of file category.c.

References [GetCategory\(\)](#), and categoryobject_::optionfunction.

Referenced by [RegisterSpectrumModules\(\)](#).

Here is the call graph for this function:



17.10.2.11 PetscErrorCode DestroyCategoryObject (categoryobject *obj*)

Deallocate a category object. See also [CreateCategoryObject\(\)](#).

Definition at line 79 of file `category.c`.

References `CHECKVALIDCATEGORY`, `categoryobject_::componentnames`, `categoryobject_::components`, `DestroyComponentObject()`, `categoryobject_::name`, `categoryobject_::ncomponents`, and `categoryobject_::types`.

Referenced by `FreeCategoryObjects()`.

Here is the call graph for this function:



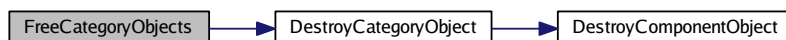
17.10.2.12 PetscErrorCode FreeCategoryObjects ()

Definition at line 40 of file `category.c`.

References `categorynames`, `DestroyCategoryObject()`, and `ncategories`.

Referenced by `AnaModFinalize()`.

Here is the call graph for this function:



17.10.2.13 `PetscErrorCode GetCategories (int * ncat, const char *** cats)`

Definition at line 150 of file `category.c`.

References `categorynames`, and `ncategories`.

Referenced by `AnaModOptionsHandling()`, `AnaModShowOptions()`, `main()`, and `Report-AnamodContent()`.

17.10.2.14 `PetscErrorCode GetCategory (const char * cat, categoryobject * catg, PetscBool * f)`

Return a named category

Definition at line 138 of file `category.c`.

References `GetCategoryIndex()`.

Referenced by `CategoryGetModules()`, `CategoryGetOptionFunction()`, `ComputeOrRetrieveQuantity()`, `DeclareCategoryOptionFunction()`, `GetDataID()`, `GetDataType()`, `HasComputeCategory()`, `HasComputeModule()`, and `HasQuantity()`.

Here is the call graph for this function:



17.10.2.15 `static PetscErrorCode GetCategoryIndex (const char * cat, int * icat, PetscBool * flag) [static]`

Test for existence of a category, and return its index if present. Both output parameters can be null.

Definition at line 100 of file category.c.

References name, and ncategories.

Referenced by GetCategory(), and GetOrCreateCategory().

17.10.2.16 `PetscErrorCode GetFirstCategory (const char ** rname, PetscBool * rfound)`

Definition at line 276 of file category.c.

References CATCMP_ENABLE, categoryreadout, categoryobject_::name, and ncategories.

Referenced by analyze_matrix().

17.10.2.17 `PetscErrorCode GetNextCategory (const char ** rname, PetscBool * rfound)`

Definition at line 294 of file category.c.

References CATCMP_ENABLE, categoryreadout, categoryobject_::name, and ncategories.

Referenced by analyze_matrix().

17.10.2.18 `PetscErrorCode GetOrCreateCategory (const char * cat, categoryobject * catg)`

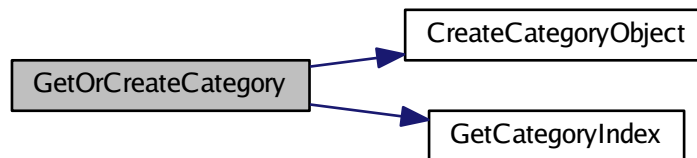
Return a named category, creating it if necessary

Definition at line 119 of file category.c.

References categorynames, CreateCategoryObject(), GetCategoryIndex(), maxcategories, categoryobject_::name, and ncategories.

Referenced by CategoryComponentSetModule().

Here is the call graph for this function:



17.10.3 Variable Documentation

17.10.3.1 `const char** categorynames`

Definition at line 22 of file `category.c`.

Referenced by `AllocCategoryObjects()`, `FreeCategoryObjects()`, `GetCategories()`, and `GetOrCreateCategory()`.

17.10.3.2 `categoryobject* categoryobjects`

Definition at line 279 of file `module_functions.c`.

17.10.3.3 `int categoryreadout = -1` [static]

Definition at line 272 of file `category.c`.

Referenced by `GetFirstCategory()`, and `GetNextCategory()`.

17.10.3.4 `int maxcategories`

Definition at line 20 of file `category.c`.

Referenced by `AllocCategoryObjects()`, and `GetOrCreateCategory()`.

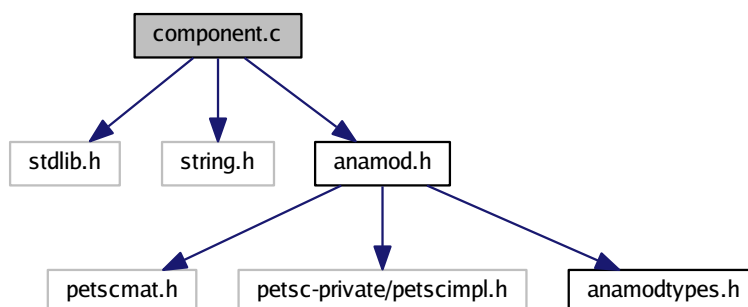
17.10.3.5 `int ncategories`

Definition at line 278 of file `module_functions.c`.

Referenced by `AllocCategoryObjects()`, `CategoryEnableByName()`, `FreeCategoryObjects()`, `GetCategories()`, `GetCategoryIndex()`, `GetFirstCategory()`, `GetNextCategory()`, and `GetOrCreateCategory()`.

17.11 component.c File Reference

```
#include <stdlib.h> #include <string.h> #include "anamod.h"
h" Include dependency graph for component.c:
```



Data Structures

- struct [componentobject_](#)

Defines

- #define [COMPONENTCOOKIE](#) 4723847
- #define [CHECKVALIDCOMPONENT](#)(x) [ANAMODCHECKVALID](#)(x,COMPONENTCOOKIE,"component")

Functions

- PetscErrorCode [CreateComponentObject](#) (const char *name, componentobject *obj)
- PetscErrorCode [DestroyComponentObject](#) (componentobject cmp)
- PetscErrorCode [ComponentSetModule](#) (componentobject cmpt, AnalysisDataType type, int id, PetscErrorCode(*f)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *))
- PetscErrorCode [ComponentCompute](#) (componentobject cmpt, AnaModNumericalProblem prob, AnalysisItem *res, int *reslen, PetscBool *success)

- PetscErrorCode [ComponentRetrieve](#) ([componentobject](#) cmpt, [AnaModNumericalProblem](#) prob, [AnalysisItem](#) *res, int *reslen, PetscBool *success)
- PetscErrorCode [ComponentGetType](#) ([componentobject](#) cmpt, [AnalysisDataType](#) *t)
- PetscErrorCode [ComponentGetName](#) ([componentobject](#) cmpt, const char **name)
- PetscErrorCode [ComponentGetId](#) ([componentobject](#) cmpt, int *id)

17.11.1 Define Documentation

17.11.1.1 **#define CHECKVALIDCOMPONENT(x) ANAMODCHECKVALID(x,COMPONENTCOOKIE,"component")**

Definition at line 6 of file component.c.

Referenced by [ComponentCompute\(\)](#), [ComponentGetId\(\)](#), [ComponentGetName\(\)](#), [ComponentGetType\(\)](#), [ComponentRetrieve\(\)](#), [ComponentSetModule\(\)](#), and [DestroyComponentObject\(\)](#).

17.11.1.2 **#define COMPONENTCOOKIE 4723847**

Definition at line 5 of file component.c.

Referenced by [CreateComponentObject\(\)](#).

17.11.2 Function Documentation

17.11.2.1 **PetscErrorCode ComponentCompute ([componentobject](#) cmpt, [AnaModNumericalProblem](#) prob, [AnalysisItem](#) * res, int * reslen, PetscBool * success)**

Definition at line 57 of file component.c.

References [CHECKVALIDCOMPONENT](#), [componentobject_::hasval](#), and [componentobject_::module](#).

Referenced by [ComputeOrRetrieveQuantity\(\)](#).

17.11.2.2 **PetscErrorCode ComponentGetId ([componentobject](#) cmpt, int * id)**

Definition at line 139 of file component.c.

References [CHECKVALIDCOMPONENT](#), and [componentobject_::dataid](#).

Referenced by [GetDataID\(\)](#), and [HasQuantity\(\)](#).

**17.11.2.3 PetscErrorCode ComponentGetName (componentobject *cmpt*, const char
** *name*)**

Definition at line 129 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::name.

Referenced by CategoryGetOrCreateComponent().

**17.11.2.4 PetscErrorCode ComponentGetType (componentobject *cmpt*,
AnalysisDataType * *t*)**

Definition at line 119 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::type.

Referenced by ComputeOrRetrieveQuantity(), GetDataType(), and HasQuantity().

**17.11.2.5 PetscErrorCode ComponentRetrieve (componentobject *cmpt*,
AnaModNumericalProblem *prob*, AnalysisItem * *res*, int * *reslen*, PetscBool
* *success*)**

Definition at line 75 of file component.c.

References ANALYSISDBLARRAY, ANALYSISDOUBLE, ANALYSISINTARRAY, ANALYSISINTEGER, CHECKVALIDCOMPONENT, componentobject_::dataid, AnalysisItem::i, AnalysisItem::ii, AnalysisItem::r, AnalysisItem::rr, and componentobject_::type.

Referenced by ComputeOrRetrieveQuantity().

**17.11.2.6 PetscErrorCode ComponentSetModule (componentobject *cmpt*, Analysis-
DataType *type*, int *id*, PetscErrorCode*)(AnaModNumericalProblem,
AnalysisItem *, int *, PetscBool *) *f*)**

Definition at line 43 of file component.c.

References CHECKVALIDCOMPONENT, componentobject_::dataid, *id*, componentobject_::module, and componentobject_::type.

Referenced by CategoryComponentSetModule().

**17.11.2.7 PetscErrorCode CreateComponentObject (const char * *name*,
componentobject * *obj*)**

Definition at line 17 of file component.c.

References COMPONENTCOOKIE, componentobject_::cookie, and componentobject_::name.

Referenced by CategoryGetOrCreateComponent().

17.11.2.8 PetscErrorCode DestroyComponentObject (componentobject *cmp*)

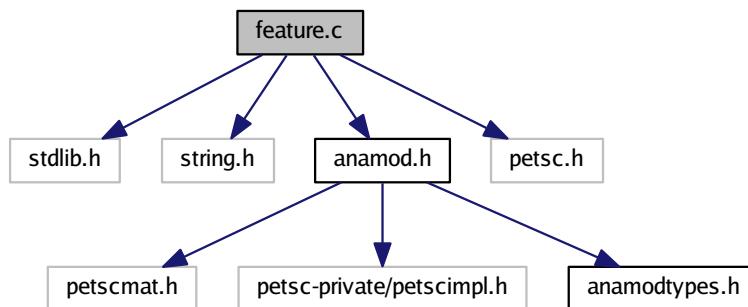
Definition at line 30 of file component.c.

References CHECKVALIDCOMPONENT, and componentobject_::name.

Referenced by DestroyCategoryObject().

17.12 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, const char *cat, const 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, PetscBool *f)
- PetscErrorCode [AnaModSetRetrievalFunction](#) (PetscErrorCode(*fun)(void *, char *, char *, [AnalysisItem](#) *, [AnalysisDataType](#) *, PetscBool *))
- PetscErrorCode [AnaModGetRetrievalFunction](#) (PetscErrorCode(*fun)(void *, char *, char *, [AnalysisItem](#) *, [AnalysisDataType](#) *, PetscBool *), PetscBool *flg)
- PetscErrorCode [AnaModCheckValidFeatureSet](#) (void *f)

Variables

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

17.12.1 Define Documentation

17.12.1.1 **#define CHECKVALIDFSET(i) {ANAMODCHECKVALID(i,FSETCOOKIE,"feature set");}**

Definition at line 59 of file feature.c.

Referenced by [AddToFeatureSet\(\)](#), [AnaModCheckValidFeatureSet\(\)](#), [DeleteFeatureSet\(\)](#), and [InstantiateFeatureSet\(\)](#).

17.12.1.2 **#define CHECKVALIDFVAL(i) {ANAMODCHECKVALID(i,FVALCOOKIE,"feature values");}**

Definition at line 65 of file feature.c.

Referenced by [DeleteFeatureValues\(\)](#), [GetFeatureValue\(\)](#), and [InstantiateFeatureSet\(\)](#).

17.12.1.3 **#define FSETCOOKIE 9876**

Definition at line 58 of file feature.c.

Referenced by [NewFeatureSet\(\)](#).

17.12.1.4 #define FVALCOOKIE 9877

Definition at line 64 of file feature.c.

Referenced by NewFeatureValues().

17.12.1.5 #define NALLOC 25

Definition at line 57 of file feature.c.

17.12.2 Function Documentation

17.12.2.1 PetscErrorCode AddToFeatureSet (FeatureSet set, const char * cat, const 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 115 of file feature.c.

References CHECKVALIDFSET, name, and StringArrayAdd().

Here is the call graph for this function:



17.12.2.2 PetscErrorCode AnaModCheckValidFeatureSet (void * f)

Definition at line 242 of file feature.c.

References CHECKVALIDFSET.

17.12.2.3 `PetscErrorCode AnaModGetRetrievalFunction (PetscErrorCode(*)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscBool *) fun, PetscBool * flag)`

Definition at line 230 of file feature.c.

References `retriever`.

Referenced by `InstantiateFeatureSet()`.

17.12.2.4 `PetscErrorCode AnaModSetRetrievalFunction (PetscErrorCode(*)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscBool *) fun)`

Definition at line 220 of file feature.c.

References `retriever`.

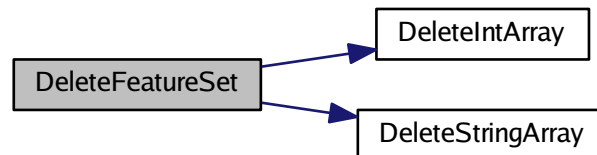
17.12.2.5 `PetscErrorCode DeleteFeatureSet (FeatureSet set)`

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:



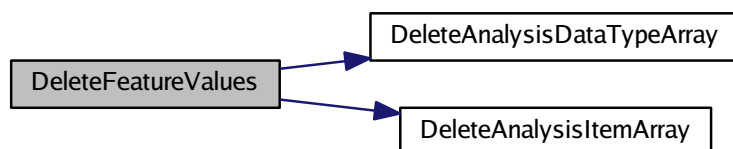
17.12.2.6 `PetscErrorCode DeleteFeatureValues (FeatureValues values)`

Free a featurevalues object. See [Feature sets](#)

Definition at line 149 of file feature.c.

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

Here is the call graph for this function:



17.12.2.7 `PetscErrorCode GetFeatureValue (FeatureValues values, int index, AnalysisItem * val, PetscBool * 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 207 of file `feature.c`.

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

Here is the call graph for this function:



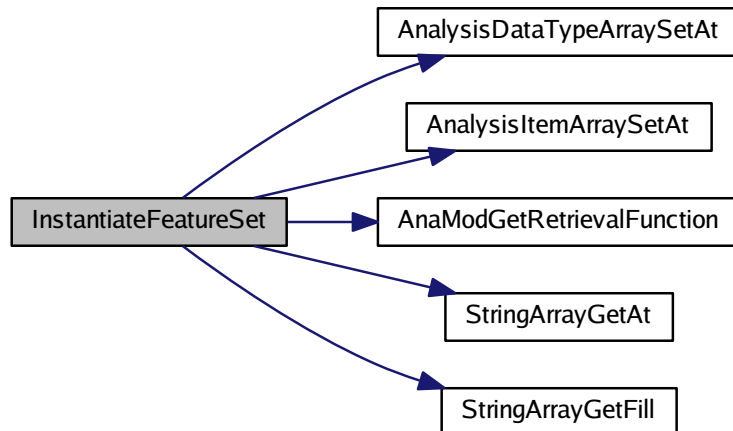
**17.12.2.8 PetscErrorCode InstantiateFeatureSet (AnaModNumericalProblem *prob*,
FeatureSet *set*, FeatureValues *values*)**

Fill in a featurevalues object. See [Feature sets](#)

Definition at line 164 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:

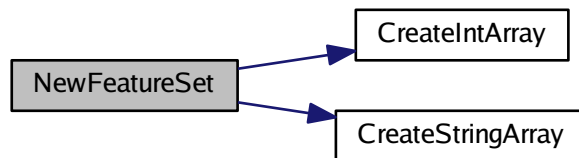
**17.12.2.9 PetscErrorCode NewFeatureSet (FeatureSet * *set*)**

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:



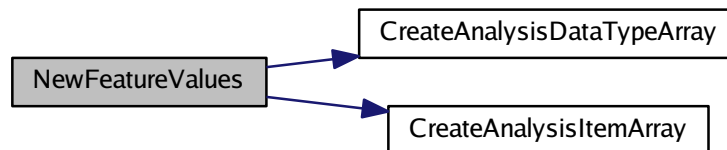
17.12.2.10 PetscErrorCode NewFeatureValues (FeatureValues * values)

Allocate a featurevalues object. See [Feature sets](#)

Definition at line 133 of file feature.c.

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

Here is the call graph for this function:



17.12.3 Variable Documentation

17.12.3.1 PetscErrorCode(* retriever)(void *, char *, char *, AnalysisItem *, AnalysisDataType *, PetscBool *) = NULL [static]

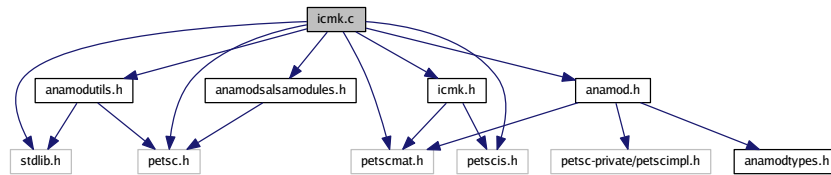
Definition at line 216 of file feature.c.

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

17.13 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 depen-
dency 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[jsp]+(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, PetscBool *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, PetscBool *flg)
- static PetscErrorCode [Splits](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, PetscBool *flg)
- PetscErrorCode [RegisterICMKModules](#) (void)

Variables

- double [mq](#)
- int [nc](#)
- int [ml](#)

17.13.1 Detailed Description

Functions for computing the ICMK structure of a matrix.

17.13.2 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.

17.13.3 Usage

Activate this module with:

PetscErrorCode [RegisterICMKModules\(\)](#);

Compute these elements with

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

Available elements are:

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

17.13.4 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](#).

17.13.5 Define Documentation

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

Referenced by CondenseChain().

17.13.5.2 #define CORNERUP 16

Referenced by MatSplitPoints().

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

Referenced by MatSplitPoints().

17.13.5.4 #define LEFTDOWN 1

Referenced by MatSplitPoints().

17.13.5.5 #define LEFTUP 4

Referenced by MatSplitPoints().

17.13.5.6 **#define OVERFLOW_DOWN** $i+tsd-1 \geq last$

Referenced by MatSplitPoints().

17.13.5.7 **#define OVERFLOW_UP** $i-ts < first$

Referenced by MatSplitPoints().

17.13.5.8 **#define RIGHTDOWN** 2

Referenced by MatSplitPoints().

17.13.5.9 **#define RIGHTUP** 8

Referenced by MatSplitPoints().

17.13.5.10 **#define SET_J** $j = Split_array[jsplit];$

Referenced by MatSplitPoints().

17.13.5.11 **#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().

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

Referenced by MatSplitPoints().

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

Referenced by MatSplitPoints().

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

Referenced by MatSplitPoints().

17.13.5.15 **#define SPLIT_BLOCK** 3

Definition at line 145 of file icmk.c.

Referenced by CanChain(), and MatSplitPoints().

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

Referenced by MatSplitPoints().

17.13.5.17 `#define VERY_FIRST_ROW (isFirst && isplit==0)`

Definition at line 253 of file icmk.c.

Referenced by MatSplitPoints().

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

Definition at line 254 of file icmk.c.

Referenced by MatSplitPoints().

17.13.6 Function Documentation

17.13.6.1 `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:



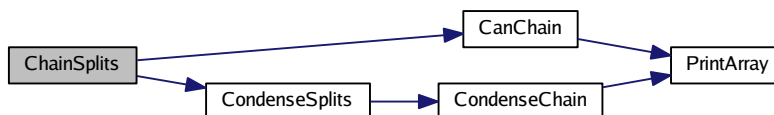
17.13.6.2 `static PetscErrorCode ChainSplits (int N, int * splits, int s_top, int p, IS * rar) [static]`

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:



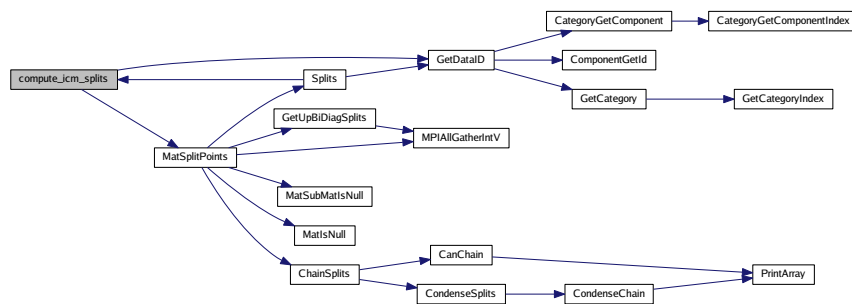
17.13.6.3 static PetscErrorCode compute_icm_splits (Mat A) [static]

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:



17.13.6.4 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:



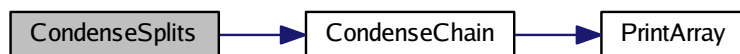
17.13.6.5 int CondenseSplits (int *p*, IS * *splits*)

Definition at line 216 of file icmk.c.

References CondenseChain().

Referenced by ChainSplits().

Here is the call graph for this function:



17.13.6.6 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:



17.13.6.7 static void iSpaces (int *len*) [static]

Definition at line 50 of file icmk.c.

17.13.6.8 int MatIsNull (Mat *A*, int * *flag*)

Definition at line 57 of file icmk.c.

Referenced by MatSplitPoints().

17.13.6.9 int MatRedistribute (Mat * *A*, IS *splits*)

Definition at line 582 of file icmk.c.

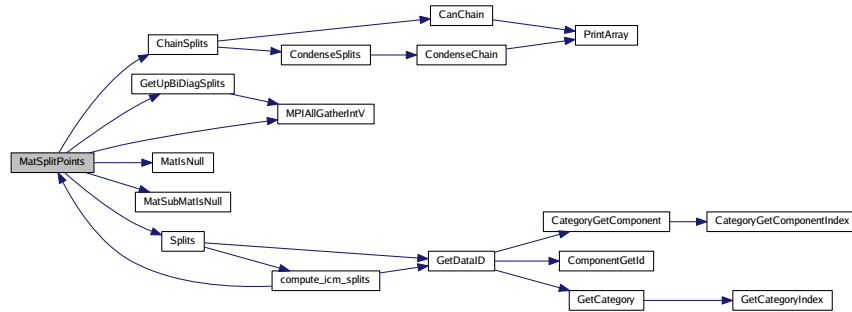
17.13.6.10 int MatSplitPoints (Mat *A*, int *np*, IS * *splitpoints*)

Definition at line 308 of file icmk.c.

References ChainSplits(), CORNERUP, GetUpBiDiagSplits(), IF_TOO_FAR_RIGHT_BREAK, LEFTDOWN, LEFTUP, MatIsNull(), MatSubMatIsNull(), 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:



17.13.6.11 int MatSubMatIsNull (Mat A, int i1, int i2, int j1, int j2, PetscBool * isnull)

Definition at line 78 of file icmk.c.

Referenced by MatSplitPoints().

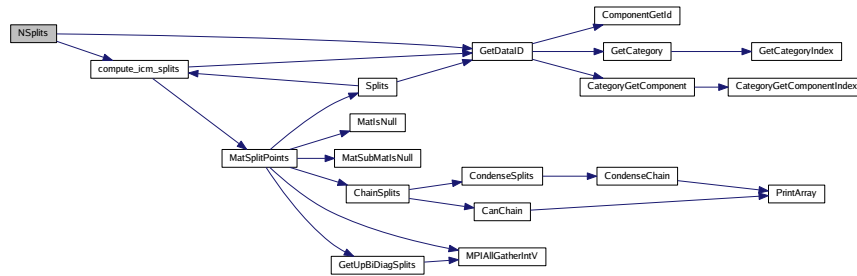
17.13.6.12 static PetscErrorCode NSplits (AnaModNumericalProblem prob, AnalysisItem * rv, PetscBool * flg) [static]

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:



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

Definition at line 99 of file icmk.c.

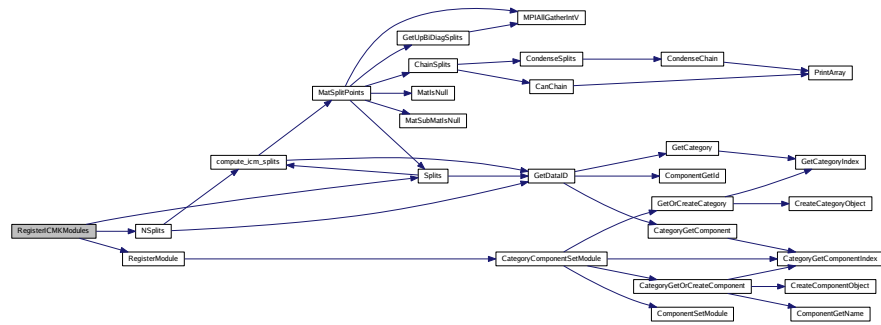
Referenced by CanChain(), and CondenseChain().

17.13.6.14 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:



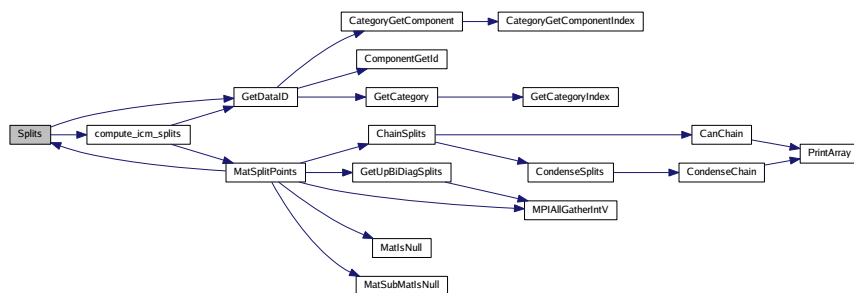
17.13.6.15 static PetscErrorCode Splits (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, PetscBool * *flg*) [static]

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:



17.13.6.16 `int VecRedistribute (Vec * V, IS splits)`

Definition at line 623 of file icmk.c.

17.13.7 Variable Documentation

17.13.7.1 int ml

Definition at line 144 of file icmk.c.

Referenced by ChainSplits().

17.13.7.2 double mq

Definition at line 144 of file icmk.c.

Referenced by ChainSplits().

17.13.7.3 int nc

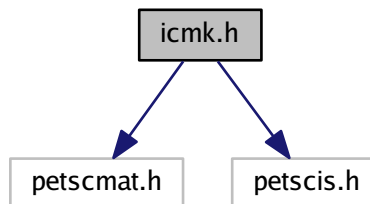
Definition at line 144 of file icmk.c.

Referenced by ChainSplits().

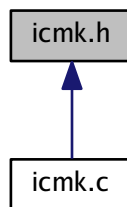
17.14 icmk.h File Reference

```
#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)

17.14.1 Function Documentation

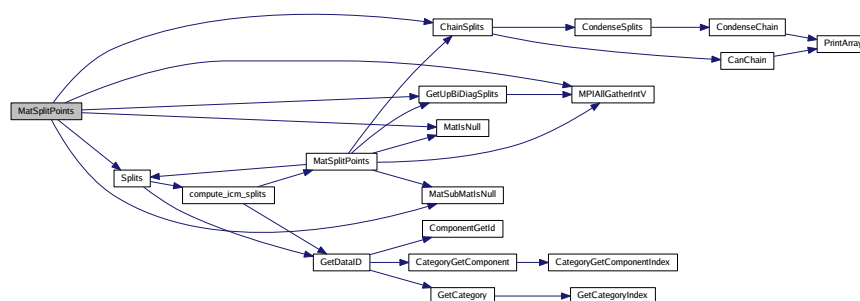
17.14.1.1 int MatSplitPoints (Mat A, int np, IS * *splitpoints*)

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:

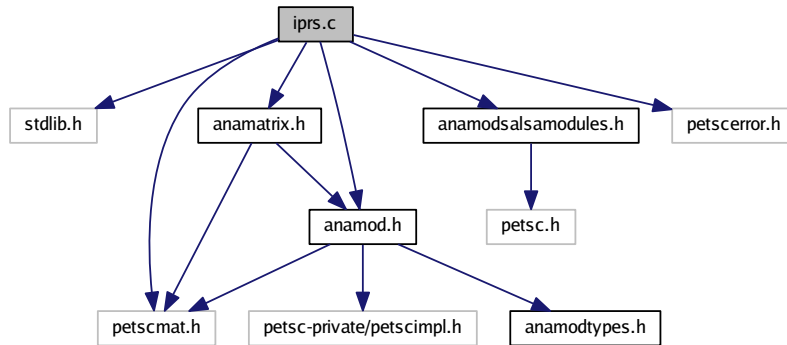


17.15 iprs.c File Reference

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

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

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, PetscBool *flg)
- static PetscErrorCode [NnzLow](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [NnzDia](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [Nnz](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [RelSymm](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [LoBand](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [UpBand](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [AvgNnzpRow](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [NDiags](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [AvgDiagDist](#) (AnaModNumericalProblem prob, AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SigmaDiagDist](#) (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)

- PetscErrorCode [RegisterIprsModules](#) (void)
- PetscErrorCode [DeRegisterIprsModules](#) (void)

17.15.1 Detailed Description

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

17.15.2 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.

17.15.3 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

17.15.4 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](#).

17.15.5 Function Documentation

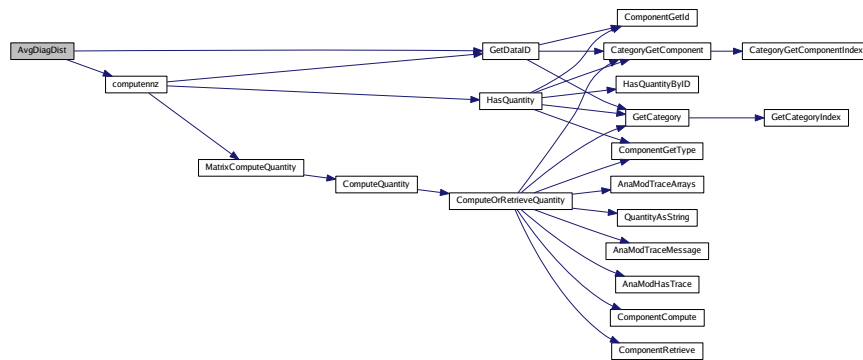
**17.15.5.1 static PetscErrorCode AvgDiagDist (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

Definition at line 488 of file iprs.c.

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

Referenced by `RegisterIprsModules()`.

Here is the call graph for this function:



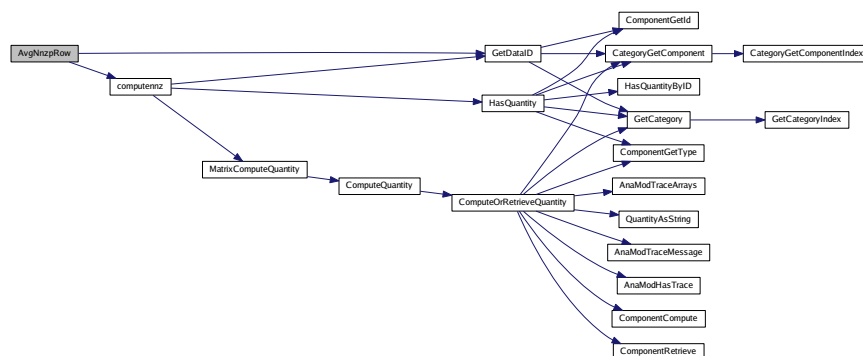
**17.15.5.2 static PetscErrorCode AvgNnzpRow (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

Definition at line 414 of file iprs.c.

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

Referenced by RegisterIprsModules().

Here is the call graph for this function:



17.15.5.3 static PetscErrorCode computennz (Mat A) [static]

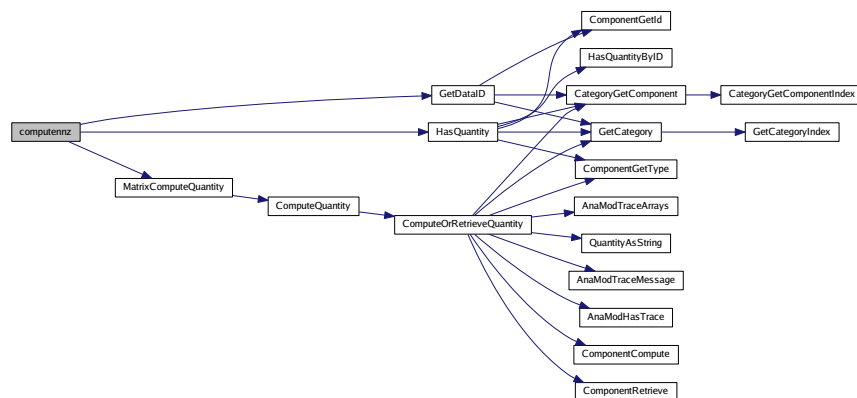
This is the computational routine for all IPRS modules. Lots of pointer chasing.

Definition at line 61 of file iprs.c.

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

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

Here is the call graph for this function:



17.15.5.4 PetscErrorCode DeRegisterIprsModules (void)

Definition at line 574 of file iprs.c.

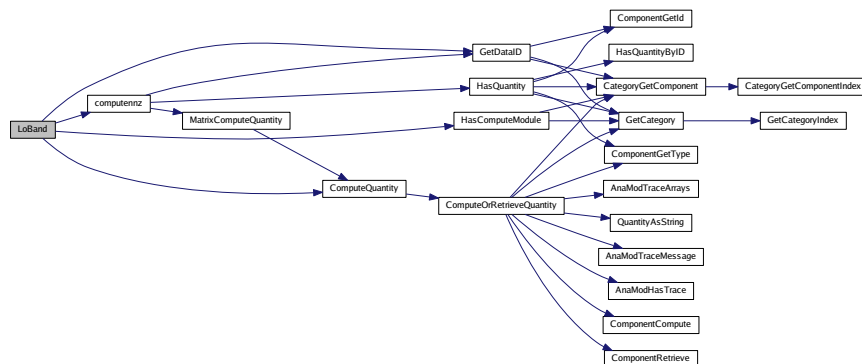
17.15.5.5 static PetscErrorCode LoBand (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 340 of file iprs.c.

References `computennz()`, `ComputeQuantity()`, `GetDataID()`, `HasComputeModule()`, `H-ASTOEXIST`, `AnalysisItem::i`, and `id`.

Referenced by `RegisterIprsModules()`.

Here is the call graph for this function:



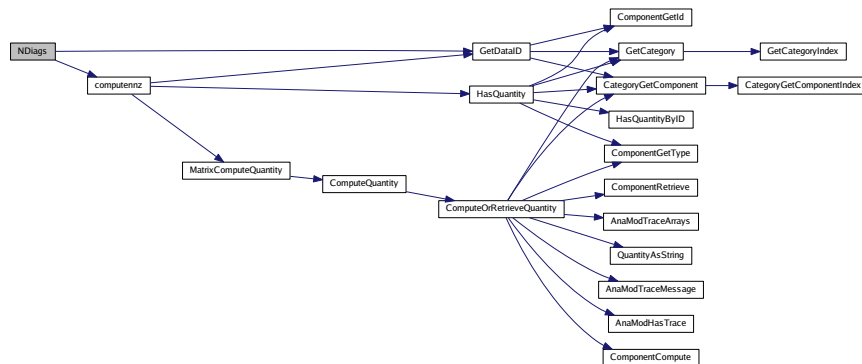
**17.15.5.6 static PetscErrorCode NDiags (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

Definition at line 464 of file iprs.c.

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

Referenced by `RegisterIprsModules()`.

Here is the call graph for this function:



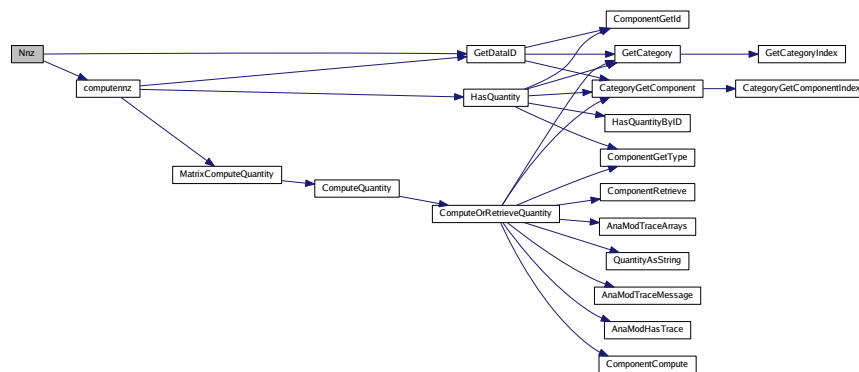
17.15.5.7 `static PetscErrorCode Nnz (AnaModNumericalProblem prob, AnalysisItem *
rv, int * lv, PetscBool * flag) [static]`

Definition at line 282 of file iprs.c.

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

Referenced by `RegisterIprsModules()`.

Here is the call graph for this function:



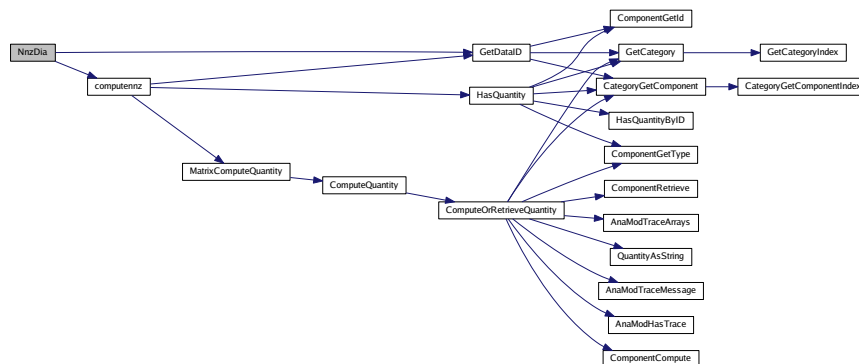
17.15.5.8 `static PetscErrorCode NnzDia (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flag) [static]`

Definition at line 258 of file iprs.c.

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

Referenced by `RegisterIprsModules()`.

Here is the call graph for this function:



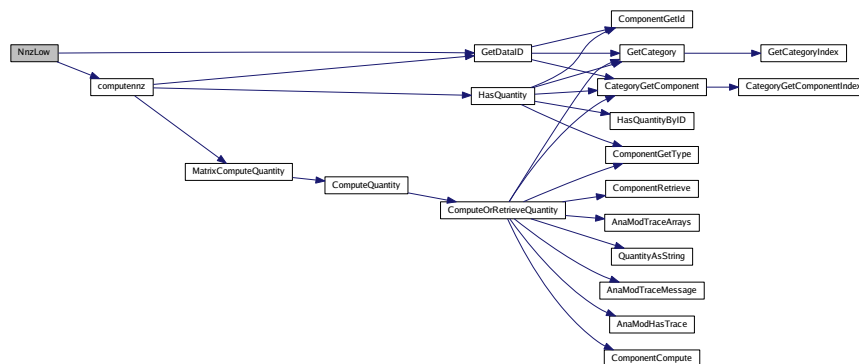
17.15.5.9 static PetscErrorCode NnzLow (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 234 of file iprs.c.

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

Referenced by RegisterIprsModules().

Here is the call graph for this function:



**17.15.5.10 static PetscErrorCode NnzUp (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

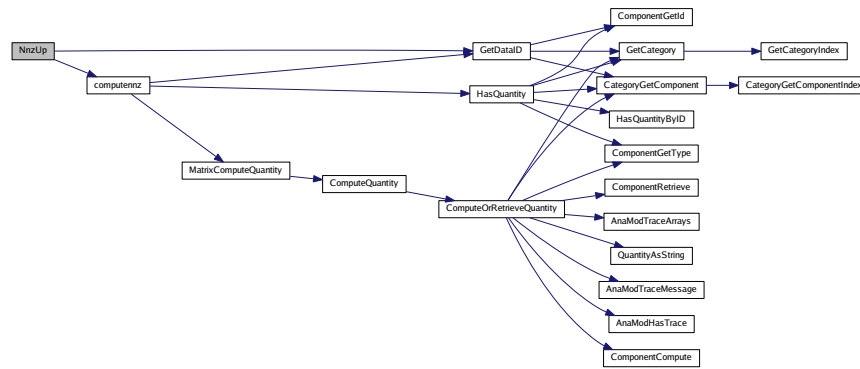
Compute the number of nonzeroes in the upper triangle of the matrix

Definition at line 210 of file iprs.c.

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

Referenced by RegisterIprsModules().

Here is the call graph for this function:



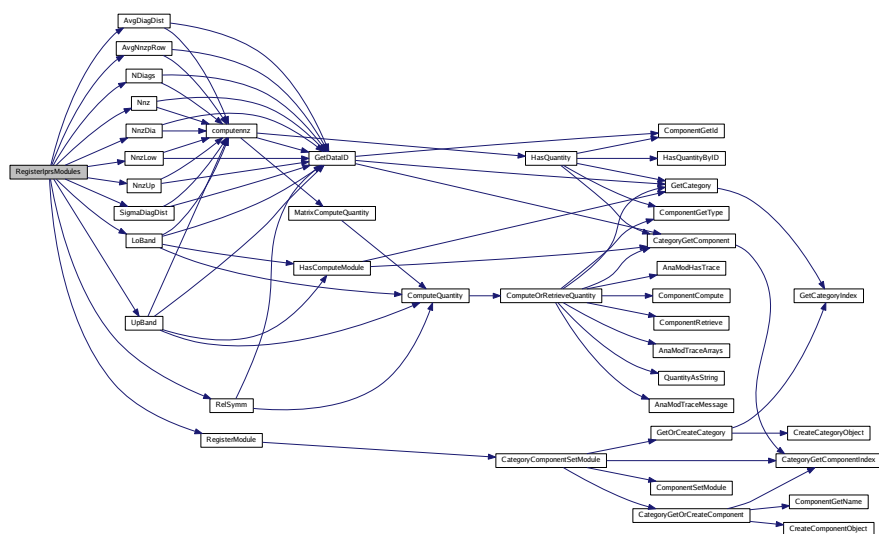
17.15.5.11 PetscErrorCode RegisterIprsModules (void)

Definition at line 535 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:



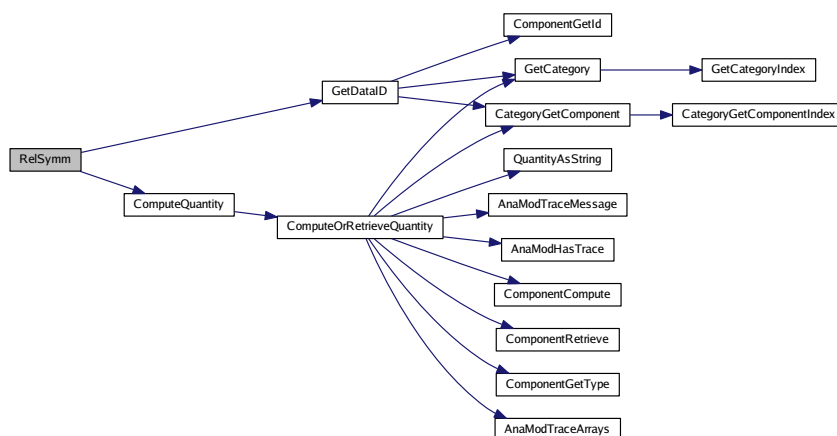
17.15.5.12 static PetscErrorCode RelSymm (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 306 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:



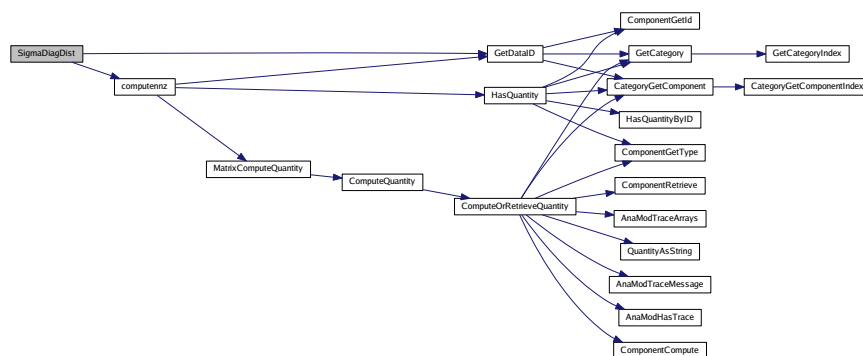
17.15.5.13 static PetscErrorCode SigmaDiagDist (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 512 of file iprs.c.

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

Referenced by RegisterIprsModules().

Here is the call graph for this function:



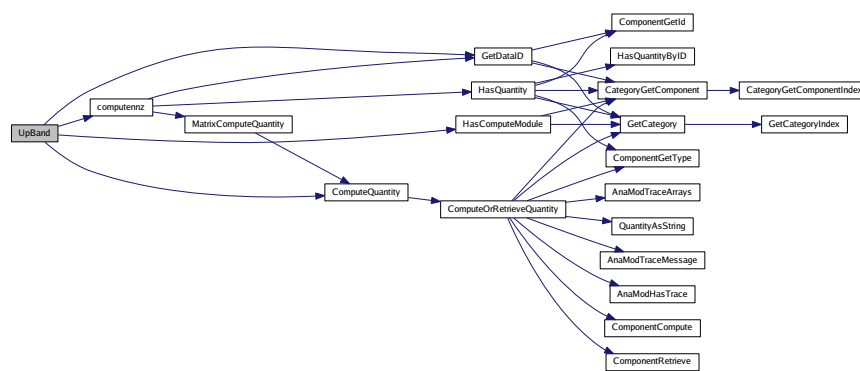
17.15.5.14 `static PetscErrorCode UpBand (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

Definition at line 377 of file iprs.c.

References `computennz()`, `ComputeQuantity()`, `GetDataID()`, `HasComputeModule()`, `H-ASTOEXIST`, `AnalysisItem::i`, and `id`.

Referenced by `RegisterIprsModules()`.

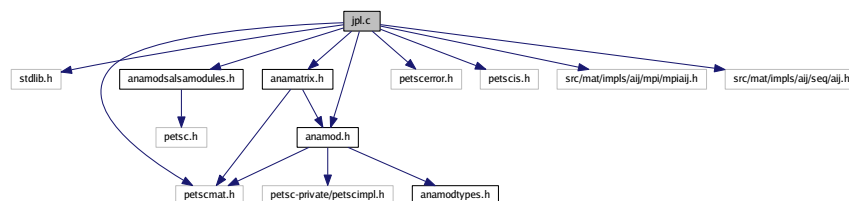
Here is the call graph for this function:



17.16 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/mpi-
aij.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, PetscBool *colour_now)
- static PetscErrorCode [JonesPlassmannColouring](#) (Mat A, PetscBool *flg)
- static PetscErrorCode [NColours](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [ColourSizes](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [ColourOffsets](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [Colours](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterJPLModules](#) (void)

17.16.1 Detailed Description

Functions for computing the JPL multicolour structure of a matrix.

17.16.2 Jones-Plassmann multi-colouring

17.16.3 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.

17.16.4 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 unstructured, 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](#).

17.16.5 Function Documentation

17.16.5.1 static PetscErrorCode ColourOffsets (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

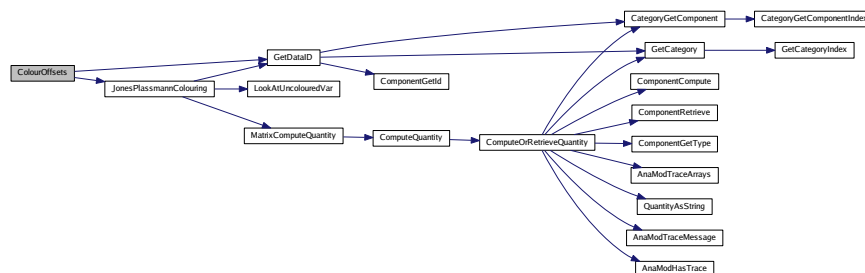
Compute an array that has the offsets of the locations of the colours, on this processor. If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

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:



17.16.5.2 static PetscErrorCode Colours (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * 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.

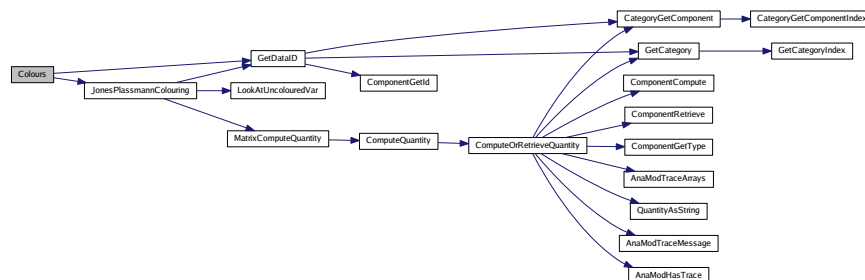
If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

Definition at line 497 of file jpl.c.

References [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), [AnalysisItem::ii](#), and [JonesPlassmannColouring\(\)](#).

Referenced by [RegisterJPLModules\(\)](#).

Here is the call graph for this function:



17.16.5.3 `static PetscErrorCode ColourSizes (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

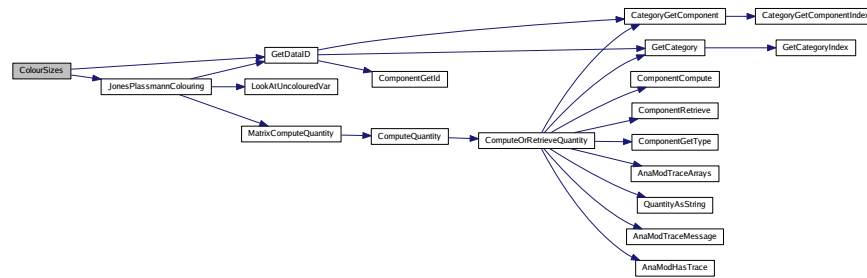
Compute an array that has the number of points of each colour, on this processor. If computing the colouring is done in parallel, then the multicolouring is stored in parallel: it is not gathered onto any processor.

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:



17.16.5.4 `static PetscErrorCode JonesPlassmannColouring (Mat A, PetscBool * flg)
[static]`

Compute a multicolour ordering of a matrix, in parallel.

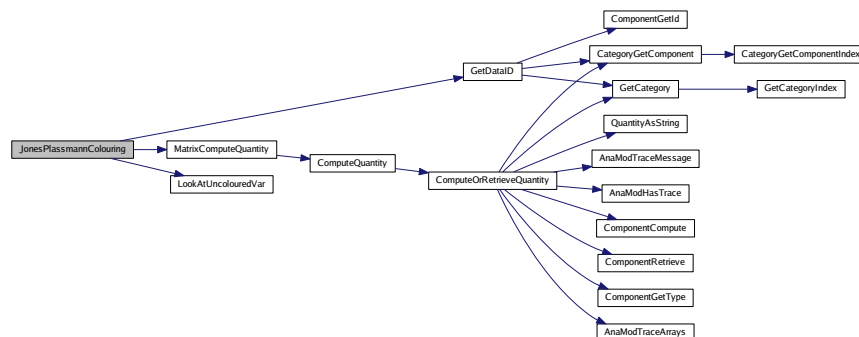
This is the main computational routine.

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:



17.16.5.5 static PetscErrorCode LookAtUncolouredVar (int *compare*, int *this_var*, Mat *A*, PetscScalar * *clr_array*, PetscScalar * *ran_array*, PetscReal *this_rand*, int * *neighb*, int * *nneigh*, PetscBool * *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().

17.16.5.6 static PetscErrorCode NColours (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

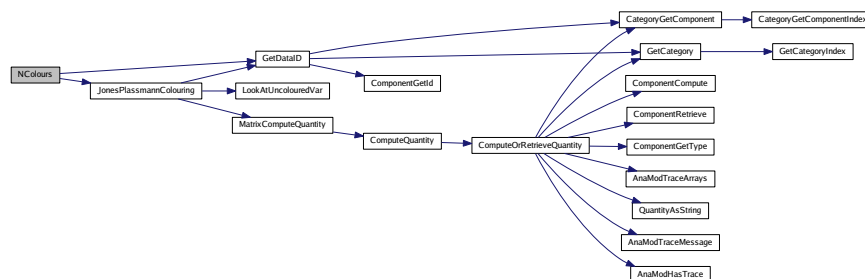
Compute the global number of colours. Note that any given processor does not need to have all the colours.

Definition at line 377 of file jpl.c.

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

Referenced by RegisterJPLModules().

Here is the call graph for this function:



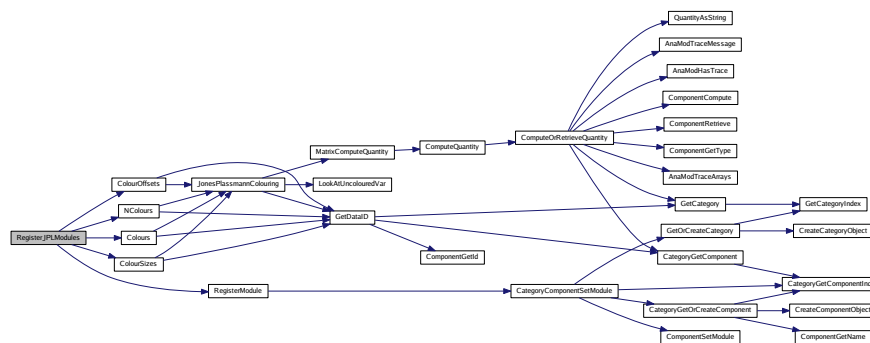
17.16.5.7 PetscErrorCode RegisterJPLModules (void)

Definition at line 523 of file jpl.c.

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

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

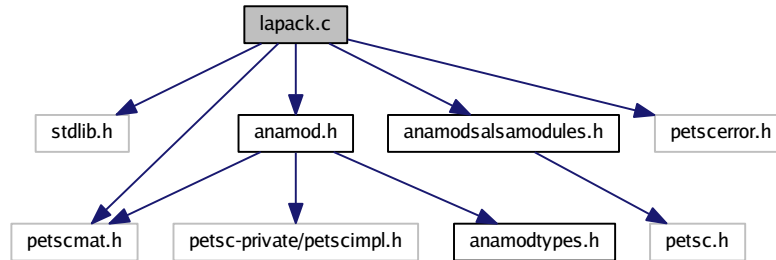
Here is the call graph for this function:



17.17 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 depen-
dency 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 *lv, PetscBool *flg)
- static PetscErrorCode `MaxEVbyMagRe` (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode `MaxEVbyMagIm` (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode `MinEVbyMagRe` (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode `MinEVbyMagIm` (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)
- static PetscErrorCode `MaxEVbyRealRe` (AnaModNumericalProblem prob, - AnalysisItem *rv, int *lv, PetscBool *flg)

- static PetscErrorCode [MaxEVbyReallm](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbyImRe](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbyImIm](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterLapackModules](#) ()

17.17.1 Detailed Description

Dense routines from Lapack: eigenvalue and schur stuff.

17.17.2 Dense calculations from Lapack

17.17.3 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](#).

17.17.4 Define Documentation

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

Referenced by [eigenvaluecomp](#)().

17.17.4.2 #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()`.

17.17.5 Function Documentation

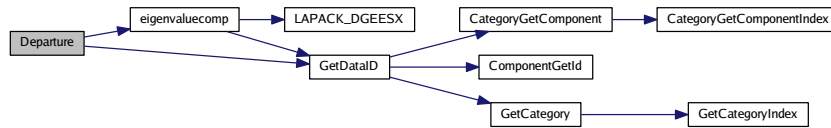
**17.17.5.1 static PetscErrorCode Departure (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

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:



17.17.5.2 static PetscErrorCode eigenvaluecomp (Mat *A*) [static]

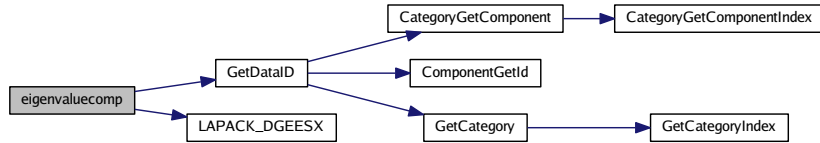
This is the computational routine for lapack eigenvalue calculation.

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:



17.17.5.3 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().

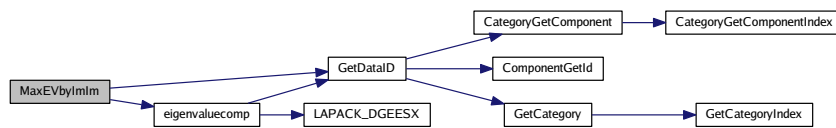
17.17.5.4 static PetscErrorCode **MaxEVbyImIm** (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



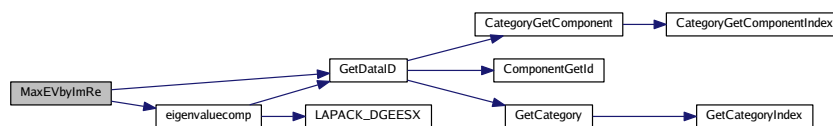
17.17.5.5 static PetscErrorCode **MaxEVbyImRe** (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



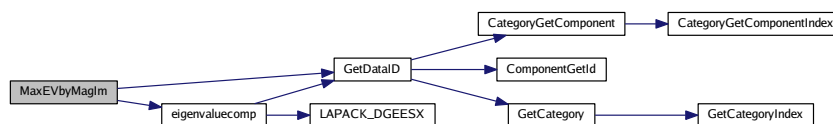
17.17.5.6 static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



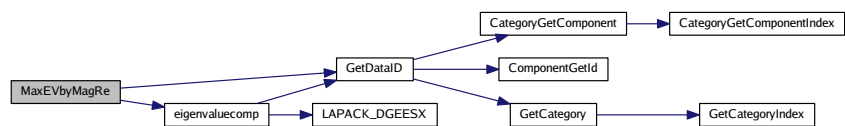
17.17.5.7 static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



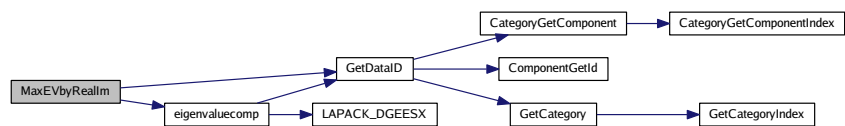
17.17.5.8 static PetscErrorCode MaxEVbyReallm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



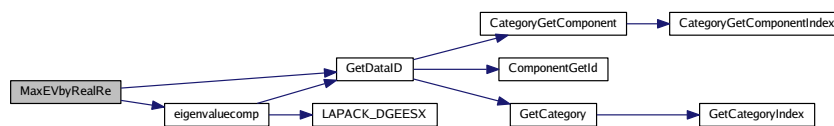
17.17.5.9 static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



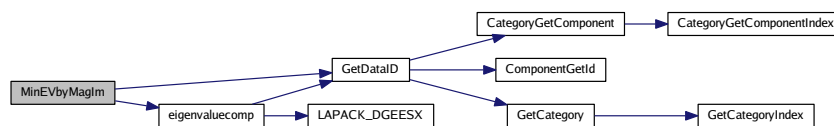
17.17.5.10 static PetscErrorCode MinEVbyMagIm (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



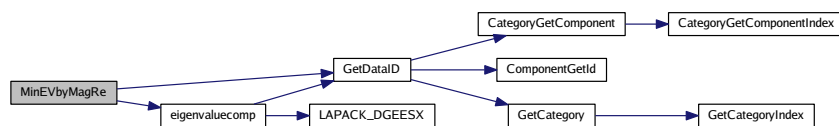
17.17.5.11 static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



17.17.5.12 PetscErrorCode RegisterLapackModules ()

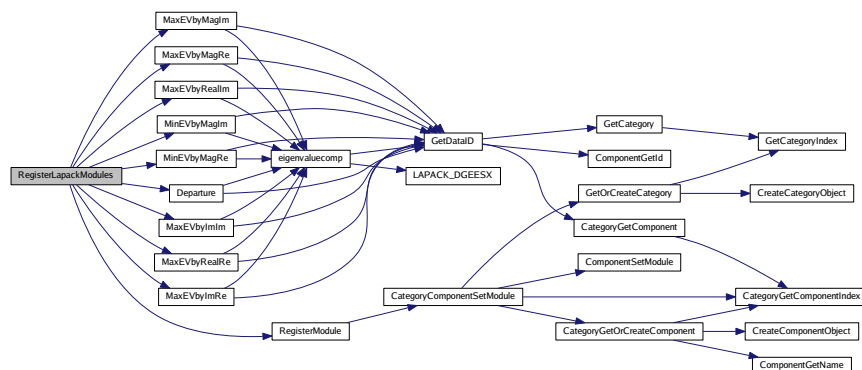
Declare norm-like modules that can be performed with lapack calculations.

Definition at line 424 of file lapack.c.

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

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

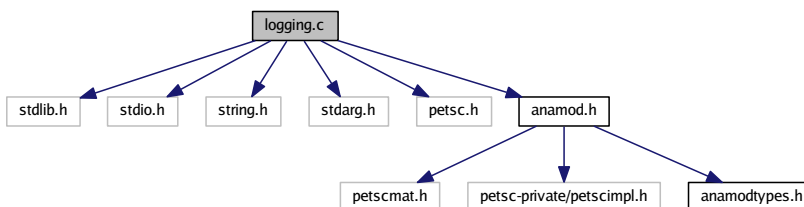
Here is the call graph for this function:



17.18 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)

17.18.1 Detailed Description

PETSc event logging in AnaMod.

17.18.2 Event logging

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

Definition in file [logging.c](#).

17.18.3 Function Documentation

17.18.3.1 PetscErrorCode CategoryLogEventRegister (char * cat, int icat)

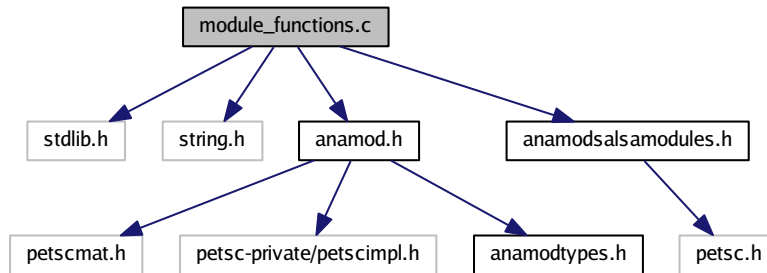
Definition at line 20 of file logging.c.

17.19 Make.inc File Reference

17.20 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 dependency graph for
module_functions.c:
```



Defines

- `#define` [MODE_RETRIEVE](#) 1
- `#define` [MODE_COMPUTE](#) 0

Functions

- `PetscErrorCode` [AnaModInitialize](#) ()
- `PetscErrorCode` [AnaModFinalize](#) ()
- `PetscErrorCode` [AnaModGetTypeNames](#) (int `id`, const char **`name`)
- `PetscErrorCode` [AnaModGetTypeMySQLName](#) (int `id`, const char **`name`)
- `PetscErrorCode` [RegisterModule](#) (const char *`cat`, const char *`cmp`, [AnalysisDataType](#) type, `PetscErrorCode`(*f)([AnaModNumericalProblem](#), [AnalysisItem](#) *, int *, `PetscBool` *))
- `PetscErrorCode` [DeRegisterCategory](#) (const char *`cat`)
- `PetscErrorCode` [DeregisterModules](#) ()
- `PetscErrorCode` [HasComputeCategory](#) (const char *`cat`, `PetscBool` *f)
- `PetscErrorCode` [HasComputeModule](#) (const char *`cat`, const char *`cmp`, `PetscBool` *f)
- static `PetscErrorCode` [ComputeOrRetrieveQuantity](#) ([AnaModNumericalProblem](#) prob, const char *`cat`, const char *`cmp`, [AnalysisItem](#) *`res`, int *`rreslen`, `PetscBool` *`success`, int `mode`)
- `PetscErrorCode` [ComputeQuantity](#) ([AnaModNumericalProblem](#) prob, const char *`cat`, const char *`cmp`, [AnalysisItem](#) *`res`, int *`rreslen`, `PetscBool` *`success`)

- PetscErrorCode [RetrieveQuantity](#) ([AnaModNumericalProblem](#) prob, const char *cat, const char *cmp, [AnalysisItem](#) *res, int *rreslen, PetscBool *success)
- PetscErrorCode [GetDataID](#) (const char *cat, const char *cmp, int *id, PetscBool *f)
- PetscErrorCode [GetDataType](#) (const char *cat, const char *cmp, [AnalysisDataType](#) *t, PetscBool *f)
- PetscErrorCode [HasQuantity](#) ([AnaModNumericalProblem](#) prob, const char *cat, const char *cmp, PetscBool *f)
- PetscErrorCode [HasQuantityByID](#) ([AnaModNumericalProblem](#) prob, int id, [AnalysisDataType](#) type, PetscBool *f)
- PetscErrorCode [RetrieveQuantityByID](#) ([AnaModNumericalProblem](#) prob, int id, [AnalysisDataType](#) type, [AnalysisItem](#) *result, PetscBool *f)
- PetscErrorCode [QuantityAsString](#) ([AnalysisItem](#) *q, [AnalysisDataType](#) t, char **s)

Variables

- struct {
 int id
 const char * name
 const char * mysqlname
} [anamodtypenames](#) [5]
- static int [nAnaModTypeNames](#)
- static int [AnaModIsInitialized](#) = 0
- int [ncategories](#) = 0
- [categoryobject](#) * [categoryobjects](#) = NULL

17.20.1 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](#).

17.20.2 Define Documentation

17.20.2.1 #define MODE_COMPUTE 0

Definition at line 477 of file module_functions.c.

Referenced by ComputeOrRetrieveQuantity(), and ComputeQuantity().

17.20.2.2 #define MODE_RETRIEVE 1

Definition at line 476 of file module_functions.c.

Referenced by ComputeOrRetrieveQuantity(), and RetrieveQuantity().

17.20.3 Function Documentation

17.20.3.1 PetscErrorCode AnaModFinalize ()

Finalization for AnaMod. See also [AnaModInitialize\(\)](#)

Definition at line 318 of file module_functions.c.

References FreeCategoryObjects().

Referenced by main().

Here is the call graph for this function:



17.20.3.2 PetscErrorCode AnaModGetTypeMySQLName (int id, const char ** name)

Definition at line 344 of file module_functions.c.

References AnaModIsInitialized, anamodtypenames, mysqlname, and nAnaModTypeNames.

Referenced by main().

17.20.3.3 PetscErrorCode AnaModGetTypeNames (int id, const char ** name)

Definition at line 328 of file module_functions.c.

References anamodtypenames, and nAnaModTypeNames.

17.20.3.4 PetscErrorCode AnaModInitialize ()

Initializatin for AnaMod. See also [AnaModFinalize\(\)](#)

Definition at line 284 of file module_functions.c.

References [AllocCategoryObjects\(\)](#), [AnaModIsInitialized](#), [anamodtypenames](#), and [nAnaModTypeNames](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



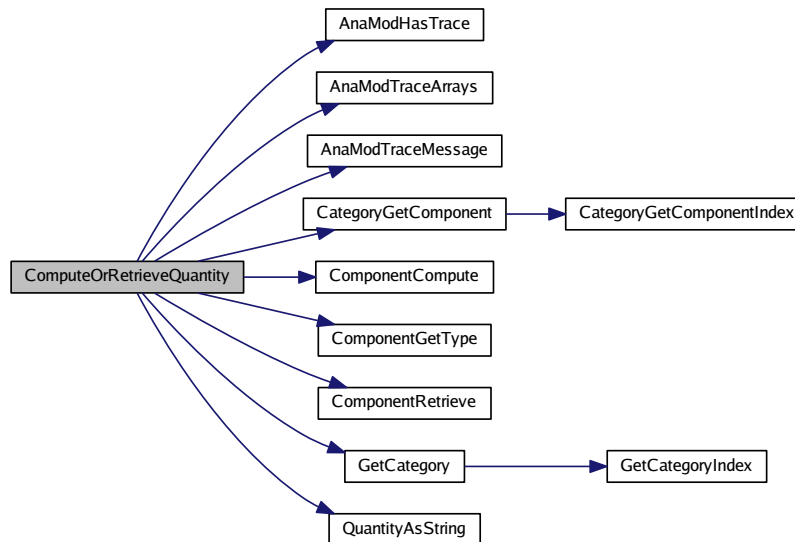
17.20.3.5 static PetscErrorCode ComputeOrRetrieveQuantity (
AnaModNumericalProblem *prob*, const char * *cat*, const char * *cmp*,
AnalysisItem * *res*, int * *reslen*, PetscBool * *success*, int *mode*) [static]

Definition at line 481 of file module_functions.c.

References [ANALYSISINTARRAY](#), [AnaModHasTrace\(\)](#), [AnaModTraceArrays\(\)](#), [AnaModTraceMessage\(\)](#), [CategoryGetComponent\(\)](#), [ComponentCompute\(\)](#), [ComponentGetType\(\)](#), [ComponentRetrieve\(\)](#), [GetCategory\(\)](#), [MODE_COMPUTE](#), [MODE_RETRIEVE](#), and [QuantityAsString\(\)](#).

Referenced by [ComputeQuantity\(\)](#), and [RetrieveQuantity\(\)](#).

Here is the call graph for this function:



17.20.3.6 `PetscErrorCode ComputeQuantity (AnaModNumericalProblem prob, const char * cat, const char * cmp, AnalysisItem * res, int * rreslen, PetscBool * 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.

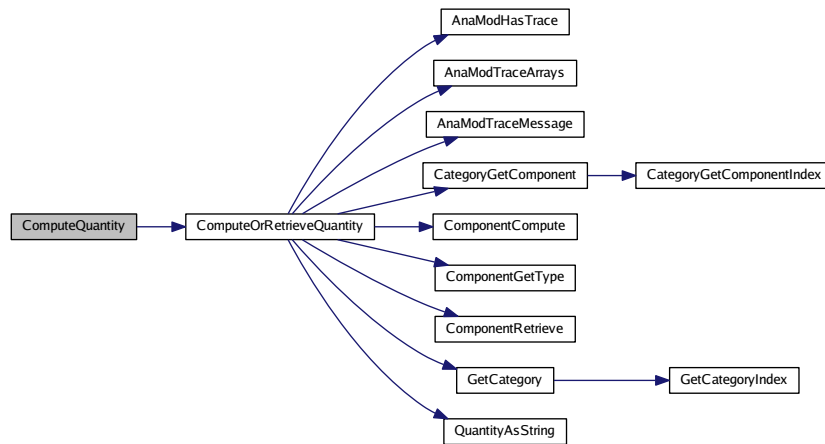
The workings of this function can be traced by specifying a trace function; see [Tracing the analysis modules](#).

Definition at line 558 of file module_functions.c.

References `AnaModIsInitialized`, `ComputeOrRetrieveQuantity()`, and `MODE_COMPUTE`.

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:



17.20.3.7 PetscErrorCode DeRegisterCategory (const char * cat)

Deallocate the storage for a particular category of analysis modules. No longer needed.

Definition at line 409 of file module_functions.c.

17.20.3.8 PetscErrorCode DeregisterModules (void)

This function is no longer needed

Definition at line 425 of file module_functions.c.

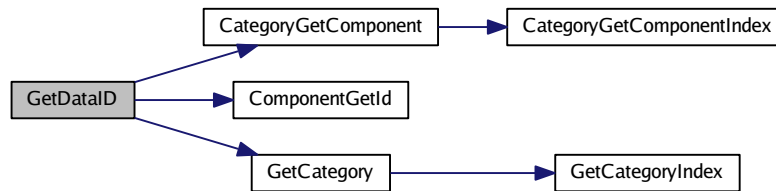
17.20.3.9 PetscErrorCode GetDataID (const char * *cat*, const char * *cmp*, int * *id*, PetscBool * *f*)

Definition at line 589 of file module_functions.c.

References CategoryGetComponent(), ComponentGetId(), and GetCategory().

Referenced by AvgDiagDist(), 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(), Kappa(), LBandWidth(), Lee95bounds(), Lee96bounds(), LeftSkyline(), LoBand(), MatCommutatorNormF(), MatSymmPartNormInf(), MaxEVbyImIm(), MaxEVbyImRe(), MaxEVbyMagIm(), MaxEVbyMagRe(), MaxEVbyReallm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUUnstruct(), 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:



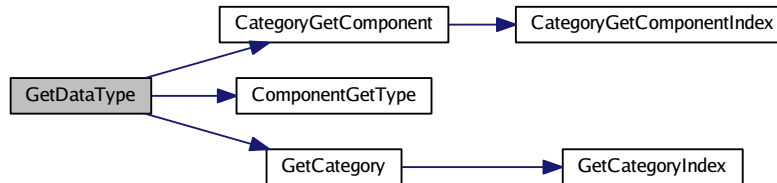
17.20.3.10 PetscErrorCode GetDataType (const char * *cat*, const char * *cmp*, AnalysisDataType * *t*, PetscBool * *f*)

Definition at line 612 of file module_functions.c.

References CategoryGetComponent(), ComponentGetType(), and GetCategory().

Referenced by analyze_matrix().

Here is the call graph for this function:



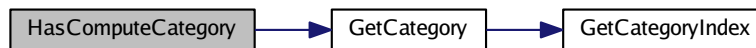
17.20.3.11 PetscErrorCode HasComputeCategory (const char * cat, PetscBool * f)

Query whether a specified category has been declared.

Definition at line 451 of file module_functions.c.

References GetCategory().

Here is the call graph for this function:



17.20.3.12 PetscErrorCode HasComputeModule (const char * cat, const char * cmp, PetscBool * f)

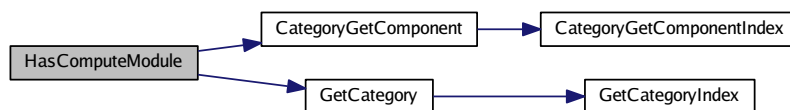
Query whether a specified module exists inside a specified category. The category need not itself have been declared.

Definition at line 465 of file module_functions.c.

References CategoryGetComponent(), and GetCategory().

Referenced by LoBand(), and UpBand().

Here is the call graph for this function:



17.20.3.13 `PetscErrorCode HasQuantity (AnaModNumericalProblem prob, const char * cat, const char * cmp, PetscBool * 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. Use [HasComputeModule\(\)](#) to test whether a category and module is known to the system.

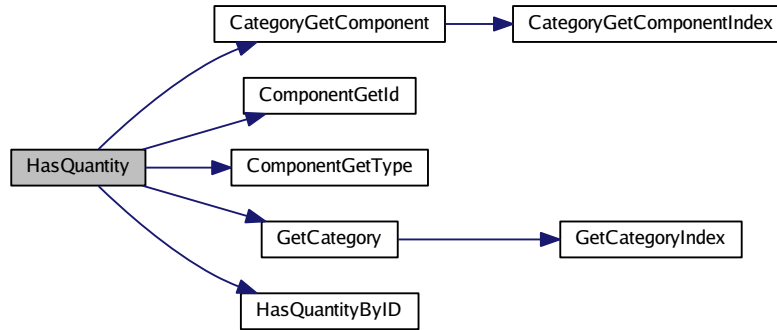
See [the page on attached quantities](#) for an explanation of the mechanism behind this routine.

Definition at line 646 of file `module_functions.c`.

References `CategoryGetComponent()`, `ComponentGetId()`, `ComponentGetType()`, `GetCategory()`, `HasQuantityById()`, and `id`.

Referenced by `computennz()`.

Here is the call graph for this function:



17.20.3.14 `PetscErrorCode HasQuantityById (AnaModNumericalProblem prob, int id, AnalysisDataType type, PetscBool * f)`

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

Definition at line 667 of file `module_functions.c`.

References `ANALYSISDBLARRAY`, `ANALYSISDOUBLE`, `ANALYSISINTARRAY`, and `ANALYSISINTEGER`.

Referenced by `HasQuantity()`.

17.20.3.15 `PetscErrorCode QuantityAsString (AnalysisItem * q, AnalysisDataType t, char ** s)`

Generate a character string for a given quantity.

Definition at line 731 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 `ComputeOrRetrieveQuantity()`, and `ReportAnamodContent()`.

17.20.3.16 `PetscErrorCode RegisterModule (const char * cat, const char * cmp, AnalysisDataType type, PetscErrorCode (*)(AnaModNumericalProblem, AnalysisItem *, int *, PetscBool *) 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)

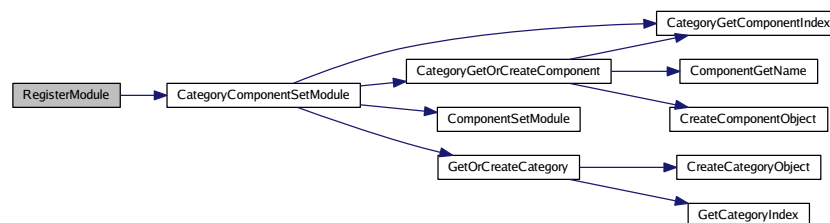
See also [HasComputeModule\(\)](#), [ComputeQuantity\(\)](#).

Definition at line 382 of file `module_functions.c`.

References `AnaModIsInitialized`, `CategoryComponentSetModule()`, and `id`.

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

Here is the call graph for this function:



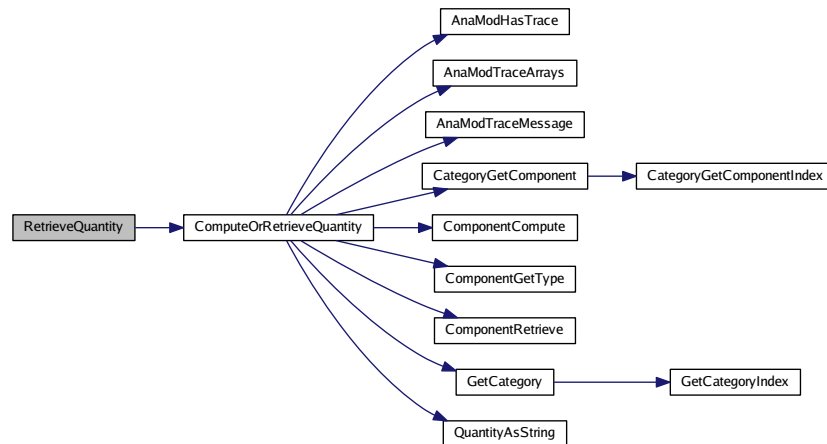
17.20.3.17 `PetscErrorCode RetrieveQuantity (AnaModNumericalProblem prob, const char * cat, const char * cmp, AnalysisItem * res, int * rreslen, PetscBool * success)`

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 577 of file module_functions.c.

References `ComputeOrRetrieveQuantity()`, and `MODE_RETRIEVE`.

Here is the call graph for this function:



17.20.3.18 `PetscErrorCode RetrieveQuantityByID (AnaModNumericalProblem prob, int id, AnalysisDataType type, AnalysisItem * result, PetscBool * f)`

See also [HasQuantityByID\(\)](#)

Definition at line 698 of file module_functions.c.

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

17.20.4 Variable Documentation

17.20.4.1 `int AnaModIsInitialized = 0` `[static]`

Definition at line 276 of file module_functions.c.

Referenced by `AnaModGetTypeMySQLName()`, `AnaModInitialize()`, `ComputeQuantity()`, and `RegisterModule()`.

17.20.4.2 struct { ... } anamodtypenames[5] [static]

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

17.20.4.3 categoryobject* categoryobjects = NULL

Definition at line 279 of file module_functions.c.

17.20.4.4 int id

Definition at line 274 of file module_functions.c.

Referenced by AvgDiagDist(), AvgNnzpRow(), BlockSize(), ColourOffsets(), Colours(), ColourSizes(), ColVariability(), Commutator(), ComponentSetModule(), 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(), MaxEVbyReallm(), MaxEVbyRealRe(), MaxNNonZerosPerRow(), MinEVbyMagIm(), MinEVbyMagRe(), MinNNonZerosPerRow(), NColours(), NDiags(), NDummyRows(), NNonZeros(), Nnz(), NnzDia(), NnzLow(), NnzUp(), norm1(), normF(), normInf(), NRitzValues(), nRows(), NSplits(), NUnstruct(), PosFraction(), RBandWidth(), RegisterModule(), regularblocks(), RelSymm(), RightSkyline(), RitzValuesC(), RitzValuesR(), RowVariability(), SigmaDiagDist(), SigmaMax(), SigmaMin(), SpectrumAX(), SpectrumAY(), SpectrumC-X(), SpectrumCY(), Splits(), Symmetry(), SymmetryANorm(), SymmetryFANorm(), SymmetryFSNorm(), SymmetrySNorm(), Trace(), TraceA2(), TraceAbs(), UpBand(), and Version().

17.20.4.5 const char * mysqlname

Definition at line 274 of file module_functions.c.

Referenced by AnaModGetTypeMySQLName().

17.20.4.6 const char* name

Definition at line 274 of file module_functions.c.

Referenced by AddToFeatureSet(), CategoryEnableByName(), and GetCategoryIndex().

17.20.4.7 int nAnaModTypeNames [static]

Definition at line 275 of file module_functions.c.

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

17.20.4.8 int ncategories = 0

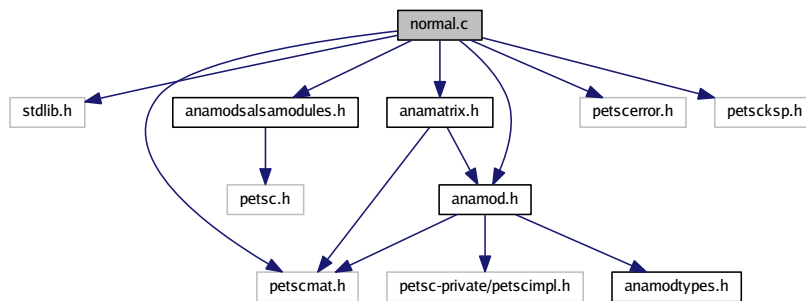
Definition at line 278 of file module_functions.c.

Referenced by AllocCategoryObjects(), CategoryEnableByName(), FreeCategoryObjects(), GetCategories(), GetCategoryIndex(), GetFirstCategory(), GetNextCategory(), and GetOrCreateCategory().

17.21 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, PetscBool *flg)
- static PetscErrorCode [compute_tracea2_seq](#) (Mat A, PetscScalar *trace)
- static PetscErrorCode [compute_tracea2](#) (Mat A, PetscBool *done)
- static PetscErrorCode [TraceA2](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [CommutatorNormFAllowSqrtTimes](#) (int n)
- static PetscErrorCode [MatCommutatorNormF_seq](#) (Mat A, PetscReal *result, - PetscBool *done)
- static PetscErrorCode [MatCommutatorNormF](#) (Mat A, PetscBool *done)
- static PetscErrorCode [Commutator](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [Lee96bounds](#) (Mat A)
- static PetscErrorCode [DepartureLee96U](#) (AnaModNumericalProblem prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DepartureLee96L](#) (AnaModNumericalProblem prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DepartureRuhe75](#) (AnaModNumericalProblem prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterNormalityModules](#) (void)

Variables

- static int [sqrt_times](#) = 50

17.21.1 Detailed Description

Various estimates of the departure from normality.

17.21.2 Estimates for the departure from normality

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

17.21.3 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 than 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 $AA^T - A^T A$
- "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]

17.21.4 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  $A^H A$  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](#).

17.21.5 Define Documentation

17.21.5.1 **#define** `max(a, b)` $(a > b ? a : b)$

Referenced by `MatCommutatorNormF_seq()`.

17.21.5.2 **#define** `min(a, b)` $(a < b ? a : b)$

Referenced by `MatCommutatorNormF_seq()`.

17.21.6 Function Documentation

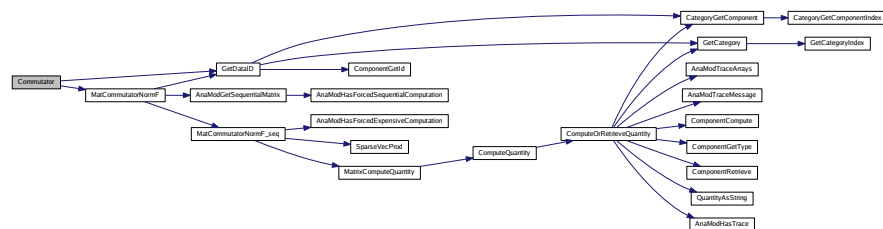
17.21.6.1 **static** `PetscErrorCode Commutator (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg)` `[static]`

Definition at line 417 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:



17.21.6.2 PetscErrorCode CommutatorNormFAllowSqrtTimes (int *n*)

Definition at line 283 of file normal.c.

Referenced by AnaModRegisterSalsaModules().

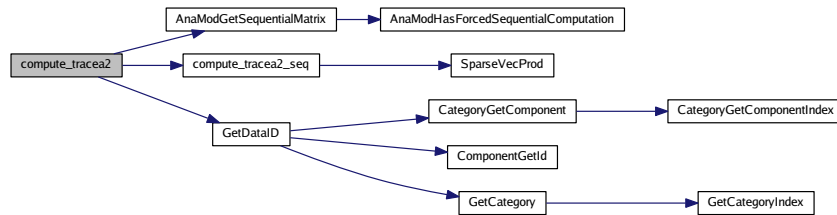
17.21.6.3 static PetscErrorCode compute_tracea2 (Mat *A*, PetscBool * *done*) [static]

Definition at line 223 of file normal.c.

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

Referenced by TraceA2().

Here is the call graph for this function:



17.21.6.4 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:



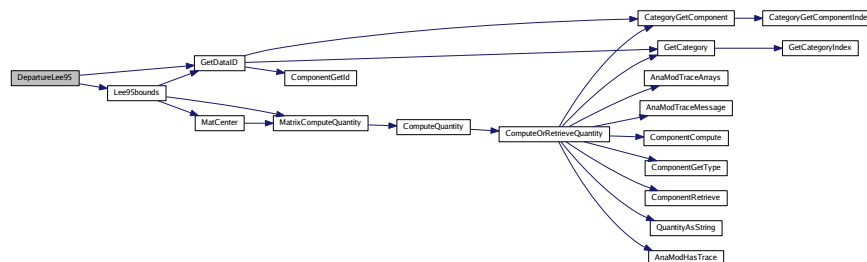
17.21.6.5 static PetscErrorCode DepartureLee95 (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

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:



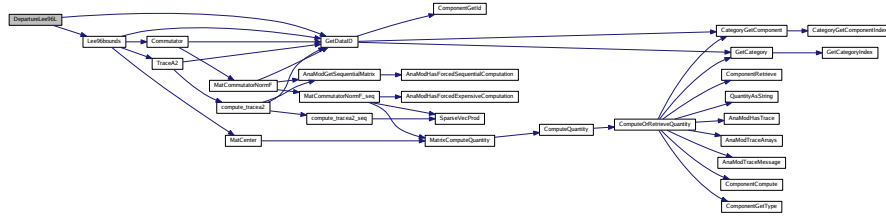
17.21.6.6 static PetscErrorCode DepartureLee96L (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 508 of file normal.c.

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

Referenced by RegisterNormalityModules().

Here is the call graph for this function:



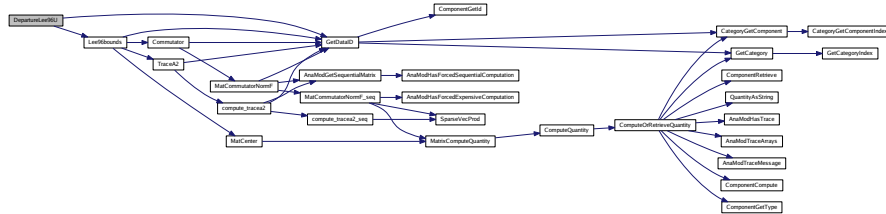
17.21.6.7 static PetscErrorCode DepartureLee96U (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 484 of file normal.c.

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

Referenced by `RegisterNormalityModules()`.

Here is the call graph for this function:



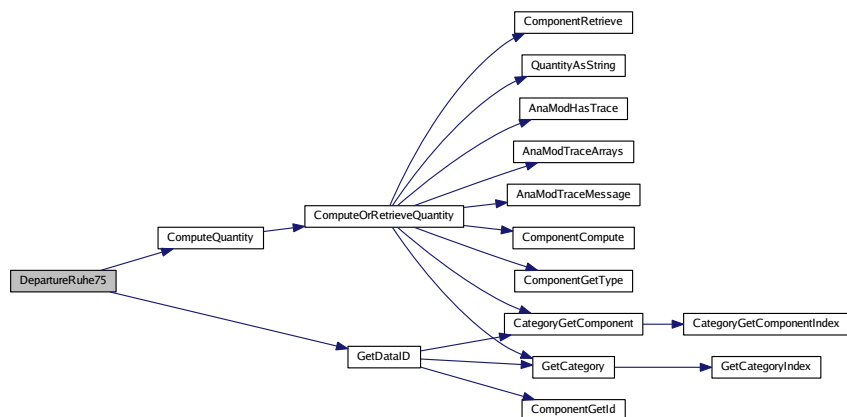
17.21.6.8 static PetscErrorCode DepartureRuhe75 (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 532 of file normal.c.

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

Referenced by `RegisterNormalityModules()`.

Here is the call graph for this function:



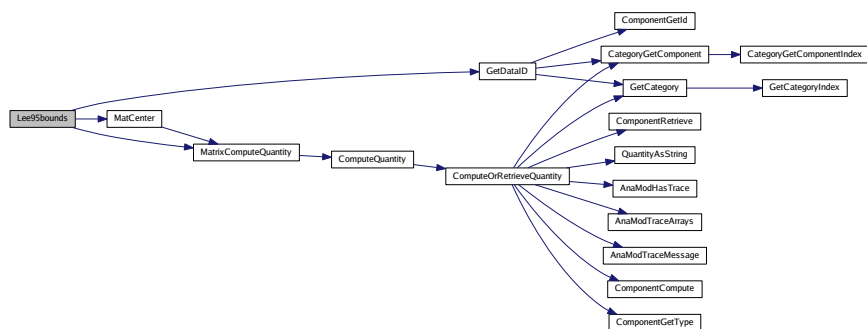
17.21.6.9 static PetscErrorCode Lee95bounds (Mat A) [static]

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:

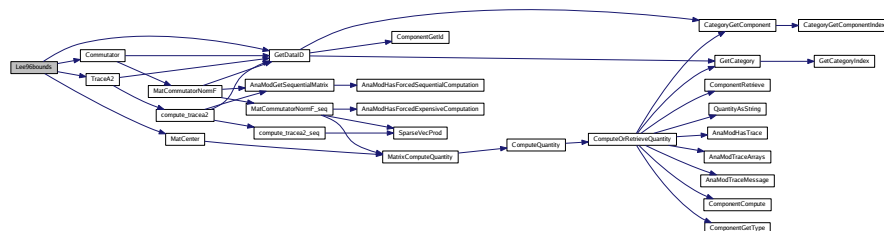


Definition at line 442 of file normal.c.

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

Referenced by DepartureLee96L(), and DepartureLee96U().

Here is the call graph for this function:

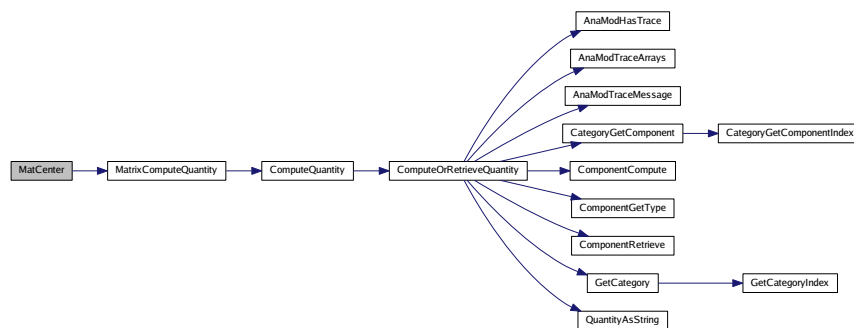


Definition at line 105 of file normal.c.

References MatrixComputeQuantity(), and AnalysisItem::r.

Referenced by Lee95bounds(), and Lee96bounds().

Here is the call graph for this function:



17.21.6.12 **static PetscErrorCode MatCommutatorNormF (Mat A, PetscBool * done)**
[static]

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

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 [CommutatorNormAllowSqrtTimes\(\)](#).

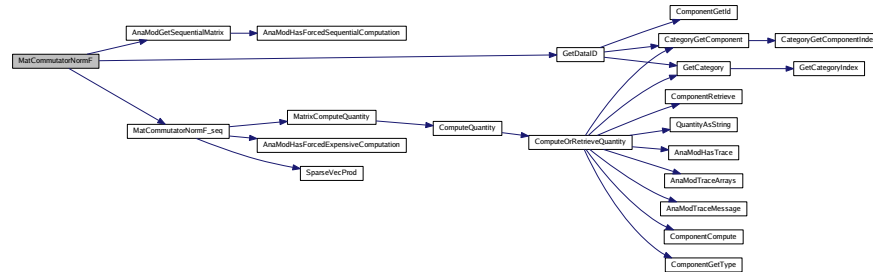
See [MatCommutatorNormF_seq\(\)](#) for the actual computation.

Definition at line 381 of file normal.c.

References [AnaModGetSequentialMatrix\(\)](#), [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), and [MatCommutatorNormF_seq\(\)](#).

Referenced by [Commutator\(\)](#).

Here is the call graph for this function:



17.21.6.13 **static PetscErrorCode MatCommutatorNormF_seq (Mat A, PetscReal * result, PetscBool * done)** [static]

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

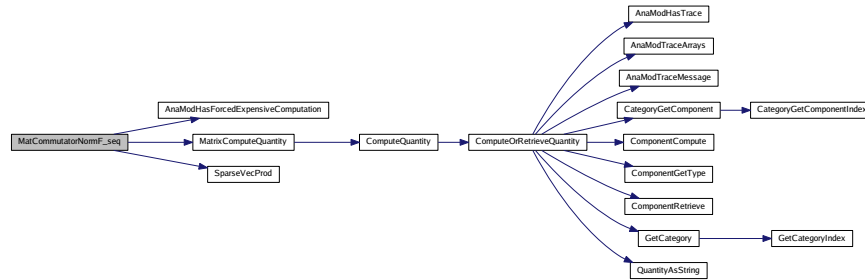
This routine is accessed through [MatCommutatorNormF\(\)](#).

Definition at line 299 of file normal.c.

References [AnaModHasForcedExpensiveComputation\(\)](#), [MatrixComputeQuantity\(\)](#), [max](#), [min](#), and [SparseVecProd\(\)](#).

Referenced by [MatCommutatorNormF\(\)](#).

Here is the call graph for this function:



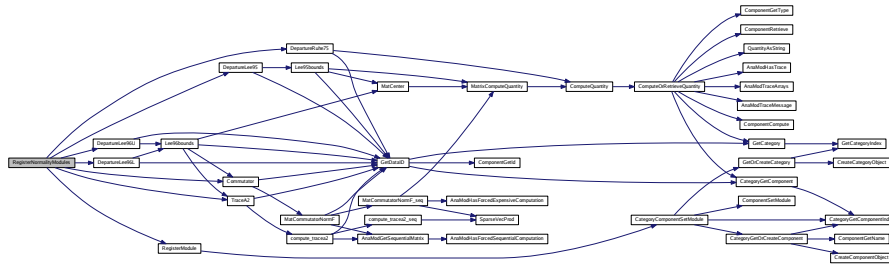
17.21.6.14 PetscErrorCode RegisterNormalityModules (void)

Definition at line 583 of file normal.c.

References ANALYSISDOUBLE, Commutator(), DepartureLee95(), DepartureLee96-L(), DepartureLee96U(), DepartureRuhe75(), RegisterModule(), and TraceA2().

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandardModules().

Here is the call graph for this function:



17.21.6.15 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().

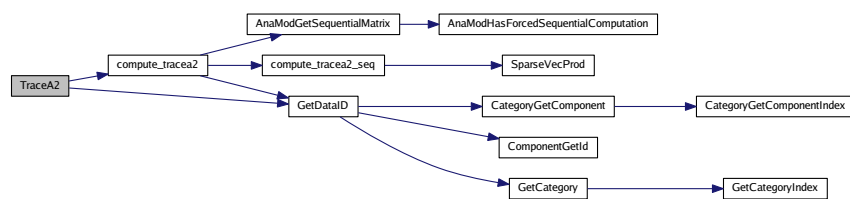
17.21.6.16 `static PetscErrorCode TraceA2 (AnaModNumericalProblem prob,
 AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

Definition at line 253 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:



17.21.7 Variable Documentation

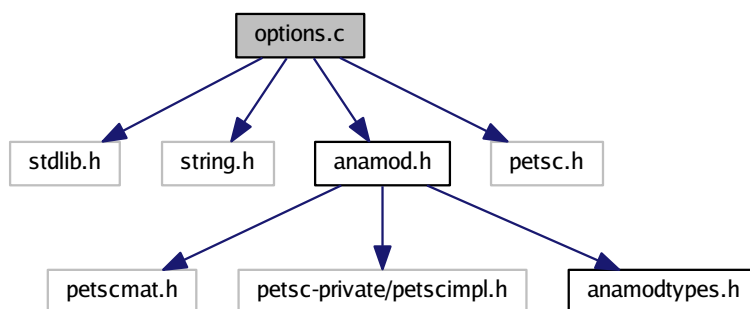
17.21.7.1 `int sqrt_times = 50 [static]`

Definition at line 276 of file normal.c.

17.22 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](#) (PetscBool *flg)
- PetscErrorCode [AnaModGetSequentialMatrix](#) (Mat A, Mat *Awork, PetscBool *mem, PetscBool *local, PetscBool *global)
- PetscErrorCode [AnaModHasForcedExpensiveComputation](#) (PetscBool *flg)

Variables

- static PetscBool [single_proc](#) = PETSC_FALSE
- static PetscBool [expensive](#) = PETSC_FALSE

17.22.1 Detailed Description

Commandline options handling.

Definition in file [options.c](#).

17.22.2 Define Documentation

17.22.2.1 #define VALUELEN 150

Referenced by AnaModOptionsHandling().

17.22.3 Function Documentation

17.22.3.1 PetscErrorCode AnaModGetSequentialMatrix (Mat *A*, Mat * *Awork*, PetscBool * *mem*, PetscBool * *local*, PetscBool * *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:

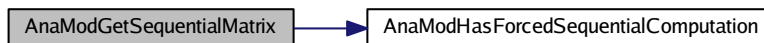
- *A* : input matrix
- *Awork* : pointer to the sequential matrix, this can be a pointer to the original matrix if it was already sequential; it is NULL if no forced sequential computation is asked (see [Commandline options](#))
- *mem* : true if *Awork* is newly allocated
- *local* : true if this processor 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 192 of file options.c.

References AnaModHasForcedSequentialComputation().

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

Here is the call graph for this function:



17.22.3.2 PetscErrorCode AnaModHasForcedExpensiveComputation (PetscBool * flg)

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

Definition at line 238 of file options.c.

References [expensive](#).

Referenced by [MatCommutatorNormF_seq\(\)](#).

17.22.3.3 PetscErrorCode AnaModHasForcedSequentialComputation (PetscBool * flg)

Query whether parallel modules should be done on processor zero.

Definition at line 165 of file options.c.

References [single_proc](#).

Referenced by [AnaModGetSequentialMatrix\(\)](#).

17.22.3.4 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.

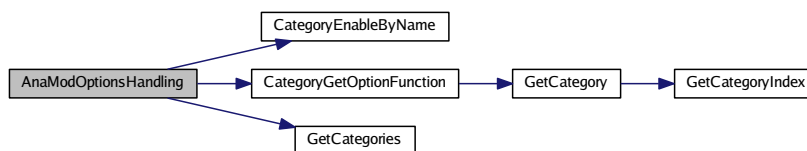
See [DeclareCategoryOptionFunction\(\)](#), [CategoryGetOptionFunction\(\)](#), [AnaModOptionsHandling\(\)](#) and section [Commandline options](#).

Definition at line 60 of file options.c.

References [CATCMP_SKIP_FROM_LOOPS](#), [CategoryEnableByName\(\)](#), [CategoryGetOptionFunction\(\)](#), [expensive](#), [GetCategories\(\)](#), [single_proc](#), and [VALUELEN](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



17.22.3.5 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](#).

Definition at line 136 of file options.c.

References [GetCategories\(\)](#).

Here is the call graph for this function:

**17.22.4 Variable Documentation****17.22.4.1 PetscBool expensive = PETSC_FALSE [static]**

Definition at line 46 of file options.c.

Referenced by [AnaModHasForcedExpensiveComputation\(\)](#), and [AnaModOptionsHandling\(\)](#).

17.22.4.2 PetscBool single_proc = PETSC_FALSE [static]

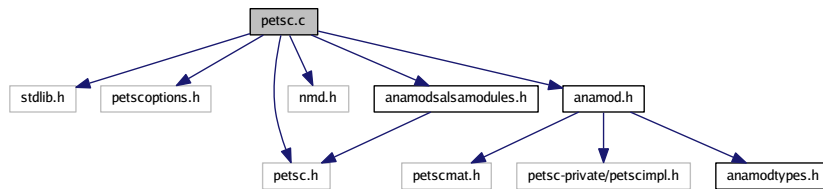
Definition at line 46 of file options.c.

Referenced by [AnaModHasForcedSequentialComputation\(\)](#), and [AnaModOptionsHandling\(\)](#).

17.23 **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, PetscBool *success)
- PetscErrorCode [analyze_matrix](#) (Mat A, NMD_metadata nmd)
- static PetscErrorCode [usage](#) (MPI_Comm comm)
- int [main](#) (int argc, char **argv)

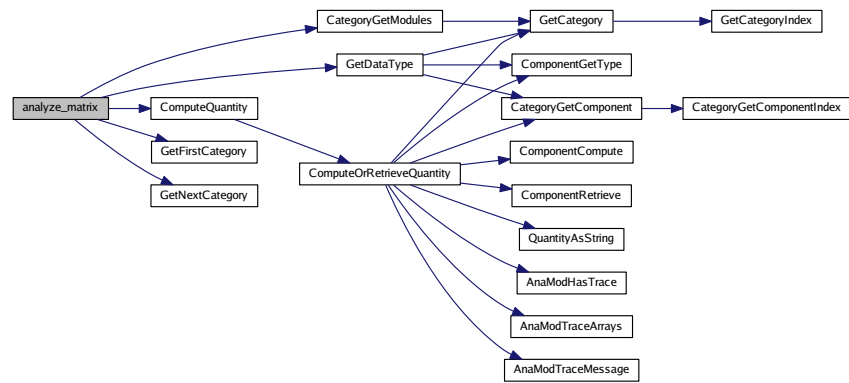
17.23.1 Function Documentation

17.23.1.1 PetscErrorCode `analyze_matrix` (Mat A, NMD_metadata *nmd*)

Definition at line 49 of file `petsc.c`.

References `CategoryGetModules()`, `ComputeQuantity()`, `GetDataType()`, `GetFirstCategory()`, and `GetNextCategory()`.

Here is the call graph for this function:



17.23.1.2 static PetscErrorCode get_matrix (Mat * A, PetscBool * success) [static]

Definition at line 14 of file Petsc.c.

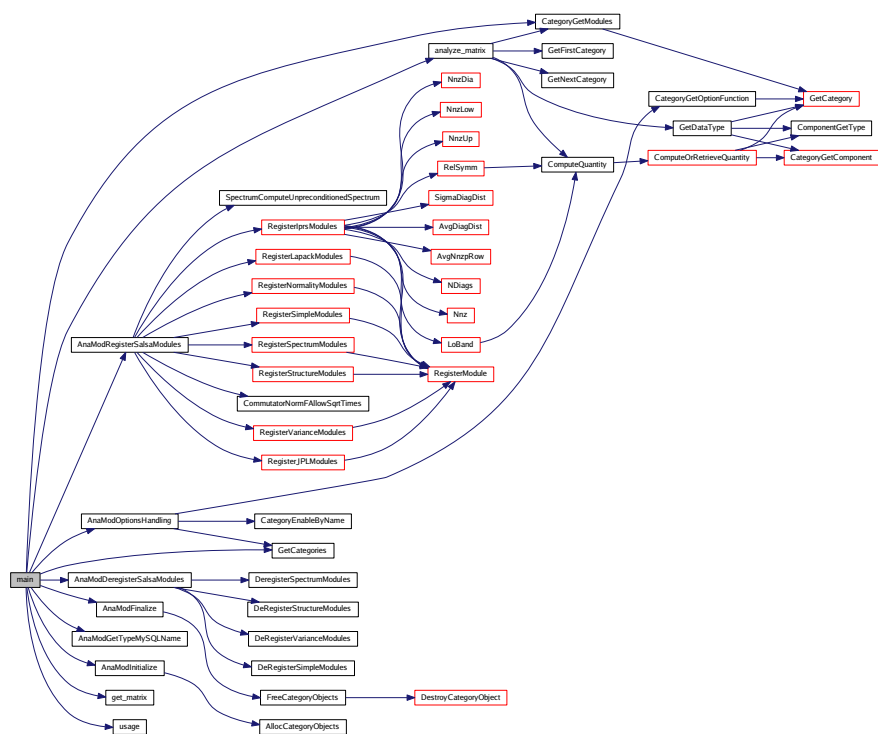
Referenced by main().

17.23.1.3 int main (int argc, char ** argv)

Definition at line 92 of file Petsc.c.

References analyze_matrix(), AnaModDeregisterSalsaModules(), AnaModFinalize(), - AnaModGetTypeMySQLName(), AnaModInitialize(), AnaModOptionsHandling(), AnaModRegisterSalsaModules(), CategoryGetModules(), get_matrix(), GetCategories(), and usage().

Here is the call graph for this function:



17.23.1.4 static PetscErrorCode usage (MPI_Comm comm) [static]

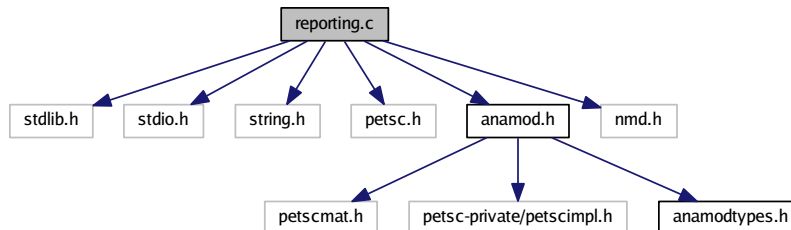
Definition at line 77 of file petsc.c.

Referenced by main().

17.24 reporting.c File Reference

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

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)

17.24.1 Function Documentation

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

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

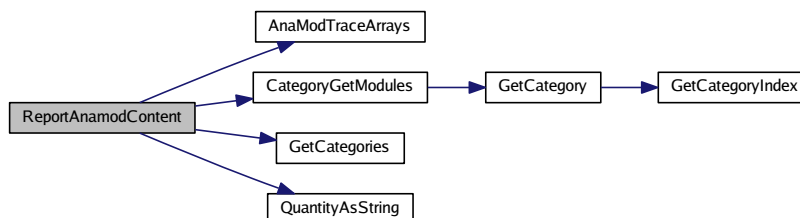
- if the rval parameter is null, no value string is generated
- separator is a single character

Definition at line 19 of file reporting.c.

References [AnaModTraceArrays\(\)](#), [CategoryGetModules\(\)](#), [GetCategories\(\)](#), and [QuantityAsString\(\)](#).

Referenced by [TabReportModules\(\)](#), and [TabReportValues\(\)](#).

Here is the call graph for this function:



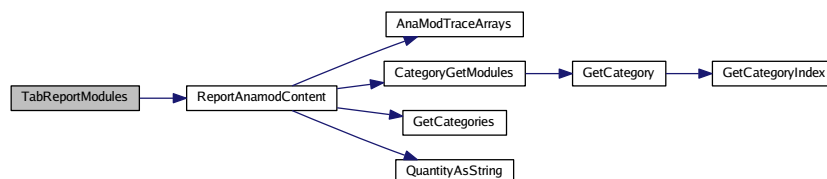
17.24.1.2 PetscErrorCode TabReportModules (char ** rkey, int separator)

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:



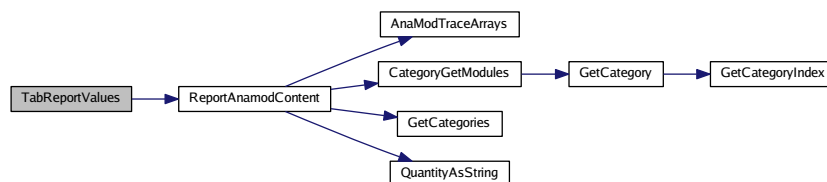
17.24.1.3 PetscErrorCode TabReportValues (NMD_metadata nmd, char ** rkey, char ** rval, int separator)

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()`.

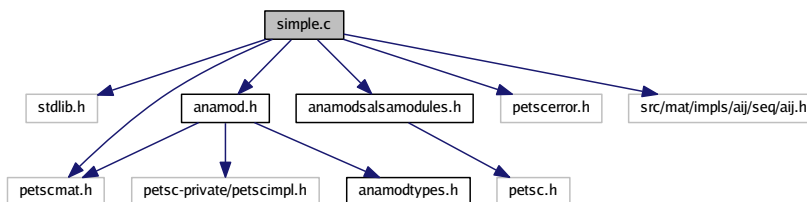
Here is the call graph for this function:



17.25 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(MPI_COMM_WORLD,1,"Violation of <%s> @ (%d,%d): %d",s,i,j);
- #define [ASSERT](#)(v, s, i, j, t) if (!(v)) SETERRQ4(MPI_COMM_WORLD,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, PetscBool *flg)
- static PetscErrorCode [TraceAbs](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [norm1](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [normF](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [normInf](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [compute_dd](#) (Mat A, PetscBool *flg)
- static PetscErrorCode [DiagonalDominance](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MatSymmPartNormInf_seq](#) (Mat A, PetscReal *sn_r, - PetscReal *an_r, PetscReal *srn_r, PetscReal *arn_r, int *nun_r, PetscBool *done)
- static PetscErrorCode [MatSymmPartNormInf](#) (Mat A, PetscBool *done)
- static PetscErrorCode [SymmetrySNorm](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SymmetryANorm](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SymmetryFSNorm](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SymmetryFANorm](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [NUnstruct](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterSimpleModules](#) ()
- PetscErrorCode [DeRegisterSimpleModules](#) (void)

17.25.1 Detailed Description

Norm-like properties.

17.25.2 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.

17.25.3 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 [N-Unstruct\(\)](#). (this is calculated here, but declared as an element of the [Structural properties](#) category.)

Definition in file [simple.c](#).

17.25.4 Define Documentation

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

Referenced by [MatSymmPartNormInf_seq\(\)](#).

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

Referenced by `MatSymmPartNormInf_seq()`.

17.25.4.3 `#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()`.

17.25.5 Function Documentation

17.25.5.1 `static PetscErrorCode compute_dd (Mat A, PetscBool * flg) [static]`

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

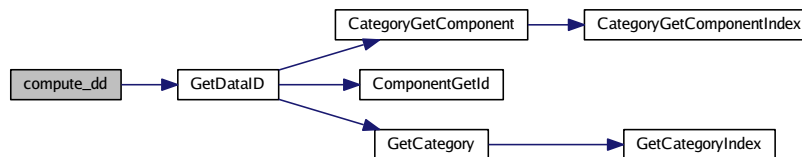
$$\min_i a_{ii} - \sum_{j \neq i} |a_{ij}|$$

Definition at line 246 of file `simple.c`.

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

Referenced by `DiagonalDominance()`.

Here is the call graph for this function:



17.25.5.2 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.

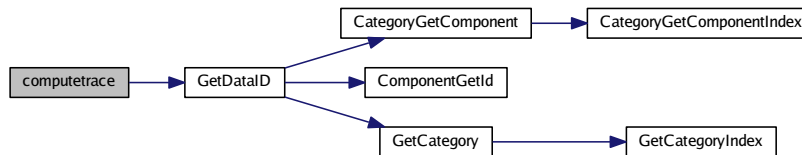
A Petsc vector is created and destroyed.

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:



17.25.5.3 PetscErrorCode DeRegisterSimpleModules (void)

Definition at line 694 of file simple.c.

Referenced by AnaModDeregisterSalsaModules().

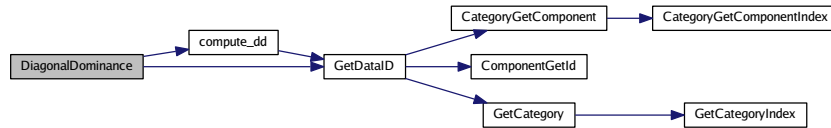
17.25.5.4 static PetscErrorCode DiagonalDominance (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

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:



17.25.5.5 static PetscErrorCode MatSymmPartNormInf (Mat A, PetscBool * 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.

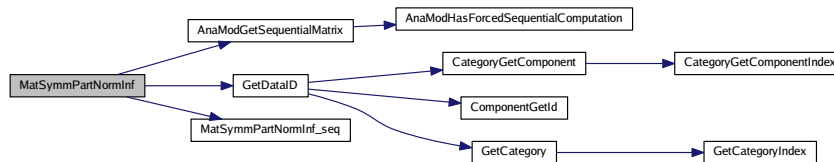
See [MatSymmPartNormInf_seq\(\)](#) for the actual computation.

Definition at line 445 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:



17.25.5.6 static PetscErrorCode MatSymmPartNormInf_seq (Mat A, PetscReal * sn_r, PetscReal * an_r, PetscReal * srn_r, PetscReal * arn_r, int * nun_r, PetscBool * 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().

17.25.5.7 `static PetscErrorCode norm1 (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

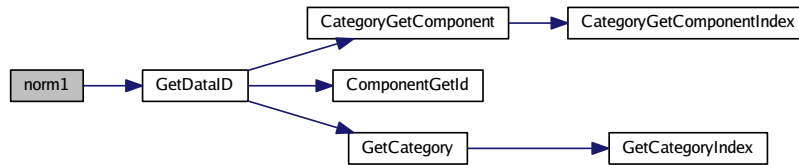
Compute the 1-norm of a matrix.

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:



17.25.5.8 `static PetscErrorCode normF (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

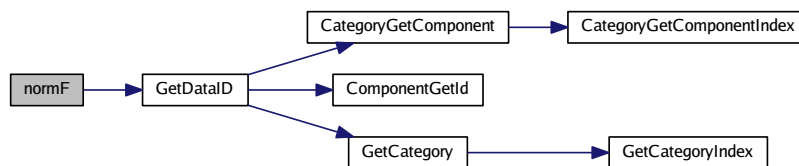
Compute the Frobenius norm of a matrix.

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:



**17.25.5.9 static PetscErrorCode normInf (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flag*) [static]**

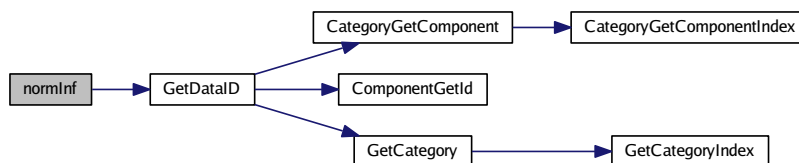
Compute the infinity-norm of a matrix.

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:



**17.25.5.10 static PetscErrorCode NUnstruct (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flag*) [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.

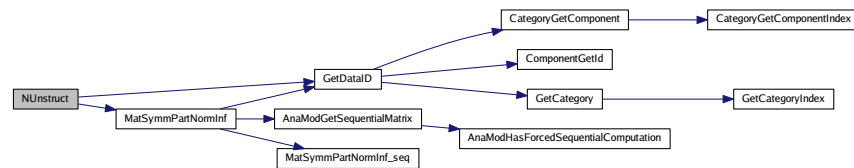
The "n-struct-unsymm" feature is declared to be in category "structure", rather than simple.

Definition at line 630 of file simple.c.

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

Referenced by `RegisterSimpleModules()`.

Here is the call graph for this function:



17.25.5.11 PetscErrorCode RegisterSimpleModules (void)

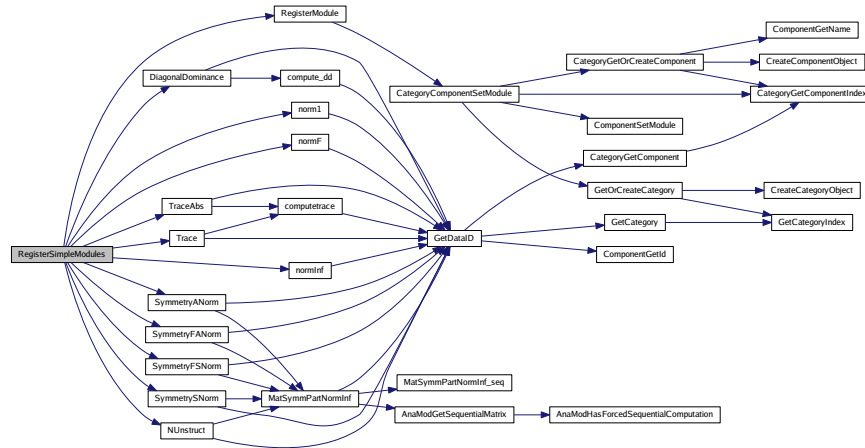
Declare norm-like modules that can be performed with simple calculations.

Definition at line 657 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:



17.25.5.12 static PetscErrorCode SymmetryANorm (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

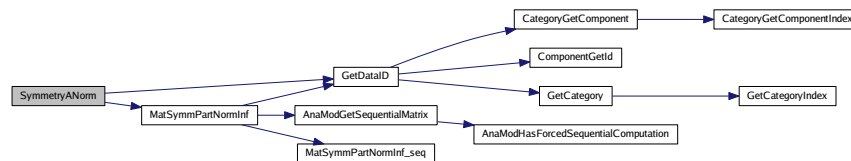
Compute and store the infinity norm of the antisymmetric part of the matrix. The actual computation is done in [MatSymmPartNormInf\(\)](#); see that routine for remarks on parallelism.

Definition at line 540 of file simple.c.

References [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), [MatSymmPartNormInf\(\)](#), and [AnalysisItem::r](#).

Referenced by [RegisterSimpleModules\(\)](#).

Here is the call graph for this function:



17.25.5.13 static PetscErrorCode SymmetryFANorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

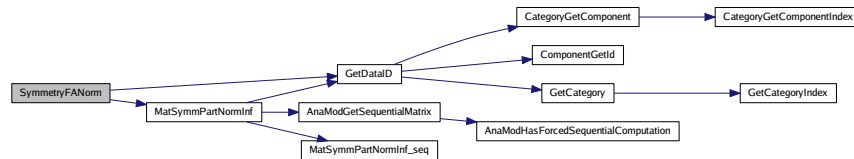
Compute and store the Frobenius norm of the antisymmetric part of the matrix. The actual computation is done in [MatSymmPartNormInf\(\)](#); see that routine for remarks on parallelism.

Definition at line 598 of file simple.c.

References [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), [MatSymmPartNormInf\(\)](#), and [AnalysisItem::r](#).

Referenced by [RegisterSimpleModules\(\)](#).

Here is the call graph for this function:



17.25.5.14 static PetscErrorCode SymmetryFSNorm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

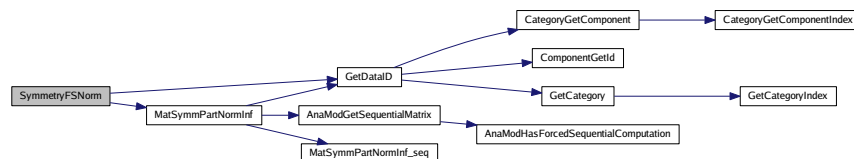
Compute and store the Frobenius norm of the symmetric part of the matrix. The actual computation is done in [MatSymmPartNormInf\(\)](#); see that routine for remarks on parallelism.

Definition at line 569 of file simple.c.

References [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), [MatSymmPartNormInf\(\)](#), and [AnalysisItem::r](#).

Referenced by [RegisterSimpleModules\(\)](#).

Here is the call graph for this function:



17.25.5.15 `static PetscErrorCode SymmetrySNorm (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

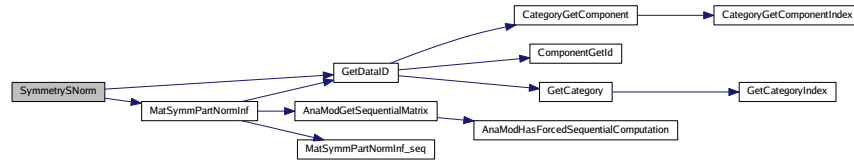
Compute and store the infinity norm of the symmetric part of the matrix. The actual computation is done in [MatSymmPartNormInf\(\)](#); see that routine for remarks on parallelism.

Definition at line 511 of file simple.c.

References [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), [MatSymmPartNormInf\(\)](#), and [AnalysisItem::r](#).

Referenced by [RegisterSimpleModules\(\)](#).

Here is the call graph for this function:



17.25.5.16 `static PetscErrorCode Trace (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

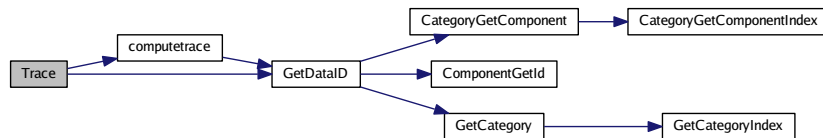
Compute and store the trace of a matrix; the computation is done in [computetrace\(\)](#).

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:



17.25.5.17 `static PetscErrorCode TraceAbs (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

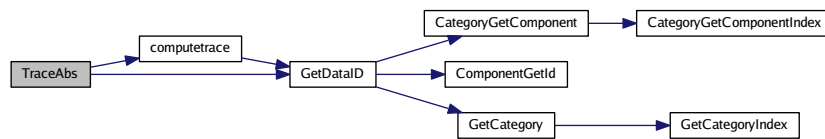
Compute and store the absolute trace of a matrix; the computation is done in [compute-trace\(\)](#).

Definition at line 135 of file simple.c.

References [computetrace\(\)](#), [GetDataID\(\)](#), [HASTOEXIST](#), [id](#), and [AnalysisItem::r](#).

Referenced by [RegisterSimpleModules\(\)](#).

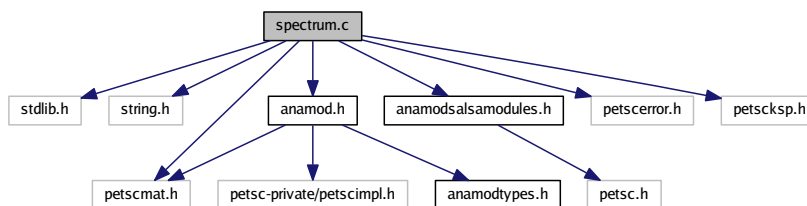
Here is the call graph for this function:



17.26 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, PetscBool *success, const char **reason)
- static PetscErrorCode [compute_eigenvalues](#) (Mat A, PetscReal **r, PetscReal **c, int *neig, PetscBool *success, const char **reason)
- static PetscErrorCode [compute_singularvalues](#) (Mat A, PetscBool *success, const 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, PetscBool *flg)
- static PetscErrorCode [RitzValuesR](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [RitzValuesC](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SpectrumAX](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SpectrumAY](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SpectrumCX](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SpectrumCY](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [PosFraction](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SigmaMax](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SigmaMin](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [Kappa](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbyMagRe](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbyMagIm](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MinEVbyMagRe](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MinEVbyMagIm](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)

- static PetscErrorCode [MaxEVbyRealRe](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbyReallm](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbylmRe](#) ([AnaModNumericalProblem](#) prob, - [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxEVbylmIm](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [SpectrumOptions](#) (char *opt)
- PetscErrorCode [RegisterSpectrumModules](#) ()
- PetscErrorCode [DeregisterSpectrumModules](#) ()

Variables

- static PetscBool [trace_spectrum](#) = PETSC_FALSE
- static PetscBool [trace_hessenberg](#) = PETSC_FALSE
- static int [use_prec](#) = 0

17.26.1 Detailed Description

Various estimates of spectrum related quantities.

17.26.2 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.

17.26.2.1 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

17.26.2.2 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.

17.26.2.3 Tracing

See [SpectrumOptions\(\)](#).

Definition in file [spectrum.c](#).

17.26.3 Function Documentation

17.26.3.1 `static PetscErrorCode compute_eigenvalues (Mat A, PetscReal ** r, PetscReal ** c, int * neig, PetscBool * success, const char ** reason) [static]`

Compute a number of eigenvalues (returned as separate real and imaginary parts) of a matrix.

Parameters:

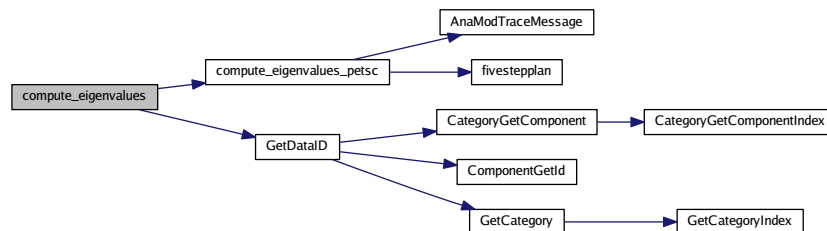
- `success` : PETSC_TRUE or PETSC_FALSE
- `reason` : explanation of the success parameter. This is a static string; does not need to be freed.
- `r,c` : arrays of real and imaginary part of the eigenvalues. If success is false, these are NULL. Otherwise, the calling environment needs to free them at some point.
- `neig` : length of the `r` and `c` arrays. This is zero if success is false.

Definition at line 466 of file `spectrum.c`.

References `compute_eigenvalues_petsc()`, `GetDataID()`, `HASTOEXIST`, and `id`.

Referenced by `NRitzValues()`, `PosFraction()`, `RitzValuesC()`, `RitzValuesR()`, `SpectrumAX()`, `SpectrumAY()`, `SpectrumCX()`, and `SpectrumCY()`.

Here is the call graph for this function:



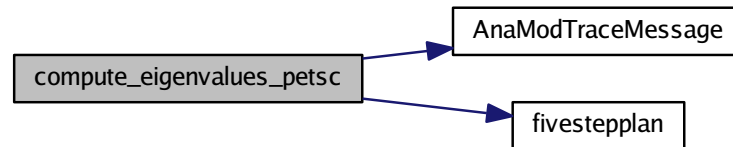
17.26.3.2 `static PetscErrorCode compute_eigenvalues_petsc (Mat A, PetscReal ** r, PetscReal ** c, int * rneig, PetscReal * rx, PetscReal * rm, PetscBool * success, const char ** reason) [static]`

Definition at line 296 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:



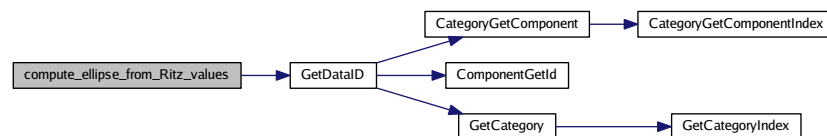
17.26.3.3 `static PetscErrorCode compute_ellipse_from_Ritz_values (Mat A, PetscReal * r, PetscReal * c, int neig) [static]`

Definition at line 544 of file spectrum.c.

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

Referenced by `NRitzValues()`, `PosFraction()`, `RitzValuesC()`, `RitzValuesR()`, `SpectrumA-X()`, `SpectrumAY()`, `SpectrumCX()`, and `SpectrumCY()`.

Here is the call graph for this function:



17.26.3.4 **static PetscErrorCode compute_singularvalues (Mat A, PetscBool * *success*, const char ** *reason*)** [static]

Compute a number of singular values of a matrix.

Parameters:

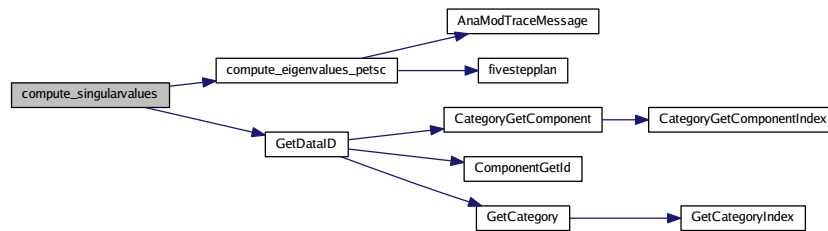
- *success* : PETSC_TRUE or PETSC_FALSE
- *reason* : explanation of the success parameter. This is a static string; does not need to be freed.

Definition at line 512 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:



17.26.3.5 **PetscErrorCode DeregisterSpectrumModules (void)**

Definition at line 1414 of file spectrum.c.

Referenced by AnaModDeregisterSalsaModules().

17.26.3.6 **static PetscErrorCode fivestepplan (KSP solver, PetscInt it, PetscReal err, KSPConvergedReason * *reason*, void * *ctx*)** [static]

Definition at line 114 of file spectrum.c.

Referenced by compute_eigenvalues_petsc().

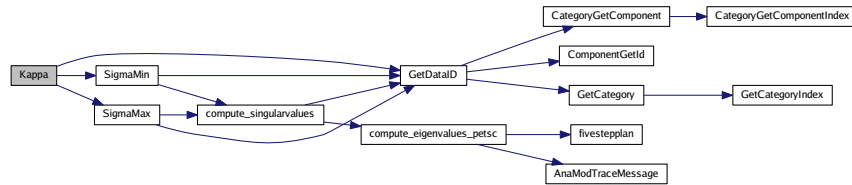
17.26.3.7 **static PetscErrorCode Kappa (AnaModNumericalProblem prob, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 970 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



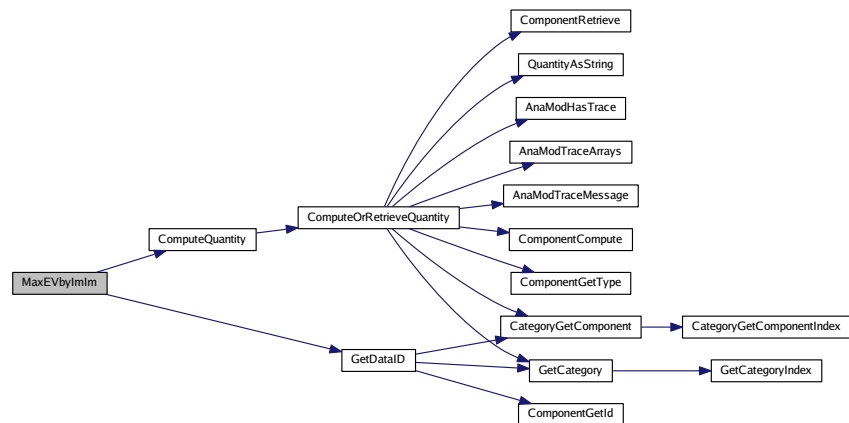
17.26.3.8 static PetscErrorCode MaxEVbyImlm (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 1282 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



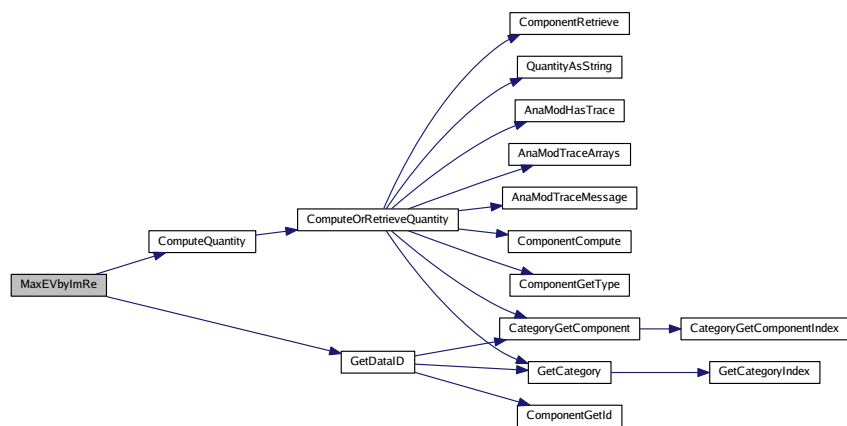
17.26.3.9 **static PetscErrorCode MaxEVbyImRe (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 1244 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



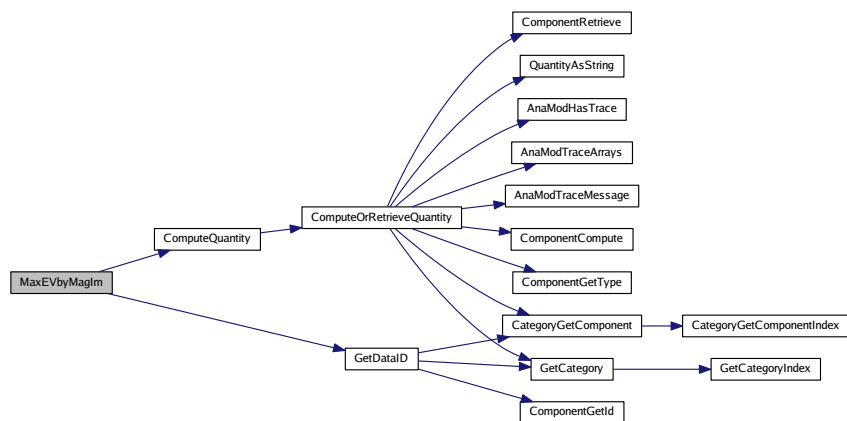
17.26.3.10 **static PetscErrorCode MaxEVbyMagIm (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 1051 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



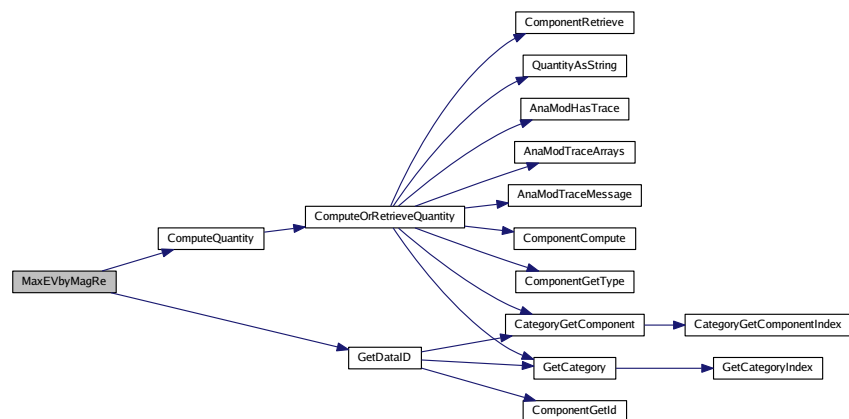
17.26.3.11 `static PetscErrorCode MaxEVbyMagRe (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg)` [static]

Definition at line 1012 of file `spectrum.c`.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



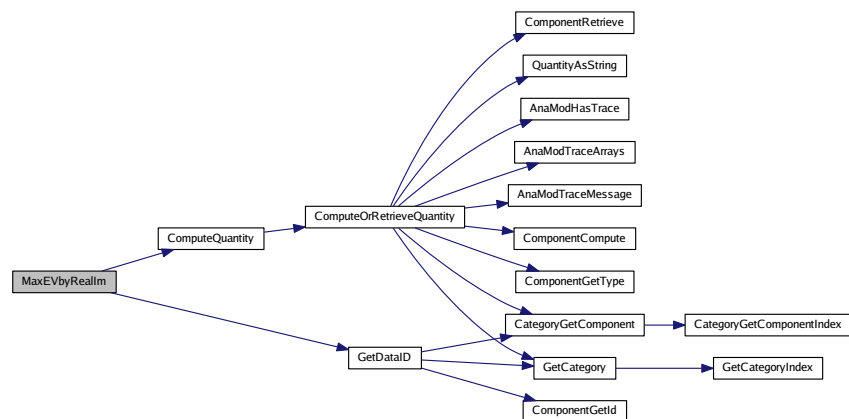
17.26.3.12 static PetscErrorCode MaxEVbyReallm (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 1206 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



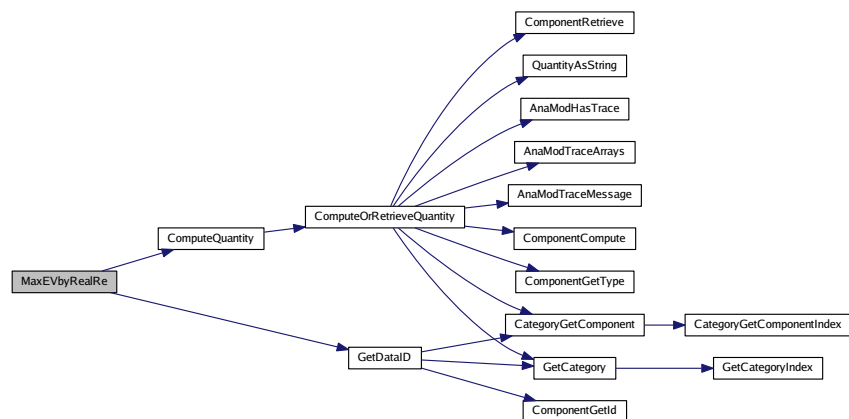
17.26.3.13 static PetscErrorCode MaxEVbyRealRe (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 1168 of file spectrum.c.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



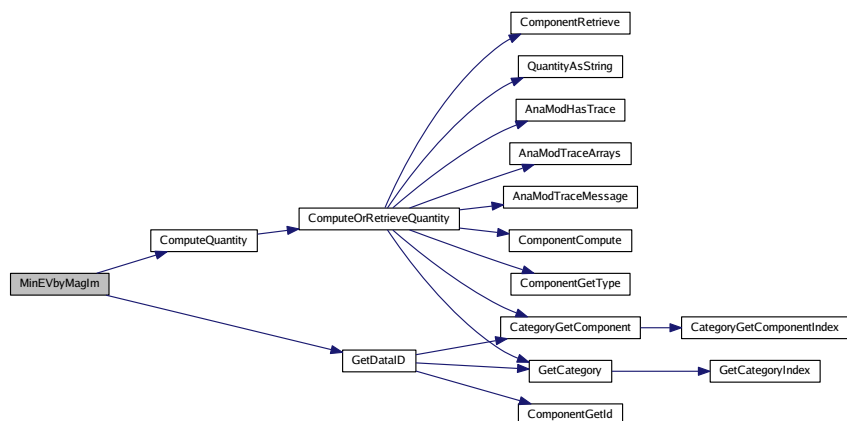
17.26.3.14 `static PetscErrorCode MinEVbyMaglm (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg)` [static]

Definition at line 1129 of file `spectrum.c`.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



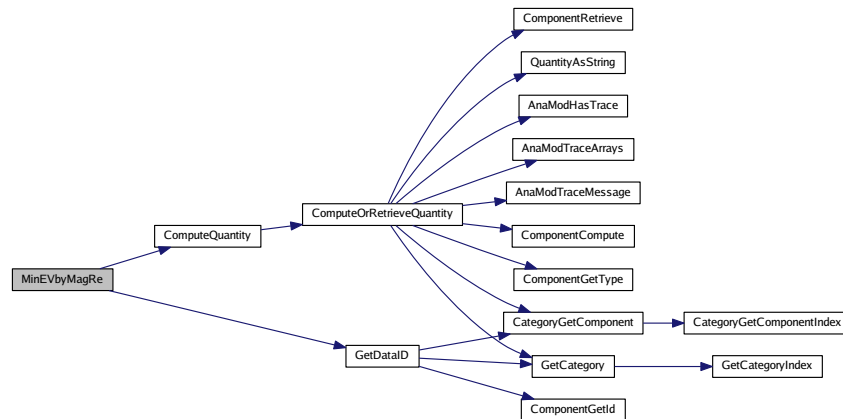
17.26.3.15 `static PetscErrorCode MinEVbyMagRe (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

Definition at line 1090 of file `spectrum.c`.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



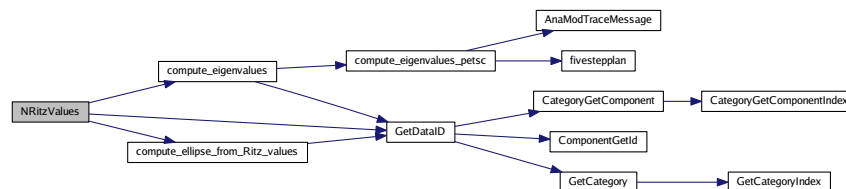
17.26.3.16 static PetscErrorCode NRitzValues (AnaModNumericalProblem *prob*,
AnalysisItem * *iv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 612 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:



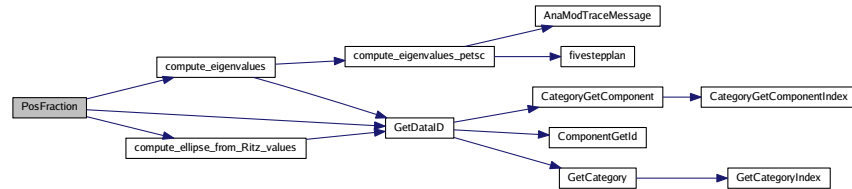
17.26.3.17 static PetscErrorCode PosFraction (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 886 of file spectrum.c.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



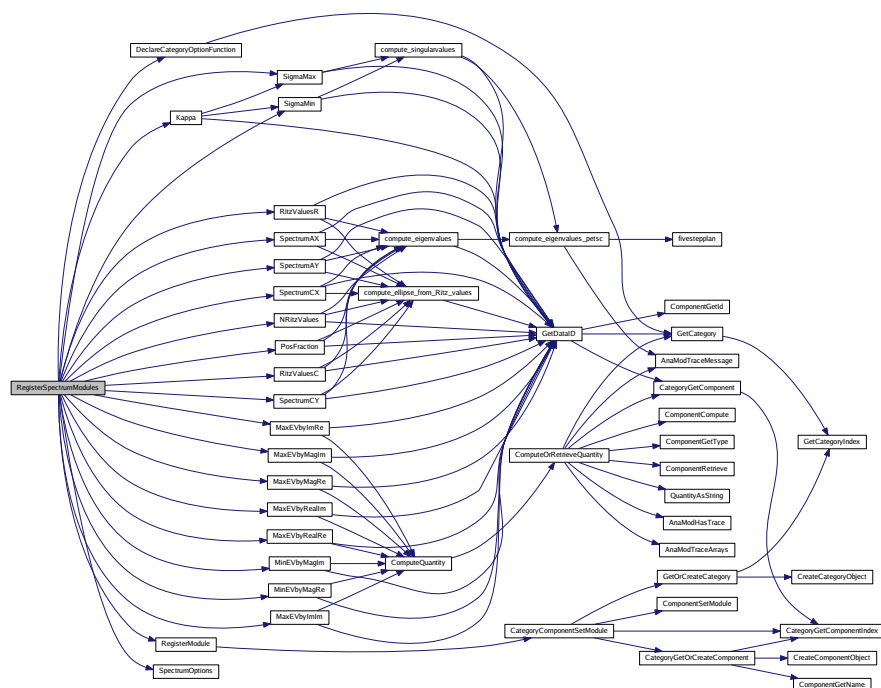
17.26.3.18 PetscErrorCode RegisterSpectrumModules (void)

Definition at line 1343 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:



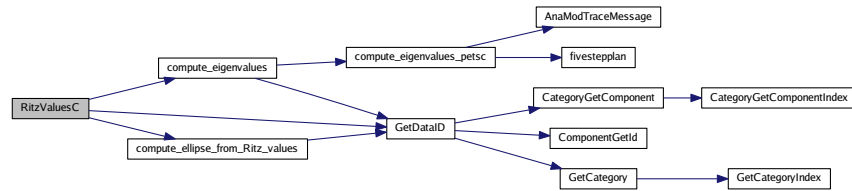
```
17.26.3.19 static PetscErrorCode RitzValuesC ( AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg ) [static]
```

Definition at line 685 of file spectrum.c.

References `compute_eigenvalues()`, `compute_ellipse_from_Ritz_values()`, `GetDataLD()`, `HASTOEXIST`, `id`, and `AnalysisItem::rr`.

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



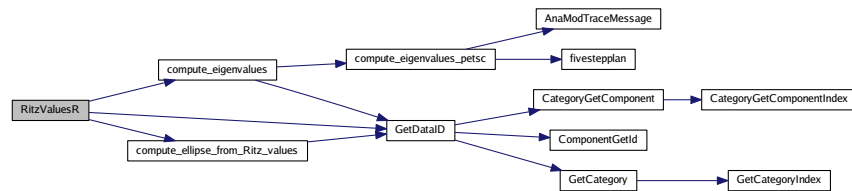
17.26.3.20 static PetscErrorCode RitzValuesR (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 645 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:



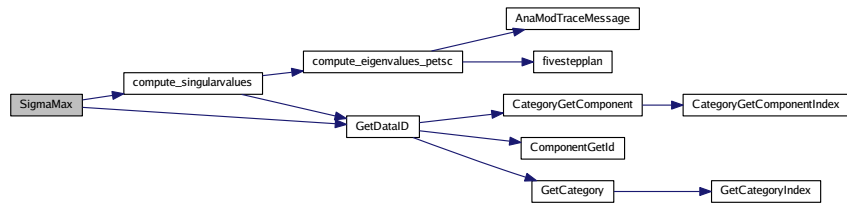
17.26.3.21 static PetscErrorCode SigmaMax (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 918 of file spectrum.c.

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

Referenced by Kappa(), and RegisterSpectrumModules().

Here is the call graph for this function:



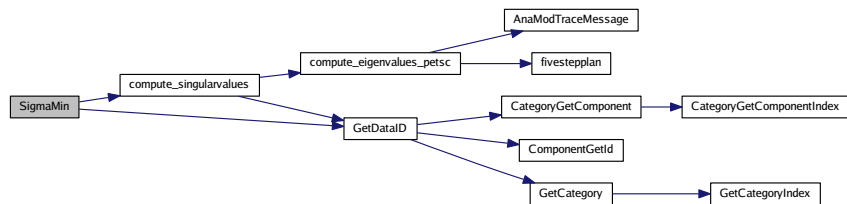
17.26.3.22 `static PetscErrorCode SigmaMin (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

Definition at line 944 of file spectrum.c.

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

Referenced by `Kappa()`, and `RegisterSpectrumModules()`.

Here is the call graph for this function:



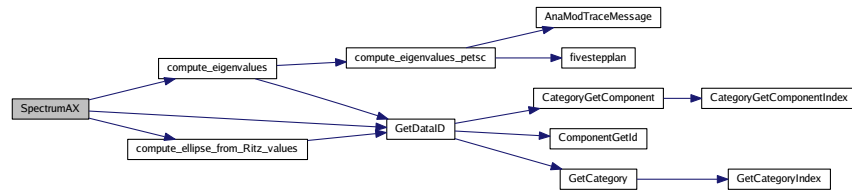
17.26.3.23 `static PetscErrorCode SpectrumAX (AnaModNumericalProblem prob,
AnalysisItem * rv, int * lv, PetscBool * flg) [static]`

Definition at line 757 of file spectrum.c.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



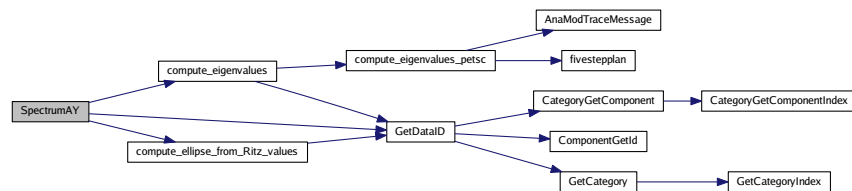
17.26.3.24 static PetscErrorCode SpectrumAY (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 789 of file spectrum.c.

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

Referenced by RegisterSpectrumModules().

Here is the call graph for this function:



17.26.3.25 PetscErrorCode SpectrumComputePreconditionedSpectrum (void)

Definition at line 95 of file spectrum.c.

References use_prec.

17.26.3.26 PetscErrorCode SpectrumComputeUnpreconditionedSpectrum (void)

Definition at line 104 of file spectrum.c.

References use_prec.

Referenced by AnaModRegisterSalsaModules().

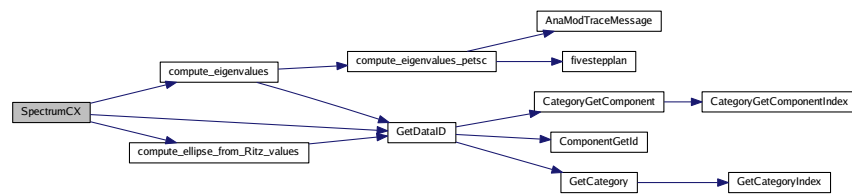
17.26.3.27 **static PetscErrorCode SpectrumCX (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 821 of file spectrum.c.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



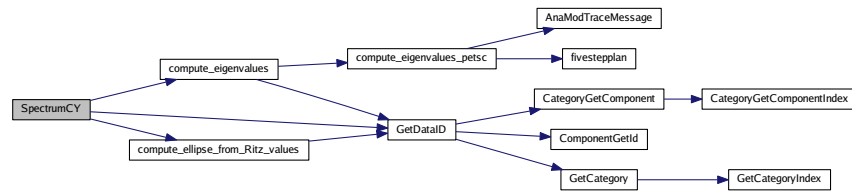
17.26.3.28 **static PetscErrorCode SpectrumCY (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 854 of file spectrum.c.

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

Referenced by `RegisterSpectrumModules()`.

Here is the call graph for this function:



17.26.3.29 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 optionsfile.

Definition at line 1327 of file spectrum.c.

References trace_hessenberg, and trace_spectrum.

Referenced by RegisterSpectrumModules().

17.26.4 Variable Documentation

17.26.4.1 PetscBool trace_hessenberg = PETSC_FALSE [static]

Definition at line 90 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), and SpectrumOptions().

17.26.4.2 PetscBool trace_spectrum = PETSC_FALSE [static]

Definition at line 89 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), and SpectrumOptions().

17.26.4.3 int use_prec = 0 [static]

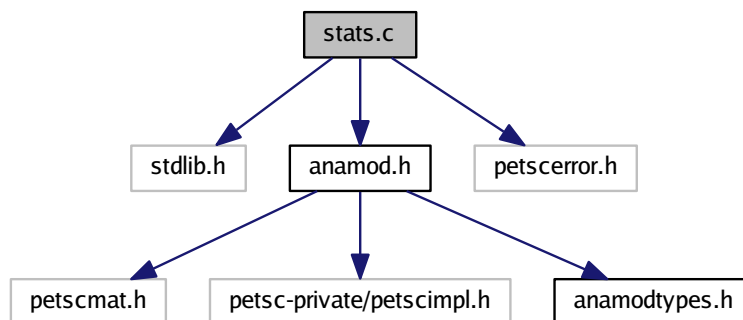
Definition at line 92 of file spectrum.c.

Referenced by compute_eigenvalues_petsc(), SpectrumComputePreconditionedSpectrum(), and SpectrumComputeUnpreconditionedSpectrum().

17.27 stats.c File Reference

Statistics on the AnaMod system.


```
#include <stdlib.h> #include "anamod.h" #include "petscerror.h"
h" Include dependency graph for stats.c:
```



Functions

- static PetscErrorCode [Version](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *l, PetscBool *flg)
- PetscErrorCode [RegisterStatsModules](#) ()

17.27.1 Detailed Description

Statistics on the AnaMod system.

17.27.2 Statistics on the AnaMod system

The stats module needs to be enabled explicitly.

Definition in file [stats.c](#).

17.27.3 Function Documentation

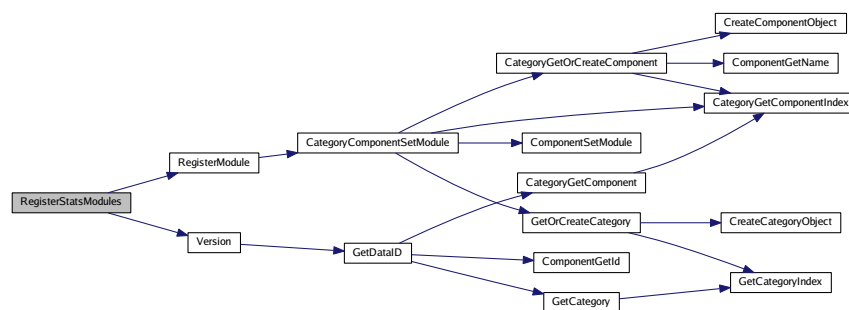
17.27.3.1 PetscErrorCode RegisterStatsModules ()

Declare statistics modules

Definition at line 34 of file stats.c.

References ANALYSISSTRING, RegisterModule(), and Version().

Here is the call graph for this function:



**17.27.3.2 static PetscErrorCode Version (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *I*, PetscBool * *flag*) [static]**

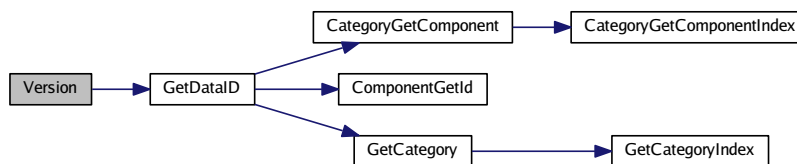
The AnaMod format version string

Definition at line 19 of file stats.c.

References ANAMOD_FORMAT_VERSION, AnalysisItem::c, GetDataID(), HASTOEXIST, and id.

Referenced by RegisterStatsModules().

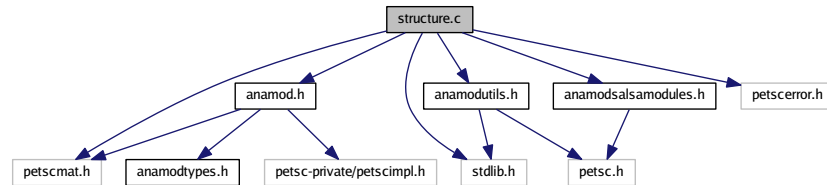
Here is the call graph for this function:



17.28 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, PetscBool *flg)
- static PetscErrorCode [Symmetry](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [compute_nnz_structure](#) ([AnaModNumericalProblem](#) prob)
- static PetscErrorCode [NNonZeros](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MaxNNonZerosPerRow](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [MinNNonZerosPerRow](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [LBandWidth](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [RBandWidth](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [LeftSkyline](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [RightSkyline](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [compute_dummy_rows](#) (Mat A)
- static PetscErrorCode [NDummyRows](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DummyRowsKind](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DummyRows](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [compute_posdiag](#) ([AnaModNumericalProblem](#) prob)

- static PetscErrorCode [DiagZeroStart](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DiagDefinite](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [regularblocks](#) ([AnaModNumericalProblem](#) prob)
- static PetscErrorCode [BlockSize](#) ([AnaModNumericalProblem](#) prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterStructureModules](#) ()
- PetscErrorCode [DeRegisterStructureModules](#) (void)

17.28.1 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.

17.28.2 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.

17.28.3 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-zero-start" : $\min\{i: \forall j > i, a_{jj} = 0\}$
- "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](#).

17.28.4 Function Documentation

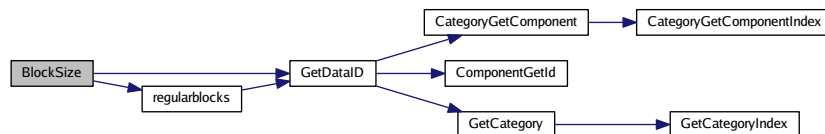
17.28.4.1 static PetscErrorCode BlockSize (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flag*) [static]

Definition at line 646 of file [structure.c](#).

References [GetDataID\(\)](#), [HASTOEXIST](#), [AnalysisItem::i](#), [id](#), and [regularblocks\(\)](#).

Referenced by [RegisterStructureModules\(\)](#).

Here is the call graph for this function:



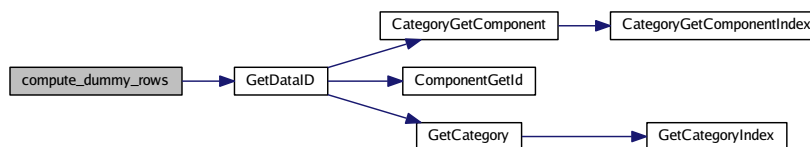
17.28.4.2 static PetscErrorCode compute_dummy_rows (Mat *A*) [static]

Definition at line 351 of file [structure.c](#).

References [GetDataID\(\)](#), [HASTOEXIST](#), and [id](#).

Referenced by [DummyRows\(\)](#), [DummyRowsKind\(\)](#), and [NDummyRows\(\)](#).

Here is the call graph for this function:



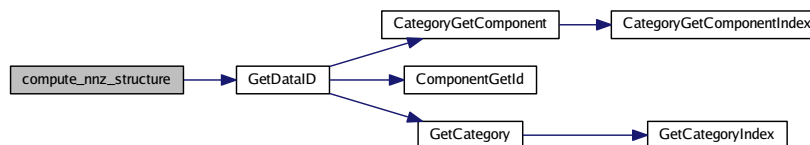
17.28.4.3 static PetscErrorCode compute_nnz_structure (AnaModNumericalProblem *prob*) [static]

Definition at line 109 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:



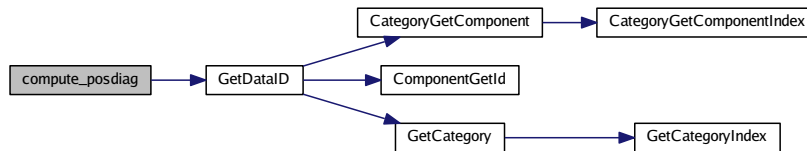
17.28.4.4 static PetscErrorCode compute_posdiag (AnaModNumericalProblem *prob*) [static]

Definition at line 498 of file structure.c.

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

Referenced by `DiagDefinite()`, and `DiagZeroStart()`.

Here is the call graph for this function:



17.28.4.5 PetscErrorCode DeRegisterStructureModules (void)

Definition at line 720 of file structure.c.

Referenced by AnaModDeregisterSalsaModules().

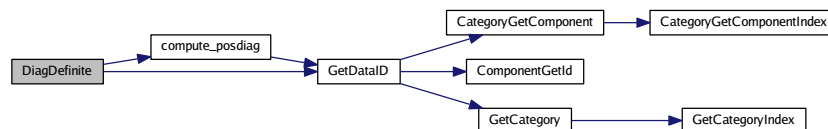
17.28.4.6 static PetscErrorCode DiagDefinite (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 566 of file structure.c.

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

Referenced by RegisterStructureModules().

Here is the call graph for this function:



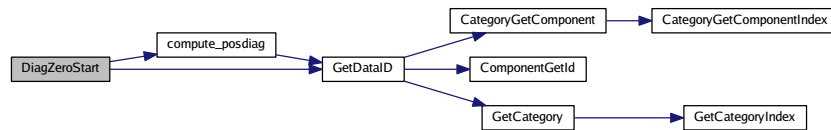
17.28.4.7 static PetscErrorCode DiagZeroStart (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 543 of file structure.c.

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

Referenced by RegisterStructureModules().

Here is the call graph for this function:



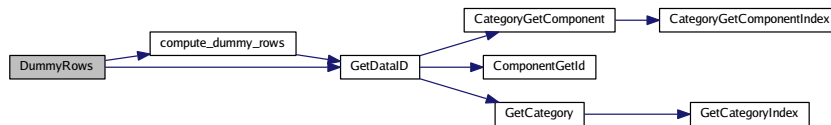
**17.28.4.8 static PetscErrorCode DummyRows (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

Definition at line 465 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:



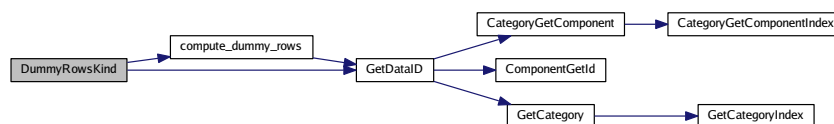
**17.28.4.9 static PetscErrorCode DummyRowsKind (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]**

Definition at line 442 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:



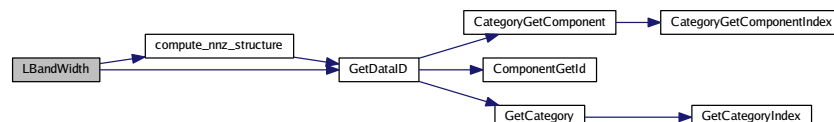
17.28.4.10 static PetscErrorCode LBandWidth (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 249 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:



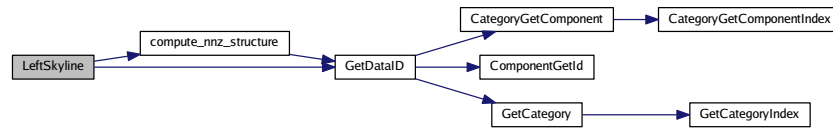
17.28.4.11 static PetscErrorCode LeftSkyline (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 295 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:



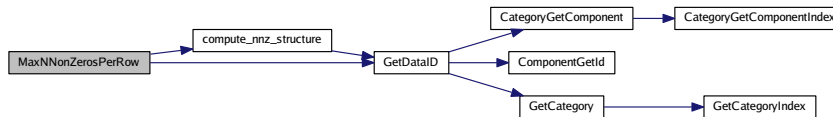
17.28.4.12 static PetscErrorCode MaxNNonZerosPerRow (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 203 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:



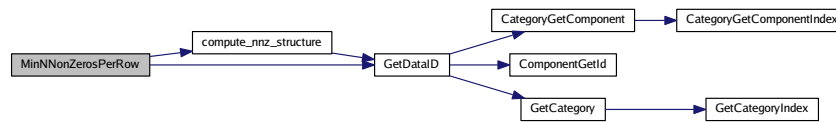
17.28.4.13 static PetscErrorCode MinNNonZerosPerRow (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 226 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:



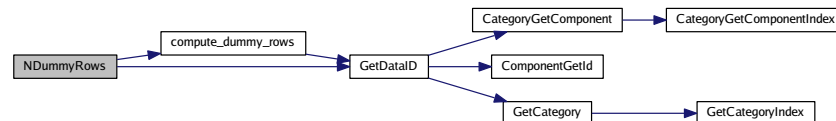
17.28.4.14 static PetscErrorCode NDummyRows (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 419 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:



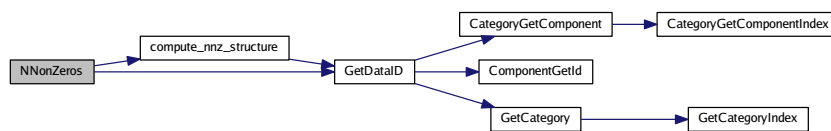
17.28.4.15 static PetscErrorCode NNonZeros (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

Definition at line 180 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:



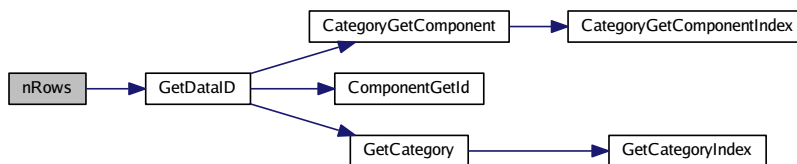
17.28.4.16 static PetscErrorCode nRows (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

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:



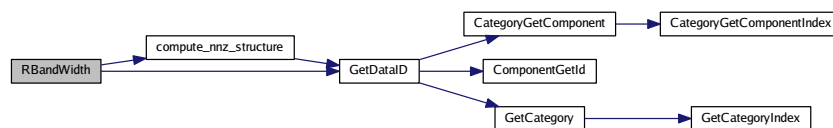
17.28.4.17 static PetscErrorCode RBandWidth (AnaModNumericalProblem prob, AnalysisItem * rv, int * lv, PetscBool * flg) [static]

Definition at line 272 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:



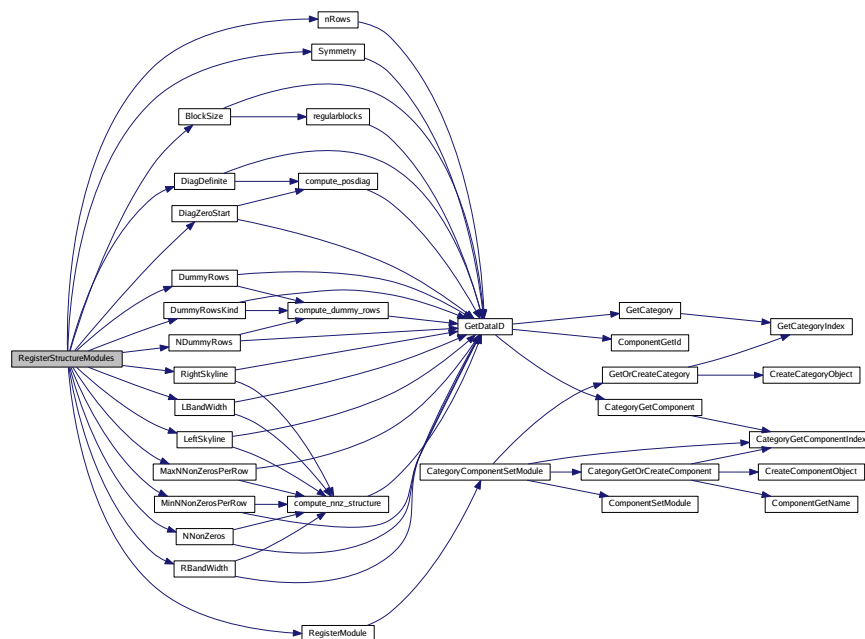
17.28.4.18 PetscErrorCode RegisterStructureModules (void)

Definition at line 671 of file structure.c.

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

Referenced by AnaModRegisterSalsaModules(), and AnaModRegisterStandard- Modules().

Here is the call graph for this function:



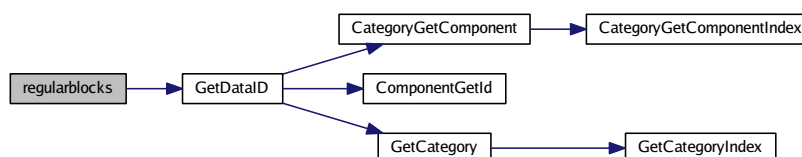
17.28.4.19 static PetscErrorCode regularblocks (AnaModNumericalProblem *prob*) [static]

Definition at line 592 of file structure.c.

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

Referenced by `BlockSize()`.

Here is the call graph for this function:



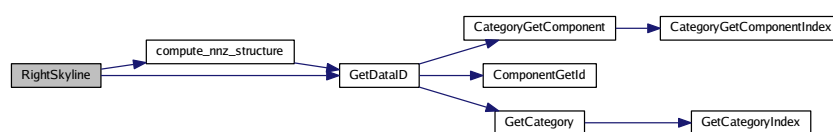
17.28.4.20 **static PetscErrorCode RightSkyline (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 322 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:



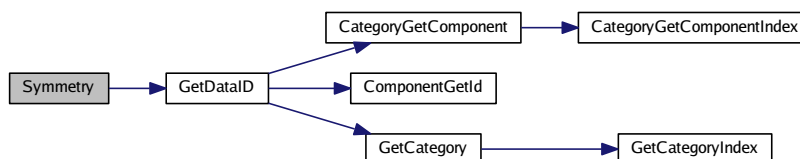
17.28.4.21 **static PetscErrorCode Symmetry (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

Definition at line 78 of file structure.c.

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

Referenced by RegisterStructureModules().

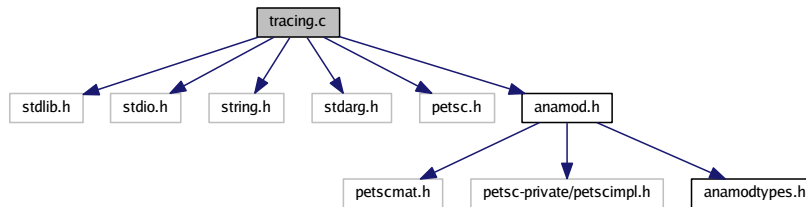
Here is the call graph for this function:



17.29 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 *, const char *, va_list))
- PetscErrorCode [AnaModDeclareTraceContext](#) (void *ctx)
- PetscErrorCode [AnaModSetTraceArrays](#) (PetscBool f)
- PetscErrorCode [AnaModTraceArrays](#) (PetscBool *f)
- PetscErrorCode [AnaModTraceMessage](#) (const char *fmt,...)
- PetscErrorCode [AnaModHasTrace](#) (PetscBool *flg)

Variables

- static PetscErrorCode(* [anamodtrace](#))(void *, const char *fmt, va_list) = NULL
- static size_t [anamodtracectx](#) = (size_t)NULL
- static PetscBool [anamodtracearrays](#) = PETSC_FALSE

17.29.1 Detailed Description

Trace routines for the anamod package.

17.29.2 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](#).

17.29.3 Function Documentation

17.29.3.1 PetscErrorCode AnaModDeclareTraceContext (void * ctx)

Definition at line 69 of file [tracing.c](#).

References [anamodtracectx](#).

17.29.3.2 PetscErrorCode AnaModDeclareTraceFunction (PetscErrorCode(*)(void *, const 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](#).

17.29.3.3 PetscErrorCode AnaModHasTrace (PetscBool * 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 `ComputeOrRetrieveQuantity()`.

17.29.3.4 `PetscErrorCode AnaModSetTraceArrays (PetscBool f)`

Definition at line 82 of file tracing.c.

References `anamodtracearrays`.

17.29.3.5 `PetscErrorCode AnaModTraceArrays (PetscBool * f)`

Definition at line 95 of file tracing.c.

References `anamodtracearrays`.

Referenced by `ComputeOrRetrieveQuantity()`, and `ReportAnamodContent()`.

17.29.3.6 `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()`, and `ComputeOrRetrieveQuantity()`.

17.29.4 Variable Documentation

17.29.4.1 `PetscErrorCode(* anamodtrace)(void *, const char *fmt, va_list) = NULL` [static]

Definition at line 23 of file tracing.c.

Referenced by `AnaModDeclareTraceFunction()`, `AnaModHasTrace()`, and `AnaModTraceMessage()`.

17.29.4.2 `PetscBool anamodtracearrays = PETSC_FALSE` [static]

Definition at line 25 of file tracing.c.

Referenced by `AnaModSetTraceArrays()`, and `AnaModTraceArrays()`.

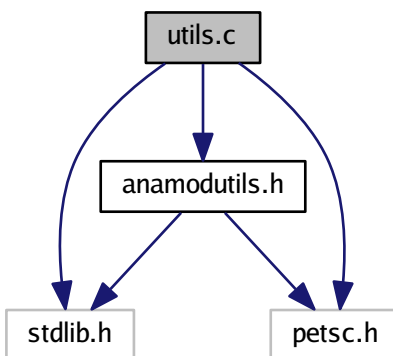
17.29.4.3 `size_t anamodtracectx = (size_t)NULL` [static]

Definition at line 24 of file tracing.c.

Referenced by `AnaModDeclareTraceContext()`, and `AnaModTraceMessage()`.

17.30 **utils.c** File Reference

```
#include <stdlib.h> #include "petsc.h" #include "anamodutils.h"
h" Include dependency graph for utils.c:
```



Functions

- int [MPIAllGatherIntV](#) (int *array, int n, int **Array, int *N, MPI_Comm comm)

17.30.1 Detailed Description

Definition in file [utils.c](#).

17.30.2 Function Documentation

17.30.2.1 int `MPIAllGatherIntV` (int * array, int n, int ** Array, int * N, MPI_Comm comm)

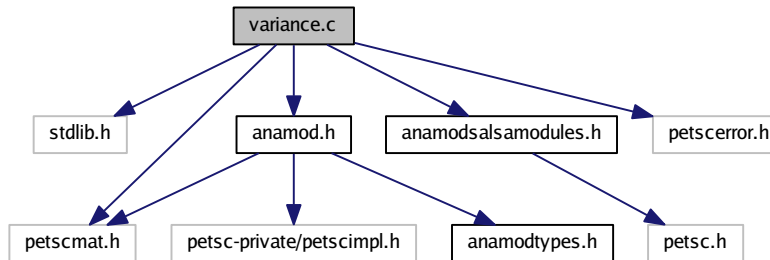
Definition at line 8 of file `utils.c`.

Referenced by `GetUpBiDiagSplits()`, and `MatSplitPoints()`.

17.31 **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, PetscBool *flg)
- static PetscErrorCode [ColVariability](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [ComputeDiagonal](#) (Mat A)
- static PetscErrorCode [DiagonalAverage](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DiagonalVariance](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- static PetscErrorCode [DiagonalSign](#) (AnaModNumericalProblem prob, [AnalysisItem](#) *rv, int *lv, PetscBool *flg)
- PetscErrorCode [RegisterVarianceModules](#) ()
- PetscErrorCode [DeRegisterVarianceModules](#) (void)

17.31.1 Detailed Description

Various estimates of how ‘wild’ a matrix is.

17.31.2 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.

17.31.3 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 [Diagonal-Average\(\)](#)
- "diagonal-variance" : standard deviation of diagonal average, by [Diagonal-Variance\(\)](#)
- "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](#).

17.31.4 Function Documentation

17.31.4.1 static PetscErrorCode ColVariability (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

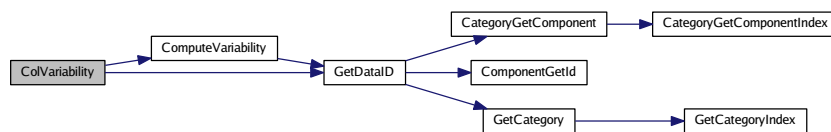
This routine relies on a call to [ComputeVariability\(\)](#)

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:



17.31.4.2 static PetscErrorCode ComputeDiagonal (Mat A) [static]

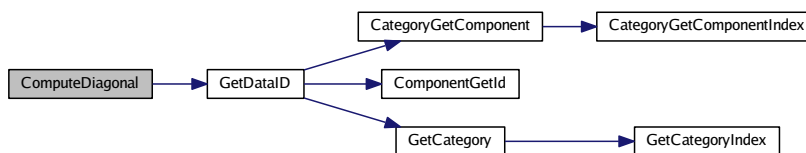
This is a computational routine; it computes the value of several modules

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:



17.31.4.3 static PetscErrorCode ComputeVariability (Mat A) [static]

Variability in rows and columns.

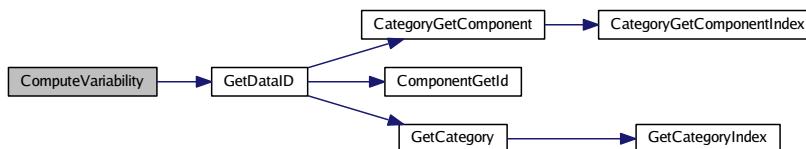
This is a computational routine.

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:



17.31.4.4 PetscErrorCode DeRegisterVarianceModules (void)

Definition at line 333 of file variance.c.

Referenced by AnaModDeregisterSalsaModules().

17.31.4.5 static PetscErrorCode DiagonalAverage (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

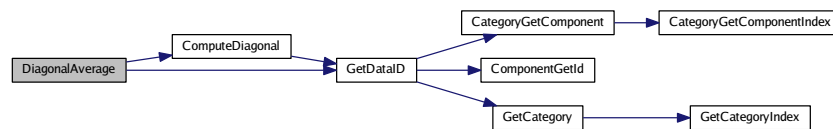
This routine relies on a call to [ComputeDiagonal\(\)](#)

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:



17.31.4.6 static PetscErrorCode DiagonalSign (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*) [static]

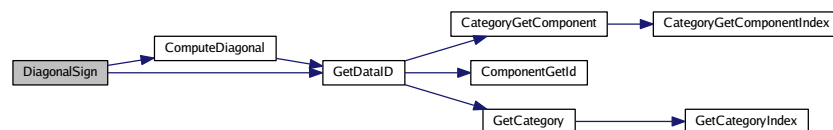
This routine relies on a call to [ComputeDiagonal\(\)](#)

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:



17.31.4.7 **static PetscErrorCode DiagonalVariance (AnaModNumericalProblem *prob*, AnalysisItem * *rv*, int * *lv*, PetscBool * *flg*)** [static]

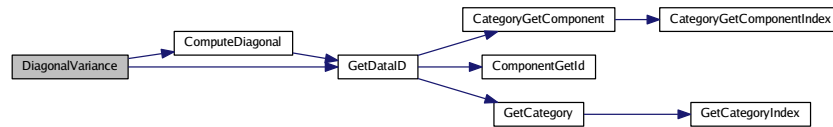
This routine relies on a call to [ComputeDiagonal\(\)](#)

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:



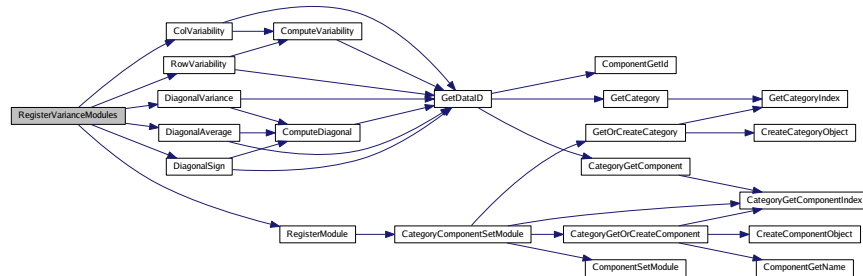
17.31.4.8 **PetscErrorCode RegisterVarianceModules (void)**

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:



17.31.4.9 static PetscErrorCode RowVariability (AnaModNumericalProblem *prob*,
AnalysisItem * *rv*, int * *lv*, PetscBool * *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:

