edos

0.1

Generated by Doxygen 1.8.6

Mon Nov 3 2014 16:03:17

Contents

Chapter 1

Documentation.

1.1 Introduction

The source directories Radau5-NG, Rock4-NG, Rock4-L, Rodas-NG, SDIRKL and SymplecticRK contains sources of different Runge-Kutta methods written in C++.

Radau5-NG is a rewritting of Hairer and Wanner code, optimized, but with some

restrictions (see documentation).

Rock4-NG is a rewritting of A. Abdulle code, avoiding vector copies.

Rock4-L is an adaptation of Rock4 method ro linear (actually affine)

problems $du/dt = Au + B.$, with A linear and B independent of t.

Rodas-NG is a rewritting of Hairer and Wanner code, optimized, but with some

restrictions (see documentation).

SDIRKL is a generic implementation of SDIRK methods in the linear (affine) case. Thus, linear systems must be solved.

SymplecticRK is the (the family of) symplectic Gauss methods.

See documentation of individual programs.

1.2 General considerations about implementation.

All is written in C++, and some program must be used with recent compilers (c++-11 norm). Everuthing has been tested with g++, icc and clang compilers.

This is a library of templates, no intermediate binary is created.

For all these programs, a class must be created wich describes the program to be solved.

1.2.1 Directory structure:

common/include contains all the classes, methods and functions common to the different programs.

for each program, the corresponding directory contains an **include** directory, which stores the different classes of the program and **test** directories.

1.2.1.1 Test directories:

Each **test** directory contains a main program, all the necessary classes, and a cmake file. To compile and test the program:

Adapt CMake file if necessary.

cd Build directory

cmake .. (possibly with an option to determine the compiler you want to use); see the Cmake files provided.

make a **./run** file is created.

run to execute the program.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 odes Namespace Reference

Compute all matrix operations.

Classes

- class fortranArray

  Array of double.
- class fortranComplexArray

  Array of complex..
- class fortranRectangularArray

  Banded Array of double.
- class fortranRectangularComplexArray

  Banded Array of complex..
- class fortranVector

  Vector class (of doubles).
- class fortranVectorF

  Vector class (of doubles).
- class logger

  For logging events.
- class Matrices

  full matrices.
- class Matrices< false, false, n, nsub, nsup >
- class Matrices< true, true, n, nsub, nsup >

  Full matrices, Hessenberg=true.
- class Matrices< false, true, n, nsub, nsup >

  Banded matrices, Hessenberg=true: error.
- struct Matrixtype
- class Radau5cc

  A C++ implementation of Radau5.

Enumerations

- enum eventType {
  success =0, rejectedStep, NewtonFailed, NewtonWillNotConverge,
  changedH, all }

Functions

- void dgetrf_ (int ∗n, int ∗m, double ∗a, int ∗lda, int ∗ipiv, int ∗info)

  prototypes for lapack routines.
- void zgetrf_ (int ∗n, int ∗m, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void dgetrs_ (const char ∗s, int ∗N, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void zgetrs_ (const char ∗s, int ∗N, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void dgbtrf_ (int ∗n, int ∗m, int ∗k1, int ∗k2, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void zgbtrf_ (int ∗n, int ∗m, int ∗k1, int ∗k2, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void dgbtrs_ (const char ∗s, int ∗N, int ∗k1, int ∗k2, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void zgbtrs_ (const char ∗s, int ∗N, int ∗k1, int ∗k2, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void dlarnv_ (int ∗idist, int iseed[], int ∗n, double ∗x)
- void dgehrd_ (int ∗n, int ∗ilo, int ∗ihi, double ∗a, int ∗lda, double tau[], double work[], int ∗lwork, int ∗info)
- void dorghr_ (int ∗n, int ∗ilo, int ∗ihi, double ∗a, int ∗lda, double tau[], double work[], int ∗lwork, int ∗info)

6.1.1 Detailed Description

Compute all matrix operations. determine the type of Jacobian matrices.

Compute all matrix operations: -compute Jacobian matrix -....... Factorisations.

- solve systems. Template for full matrix, with specializations for banded matrices and full matrix, transformed in Hessenberg form. Trying to instanciate a banded matrix matrix in Hessenberg form is impossible and should be capturated with "compat.hpp"

Helper class, used at compile time to determine the type of Jacobian matrices.

6.1.2 Enumeration Type Documentation

6.1.2.1 enum odes::eventType

Enumerator

success

rejectedStep

NewtonFailed

NewtonWillNotConverge

changedH

all

6.1.3 Function Documentation

6.1.3.1 void odes::dgbtrf_ (int ∗ *n,* int ∗ *m,* int ∗ *k1,* int ∗ *k2,* double ∗ *a,* int ∗ *lda,* int ∗ *ipiv,* int ∗ *info*)

6.1.3.2 void odes::dgbtrs_ (const char ∗ *s,* int ∗ *N,* int ∗ *k1,* int ∗ *k2,* int ∗ *NRHS,* double ∗ *A,* int ∗ *LDA,* int ∗ *IPIV,* double ∗ *B,* int ∗ *LDB,* int ∗ *INFO*)

6.1.3.3 void odes::dgehrd_ (int ∗ *n,* int ∗ *ilo,* int ∗ *ihi,* double ∗ *a,* int ∗ *lda,* double *tau[ ],* double *work[ ],* int ∗ *lwork,* int ∗ *info*)

6.1.3.4 void odes::dgetrf_ (int ∗ *n,* int ∗ *m,* double ∗ *a,* int ∗ *lda,* int ∗ *ipiv,* int ∗ *info*)

prototypes for lapack routines.

6.1.3.5 void odes::dgetrs_ (const char ∗ *s*, int ∗ *N*, int ∗ *NRHS*, double ∗ *A*, int ∗ *LDA*, int ∗ *IPIV*, double ∗ *B*, int ∗ *LDB*, int ∗ *INFO*)

6.1.3.6 void odes::dlarnv_ (int ∗ *idist*, int *iseed[ ]*, int ∗ *n*, double ∗ *x*)

6.1.3.7 void odes::dorghr_ (int ∗ *n*, int ∗ *ilo*, int ∗ *ihi*, double ∗ *a*, int ∗ *lda*, double *tau[ ]*, double *work[ ]*, int ∗ *lwork*, int ∗ *info*)

6.1.3.8 void odes::zgbtrf_ (int ∗ *n*, int ∗ *m*, int ∗ *k1*, int ∗ *k2*, double ∗ *a*, int ∗ *lda*, int ∗ *ipiv*, int ∗ *info*)

6.1.3.9 void odes::zgbtrs_ (const char ∗ *s*, int ∗ *N*, int ∗ *k1*, int ∗ *k2*, int ∗ *NRHS*, double ∗ *A*, int ∗ *LDA*, int ∗ *IPIV*, double ∗ *B*, int ∗ *LDB*, int ∗ *INFO*)

6.1.3.10 void odes::zgetrf_ (int ∗ *n*, int ∗ *m*, double ∗ *a*, int ∗ *lda*, int ∗ *ipiv*, int ∗ *info*)

6.1.3.11 void odes::zgetrs_ (const char ∗ *s*, int ∗ *N*, int ∗ *NRHS*, double ∗ *A*, int ∗ *LDA*, int ∗ *IPIV*, double ∗ *B*, int ∗ *LDB*, int ∗ *INFO*)

Chapter 7

Class Documentation

7.1 compat< full, Hessenberg > Struct Template Reference

```
#include <compat.hpp>
```

Public Member Functions

- ct_assert (full||(!full &&!Hessenberg))

7.1.1 Member Function Documentation

7.1.1.1 template<bool full, bool Hessenberg> compat< full, Hessenberg >::ct_assert (full|| *!full &&!Hessenberg*)

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

- common/include/compat.hpp

7.2 odes::logger::event Struct Reference

Public Member Functions

- event (double _time, int _step, eventType _TheEvent, double _value=0.0)

Public Attributes

- double time
- double value
- int step
- eventType TheEvent

7.2.1 Constructor & Destructor Documentation

7.2.1.1 odes::logger::event::event (double *_time,* int *_step,* eventType *_TheEvent,* double *_value* = `0.0`) `[inline]`

7.2.2 Member Data Documentation

7.2.2.1 int odes::logger::event::step

7.2.2.2 eventType odes::logger::event::TheEvent

7.2.2.3 double odes::logger::event::time

7.2.2.4 double odes::logger::event::value

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

- common/include/logger.hpp

7.3 odes::fortranArray< n > Class Template Reference

Array of double.

`#include <fortranArray.hpp>`

Public Member Functions

- fortranArray ()

  constructor
- fortranArray (const fortranArray< n > &AA)

  constructor (copy).
- ∼fortranArray ()

  destructor
- double operator() (int i, int j) const
- double & operator() (int i, int j)
- void equal_minus (fortranArray< n > &X)
- void addDiag (double v)
- double ∗ operator& ()

  return adress of double array.
- void print (std::string s="") const

  print

Private Attributes

- double ∗ x

Static Private Attributes

- static const int n2 =n∗n

7.3.1 Detailed Description

template<int n>class odes::fortranArray< n >

Array of double.

Array of double, of fixed size n.n, fortran indexing.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 template<int n> odes::fortranArray< n >::fortranArray ( ) [inline]

constructor

7.3.2.2 template<int n> odes::fortranArray< n >::fortranArray (const fortranArray< n > & *AA*) [inline]

constructor (copy).

define the ivdep pragma for different compilers.

7.3.2.3 template<int n> odes::fortranArray< n >::∼fortranArray ( ) [inline]

destructor

7.3.3 Member Function Documentation

7.3.3.1 template<int n> void odes::fortranArray< n >::addDiag (double *v*) [inline]

add a value to diagonal.

Parameters

v	the value.

define the ivdep pragma for different compilers.

7.3.3.2 template<int n> void odes::fortranArray< n >::equal_minus (fortranArray< n > & *X*) [inline]

this = -array.

Parameters

X	the array.

define the ivdep pragma for different compilers.

7.3.3.3 template<int n> double∗ odes::fortranArray< n >::operator& ( ) [inline]

return adress of double array.

7.3.3.4 template<int n> double odes::fortranArray< n >::operator() (int *i*, int *j*) const [inline]

indexing

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.3.3.5 template<int n> double& odes::fortranArray< n >::operator() (int *i*, int *j*) [inline]

indexing.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).
returns a reference, possible leftvalue.

7.3.3.6 template<int n> void odes::fortranArray< n >::print (std::string *s* = "") const [inline]

print

7.3.4 Member Data Documentation

7.3.4.1 template<int n> const int odes::fortranArray< n >::n2 =n∗n [static], [private]

7.3.4.2 template<int n> double∗ odes::fortranArray< n >::x [private]

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

- common/include/fortranArray.hpp

7.4 odes::fortranComplexArray< n > Class Template Reference

Array of complex..

`#include <fortranComplexArray.hpp>`

Public Member Functions

- fortranComplexArray ()
  contructor
- ∼fortranComplexArray ()
  destructor
- void set (int i, int j, double RealPart, double ImagPart=0)
- double & Re (int i, int j)
- double & Im (int i, int j)
- double ∗ operator& ()
  return adress of double array.
- void print (std::string s="")
  print

Private Attributes

- double ∗ x

Static Private Attributes

- static const int d2 =2∗n
- static const int size =d2∗n

7.4.1 Detailed Description

template<int n>class odes::fortranComplexArray< n >

Array of complex..

Array of Complex, of fixed size n.n, fortran indexing. Actually, we do not make use of complex<double> class (we just want to store coefficients for lapack zge∗ routines).

7.4.2 Constructor & Destructor Documentation

7.4.2.1 template<int n> **odes::fortranComplexArray< n >::fortranComplexArray** ( ) `[inline]`

contructor

7.4.2.2 template<int n> **odes::fortranComplexArray< n >::∼fortranComplexArray** ( ) `[inline]`

destructor

7.4.3 Member Function Documentation

7.4.3.1 template<int n> double& **odes::fortranComplexArray< n >::Im** (int *i*, int *j*) `[inline]`

Indexing, return a reference to "Imaginary" part.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.4.3.2 template<int n> double∗ **odes::fortranComplexArray< n >::operator&** ( ) `[inline]`

return adress of double array.

7.4.3.3 template<int n> void **odes::fortranComplexArray< n >::print** (std::string *s* = "") `[inline]`

print

7.4.3.4 template<int n> double& **odes::fortranComplexArray< n >::Re** (int *i*, int *j*) `[inline]`

Indexing, return a reference to "real" part.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.4.3.5 template<int n> void odes::fortranComplexArray< n >::set (int *i*, int *j*, double *RealPart*, double *ImagPart* = 0) [inline]

7.4.4 Member Data Documentation

7.4.4.1 template<int n> const int odes::fortranComplexArray< n >::d2 =2∗n [static], [private]

7.4.4.2 template<int n> const int odes::fortranComplexArray< n >::size =d2∗n [static], [private]

7.4.4.3 template<int n> double∗ odes::fortranComplexArray< n >::x [private]

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

- common/include/fortranComplexArray.hpp

7.5 odes::fortranRectangularArray< n, kl, ku > Class Template Reference

Banded Array of double.

`#include <fortranRectangularArray.hpp>`

Public Member Functions

- fortranRectangularArray ()

  constructor
- ∼fortranRectangularArray ()

  destructor
- double operator() (int i, int j) const
- double & operator() (int i, int j)
- void equal_minus (const fortranRectangularArray< n, kl, ku > &X)
- void addDiag (double v)
- double ∗ operator& ()

  return adress of double array.
- void print (std::string s="")

  print

Private Attributes

- double ∗ x

Static Private Attributes

- static const int ldab =2∗kl+ku+1
- static const int klku =kl+ku
- static const int size =ldab∗n
- static const int l1 =ldab-1
- static const int cc =klku-ldab

7.5.1 Detailed Description

template<int n, int kl, int ku>class odes::fortranRectangularArray< n, kl, ku >

Banded Array of double.

Banded Arrray of double, of fixed size n, with kl subdiagonals and ku superdiagonals, fortran indexing, like in lapack routines (see dgbtrf for example).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 template<int n, int kl, int ku> odes::fortranRectangularArray< n, kl, ku >::fortranRectangularArray ( ) [inline]

constructor

7.5.2.2 template<int n, int kl, int ku> odes::fortranRectangularArray< n, kl, ku >::∼fortranRectangularArray ( ) [inline]

destructor

7.5.3 Member Function Documentation

7.5.3.1 template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::addDiag (double *v*) [inline]

add a value to diagonal.

Parameters

v	the value.

define the ivdep pragma for different compilers.

7.5.3.2 template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::equal_minus (const fortranRectangularArray< n, kl, ku > & *X*) [inline]

this = -X.

Parameters

X	the array.

define the ivdep pragma for different compilers.

7.5.3.3 template<int n, int kl, int ku> double∗ odes::fortranRectangularArray< n, kl, ku >::operator& ( ) [inline]

return adress of double array.

7.5.3.4 template<int n, int kl, int ku> double odes::fortranRectangularArray< n, kl, ku >::operator() (int *i*, int *j*) const [inline]

indexing

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.5.3.5 template<int n, int kl, int ku> double& odes::fortranRectangularArray< n, kl, ku >::operator() (int *i*, int *j*) `[inline]`

indexing.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).
returns a reference, possible leftvalue.

7.5.3.6 template<int n, int kl, int ku> void odes::fortranRectangularArray< n, kl, ku >::print (std::string *s* = " ") `[inline]`

print

7.5.4 Member Data Documentation

7.5.4.1 template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::cc =klku-ldab `[static]`, `[private]`

7.5.4.2 template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::klku =kl+ku `[static]`, `[private]`

7.5.4.3 template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::l1 =ldab-1 `[static]`, `[private]`

7.5.4.4 template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::ldab =2*kl+ku+1 `[static]`, `[private]`

7.5.4.5 template<int n, int kl, int ku> const int odes::fortranRectangularArray< n, kl, ku >::size =ldab*n `[static]`, `[private]`

7.5.4.6 template<int n, int kl, int ku> double* odes::fortranRectangularArray< n, kl, ku >::x `[private]`

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

- common/include/fortranRectangularArray.hpp

7.6 odes::fortranRectangularComplexArray< n, kl, ku > Class Template Reference

Banded Array of complex..

```
#include <fortranRectangularComplexArray.hpp>
```

Public Member Functions

- fortranRectangularComplexArray ()

  contructor
- ∼fortranRectangularComplexArray ()

  destructor
- void set (int i, int j, double RealPart, double ImagPart=0)
- double & Re (int i, int j)
- double & Im (int i, int j)
- double ∗ operator& ()

  return adress of double array.
- void print (std::string s="")

  print

Private Attributes

- double ∗ x

Static Private Attributes

- static const int ldab =2∗kl+ku+1
- static const int klku =kl+ku
- static const int size =2∗ldab∗n
- static const int l1 =ldab-1
- static const int cc =klku-ldab

7.6.1 Detailed Description

template<int n, int kl, int ku>class odes::fortranRectangularComplexArray< n, kl, ku >

Banded Array of complex..

Banded Array of Complex, of fixed size n, with kl subdiagonals and ku superdiagonals, fortran indexing, like in lapack routines (see zgbtrf for axample). Actually, we do not make use of complex<double> class (we just want to store coefficients for lapack zge∗ routines).

7.6.2 Constructor & Destructor Documentation

7.6.2.1 template<int n, int kl, int ku> odes::fortranRectangularComplexArray< n, kl, ku >::fortranRectangularComplexArray ( ) [inline]

contructor

7.6.2.2 template<int n, int kl, int ku> odes::fortranRectangularComplexArray< n, kl, ku >::∼fortranRectangularComplexArray ( ) [inline]

destructor

7.6.3 Member Function Documentation

7.6.3.1 template<int n, int kl, int ku> double& odes::fortranRectangularComplexArray< n, kl, ku >::Im (int *i*, int *j*) [inline]

Indexing, return a reference to "Imaginary" part.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.6.3.2 template<int n, int kl, int ku> double* odes::fortranRectangularComplexArray< n, kl, ku >::operator& ( ) [inline]

return adress of double array.

7.6.3.3 template<int n, int kl, int ku> void odes::fortranRectangularComplexArray< n, kl, ku >::print (std::string *s* = "") [inline]

print

7.6.3.4 template<int n, int kl, int ku> double& odes::fortranRectangularComplexArray< n, kl, ku >::Re (int *i*, int *j*) [inline]

Indexing, return a reference to "real" part.

Parameters

i	
j	

Note

fortran indexing (row major, i>=1, j>=1,i<=n, j<=n).

7.6.3.5 template<int n, int kl, int ku> void odes::fortranRectangularComplexArray< n, kl, ku >::set (int *i*, int *j*, double *RealPart*, double *ImagPart* = 0) [inline]

7.6.4 Member Data Documentation

7.6.4.1 template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::cc =klku-ldab [static], [private]

7.6.4.2 template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::klku =kl+ku [static], [private]

7.6.4.3 template<int n, int kl, int ku> const int odes::fortranRectangularComplexArray< n, kl, ku >::l1 =ldab-1 [static], [private]

7.6.4.4 template<int n, int kl, int ku> const int **odes::fortranRectangularComplexArray**< n, kl, ku >::ldab =2∗kl+ku+1 [static], [private]

7.6.4.5 template<int n, int kl, int ku> const int **odes::fortranRectangularComplexArray**< n, kl, ku >::size =2∗**ldab**∗n [static], [private]

7.6.4.6 template<int n, int kl, int ku> double∗ **odes::fortranRectangularComplexArray**< n, kl, ku >::x [private]

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

- common/include/fortranRectangularComplexArray.hpp

7.7 odes::fortranVector Class Reference

Vector class (of doubles).

`#include <fortranVector.hpp>`

Public Member Functions

- fortranVector ()
- fortranVector (int k)
- fortranVector (double ∗_x, int _size)
- ∼fortranVector ()

  destructor
- double operator() (int i) const
- double & operator() (int i)
- double ∗ operator& () const

  return a pointer to the vector of double.
- void operator= (fortranVector V)
- void operator+= (fortranVector &F)
- void setsize (unsigned int _size)
- int get_size () const

  get the size
- void print (string s="") const

  print

Protected Attributes

- double ∗ x
- int size
- bool deletable

Friends

- template<int k>
  class fortranArray
- template<int k>
  class fortranVectorF

7.7.1 Detailed Description

Vector class (of doubles).

Vector class (of doubles). We want to use fortran indexing (from 1 to...): this is just a wrapper around an array of doubles. Note: this is not a very *safe* class!. (we are adult programmers :-)).

7.7.2 Constructor & Destructor Documentation

7.7.2.1 odes::fortranVector::fortranVector ( ) `[inline]`

constructor

Note

> we construct an empty class, not usable.

7.7.2.2 odes::fortranVector::fortranVector (int *k*) `[inline]`

constructor

Parameters

k	size of the vector

7.7.2.3 odes::fortranVector::fortranVector (double ∗ *_x*, int *_size*) `[inline]`

build a fortranVector.

Parameters

_x	a vector of double
_size	size of the vector

Note

> we "adopt" a vector, to be able to use fortran indexing; this is not safe!

7.7.2.4 odes::fortranVector::∼fortranVector ( ) `[inline]`

destructor

7.7.3 Member Function Documentation

7.7.3.1 int odes::fortranVector::get_size ( ) const `[inline]`

get the size

7.7.3.2 double∗ odes::fortranVector::operator& ( ) const `[inline]`

return a pointer to the vector of double.

7.7.3.3 double odes::fortranVector::operator() (int *i*) const `[inline]`

operator (): indexation

Parameters

i	index (between 1 and n (included)).

7.7.3.4 double& odes::fortranVector::operator() (int *i*) [inline]

operator (): indexation

Parameters

i	index (between 1 and n (included)).

Note

returns a reference (can be used as lvalue).

7.7.3.5 void odes::fortranVector::operator+= (fortranVector & *F*) [inline]

operator +=

Parameters

F	to be added.

define the ivdep pragma for different compilers.

7.7.3.6 void odes::fortranVector::operator= (fortranVector *V*) [inline]

copy constructor

Parameters

V	object to copy

Note

nothing is checked! not safe.

define the ivdep pragma for different compilers.

7.7.3.7 void odes::fortranVector::print (string *s* = "") const [inline]

print

7.7.3.8 void odes::fortranVector::setsize (unsigned int *_size*) [inline]

set size _param _size the size.

7.7.4 Friends And Related Function Documentation

7.7.4.1 template<int k> friend class fortranArray [friend]

7.7.4.2 template<int k> friend class fortranVectorF [friend]

7.7.5 Member Data Documentation

7.7.5.1 bool odes::fortranVector::deletable [protected]

7.7.5.2 int odes::fortranVector::size [protected]

7.7.5.3 double∗ odes::fortranVector::x [protected]

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

- common/include/fortranVector.hpp

7.8 odes::fortranVectorF< n > Class Template Reference

Vector class (of doubles).

`#include <fortranVector.hpp>`

Public Member Functions

- fortranVectorF ()

  constructor. Object created is usable.
- ∼fortranVectorF ()

  destructor
- void switchadr (fortranVectorF< n > &v)
- void operator= (double v)
- void operator= (double tab[ ])
- void setsum2 (fortranVector &X, fortranVector &Y)
- void operator= (fortranVectorF< n > &X)
- void operator+= (fortranVectorF< n > &F)
- void sum (fortranVectorF< n > &X, fortranVectorF< n > &Y)
- void sum (fortranVectorF< n > &X, fortranVector &Y)
- void a_Y (double a, const fortranVectorF< n > &Y)
- void x_lc2 (fortranVectorF< n > &X, double a1, fortranVectorF< n > &Y1, double a2, fortranVectorF< n > &Y2)
- void lc2 (double a, fortranVectorF< n > &x1, double b, fortranVectorF< n > &x2)
- void lc3 (double a, fortranVectorF< n > &x1, double b, fortranVectorF< n > &x2, double c, fortranVectorF< n > &x3)
- void lc4 (double a, fortranVectorF< n > &x1, double b, fortranVectorF< n > &x2, double c, fortranVectorF< n > &x3, double d, fortranVectorF< n > &x4)
- void lc5 (double a, fortranVectorF< n > &x1, double b, fortranVectorF< n > &x2, double c, fortranVectorF< n > &x3, double d, fortranVectorF< n > &x4, double e, fortranVectorF< n > &x5)
- void x_lc3 (fortranVectorF< n > &X, double a1, fortranVectorF< n > &Y1, double a2, fortranVectorF< n > &Y2, double a3, fortranVectorF< n > &Y3)
- void x_lc4 (fortranVectorF< n > &X, double a1, fortranVectorF< n > &Y1, double a2, fortranVectorF< n > &Y2, double a3, fortranVectorF< n > &Y3, double a4, fortranVectorF< n > &Y4)
- void a_x_plus_y (double a, fortranVectorF< n > &X, fortranVectorF< n > &Y)
- void put (double _x[ ])
- void print (string s="") const

  print
- double operator() (int i) const
- double & operator() (int i)
- double ∗ operator& () const

  return a pointer to the vector of double.
- void operator+= (fortranVector &F)

- void setsize (unsigned int _size)
- int get_size () const

  get the size

Protected Attributes

- double $*$ x
- int size
- bool deletable

Private Attributes

- double $*$ y

7.8.1 Detailed Description

template$<$int n$>$class odes::fortranVectorF$<$ n $>$

Vector class (of doubles).

Vector class (of doubles) OF FIXED SIZE n. We want to use fortran indexing (from 1 to...): this is just a wrapper around an array of doubles. Note: this is not a very safe class!.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 template$<$int n$>$ odes::fortranVectorF$<$ n $>$::fortranVectorF ( ) `[inline]`

constructor. Object created is usable.

7.8.2.2 template$<$int n$>$ odes::fortranVectorF$<$ n $>$::~fortranVectorF ( ) `[inline]`

destructor

7.8.3 Member Function Documentation

7.8.3.1 template$<$int n$>$ void odes::fortranVectorF$<$ n $>$::a_x_plus_y (double *a*, fortranVectorF$<$ n $>$ & *X*, fortranVectorF$<$ n $>$ & *Y*) `[inline]`

Parameters

a	coefficient
X	fortranVectorF<n>
Y	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.2 template$<$int n$>$ void odes::fortranVectorF$<$ n $>$::a_Y (double *a*, const fortranVectorF$<$ n $>$ & *Y*) `[inline]`

this= a$*$Y

Parameters

a	a double
Y	fortranVector<n>

define the ivdep pragma for different compilers.

7.8.3.3 int odes::fortranVector::get_size ( ) const [inline], [inherited]

get the size

7.8.3.4 template<int n> void odes::fortranVectorF< n >::lc2 (double *a*, fortranVectorF< n > & *x1*, double *b*, fortranVectorF< n > & *x2*) [inline]

this = linear combination of 2 fortranVectorF<n>

Parameters

a	coefficient
x1	fortranVectorF<n>
b	coefficient
x2	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.5 template<int n> void odes::fortranVectorF< n >::lc3 (double *a*, fortranVectorF< n > & *x1*, double *b*, fortranVectorF< n > & *x2*, double *c*, fortranVectorF< n > & *x3*) [inline]

this = linear combination of 3 fortranVectorF<n>

Parameters

a	coefficient
x1	fortranVectorF<n>
b	coefficient
x2	fortranVectorF<n>
c	coefficient
x3	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.6 template<int n> void odes::fortranVectorF< n >::lc4 (double *a*, fortranVectorF< n > & *x1*, double *b*, fortranVectorF< n > & *x2*, double *c*, fortranVectorF< n > & *x3*, double *d*, fortranVectorF< n > & *x4*) [inline]

this = linear combination of 4 fortranVectorF<n>

Parameters

a	coefficient
x1	fortranVectorF<n>
b	coefficient
x2	fortranVectorF<n>
c	coefficient
x3	fortranVectorF<n>

d	coefficient
x4	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.7 template<int n> void odes::fortranVectorF< n >::lc5 (double *a*, fortranVectorF< n > & *x1*, double *b*, fortranVectorF< n > & *x2*, double *c*, fortranVectorF< n > & *x3*, double *d*, fortranVectorF< n > & *x4*, double *e*, fortranVectorF< n > & *x5*) `[inline]`

this = linear combination of 5 fortranVectorF<n>

Parameters

a	coefficient
x1	fortranVectorF<n>
b	coefficient
x2	fortranVectorF<n>
c	coefficient
x3	fortranVectorF<n>
d	coefficient
x4	fortranVectorF<n>
e	coefficient
x5	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.8 double∗ odes::fortranVector::operator& ( ) const `[inline]`, `[inherited]`

return a pointer to the vector of double.

7.8.3.9 double odes::fortranVector::operator() (int *i*) const `[inline]`, `[inherited]`

operator (): indexation

Parameters

i	index (between 1 and n (included)).

7.8.3.10 double& odes::fortranVector::operator() (int *i*) `[inline]`, `[inherited]`

operator (): indexation

Parameters

i	index (between 1 and n (included)).

Note

> returns a reference (can be used as lvalue).

7.8.3.11 void odes::fortranVector::operator+= (fortranVector & *F*) `[inline]`, `[inherited]`

operator +=

Parameters

F	to be added.

define the ivdep pragma for different compilers.

7.8.3.12 template<int n> void odes::fortranVectorF< n >::operator+= (fortranVectorF< n > & *F*) [inline]

operator +=

Parameters

F	to be added.

define the ivdep pragma for different compilers.

7.8.3.13 template<int n> void odes::fortranVectorF< n >::operator= (double *v*) [inline]

put the same value everywhere.

Parameters

v	the value.

define the ivdep pragma for different compilers.

7.8.3.14 template<int n> void odes::fortranVectorF< n >::operator= (double *tab[]*) [inline]

copy an array of double.

Parameters

tab[]	the array.

Note

> we do not check the size of tab[].

define the ivdep pragma for different compilers.

7.8.3.15 template<int n> void odes::fortranVectorF< n >::operator= (fortranVectorF< n > & *X*) [inline]

operator=

Parameters

X	∗this=X.

define the ivdep pragma for different compilers.

7.8.3.16 template<int n> void odes::fortranVectorF< n >::print (string *s* = " ") const [inline]

print

7.8.3.17 template<int n> void odes::fortranVectorF< n >::put (double *_x[]*) [inline]

copy in a vector of double.

Parameters

_x[]	the vector (result).

Note

we cannot check for the size of x[].

define the ivdep pragma for different compilers.

7.8.3.18 void odes::fortranVector::setsize (unsigned int *_size*) [inline], [inherited]

set size _param _size the size.

7.8.3.19 template<int n> void odes::fortranVectorF< n >::setsum2 (fortranVector & *X*, fortranVector & *Y*) [inline]

this<- X+Y.

Parameters

X	
Y	

Note

we do not make any check on X and Y

define the ivdep pragma for different compilers.

7.8.3.20 template<int n> void odes::fortranVectorF< n >::sum (fortranVectorF< n > & *X*, fortranVectorF< n > & *Y*) [inline]

this = X +Y.

Parameters

X	fortranVectorF<n>
Y	fortranVectorF<n>

define the ivdep pragma for different compilers.

7.8.3.21 template<int n> void odes::fortranVectorF< n >::sum (fortranVectorF< n > & *X*, fortranVector & *Y*) [inline]

this = X +Y.

Parameters

X	fortranVectorF<n>
Y	fortranVector<n>

define the ivdep pragma for different compilers.

7.8.3.22 template<int n> void odes::fortranVectorF< n >::switchadr (fortranVectorF< n > & *v*) [inline]

swith adress with an other fortranVectorF<n>

Parameters

v	we switch with v.

7.8.3.23 template<int n> void odes::fortranVectorF< n >::x_lc2 (fortranVectorF< n > & *X*, double *a1*, fortranVectorF< n > & *Y1*, double *a2*, fortranVectorF< n > & *Y2*) [inline]

this= X+ a1∗Y1 + a2∗Y2.

Parameters

X	
a1	
Y1	
a2	
Y2	

define the ivdep pragma for different compilers.

7.8.3.24 template<int n> void odes::fortranVectorF< n >::x_lc3 (fortranVectorF< n > & *X*, double *a1*, fortranVectorF< n > & *Y1*, double *a2*, fortranVectorF< n > & *Y2*, double *a3*, fortranVectorF< n > & *Y3*) [inline]

this= X+ a1∗Y1 + a2∗Y2 + a3∗Y3

Parameters

X	
a1	
Y1	
a2	
Y2	
a3	
Y3	

define the ivdep pragma for different compilers.

7.8.3.25 template<int n> void odes::fortranVectorF< n >::x_lc4 (fortranVectorF< n > & *X*, double *a1*, fortranVectorF< n > & *Y1*, double *a2*, fortranVectorF< n > & *Y2*, double *a3*, fortranVectorF< n > & *Y3*, double *a4*, fortranVectorF< n > & *Y4*) [inline]

this= X+ a1∗Y1 + a2∗Y2 + a3∗Y3 + a4∗ Y4

Parameters

X	
a1	
Y1	
a2	
Y2	
a3	
Y3	
a4	
Y4	

define the ivdep pragma for different compilers.

7.8.4 Member Data Documentation

7.8.4.1 bool odes::fortranVector::deletable `[protected]`, `[inherited]`

7.8.4.2 int odes::fortranVector::size `[protected]`, `[inherited]`

7.8.4.3 double∗ odes::fortranVector::x `[protected]`, `[inherited]`

7.8.4.4 template<int n> double∗ odes::fortranVectorF< n >::y `[private]`

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

- common/include/fortranVector.hpp

7.9 GenericException Class Reference

```
#include <GenericException.hpp>
```

Public Member Functions

- GenericException ()

  O argument.
- template<class A >
  GenericException (A x)

  1 argument
- template<class A , class B >
  GenericException (A x, B y)

  2 arguments
- template<class A , class B , class C >
  GenericException (A x, B y, C z)

  3 arguments
- template<class A , class B , class C , class D >
  GenericException (A x, B y, C z, D a)

  4 arguments
- template<class A , class B , class C , class D , class E >
  GenericException (A x, B y, C z, D a, E b)

  5 arguments
- template<class A , class B , class C , class D , class E , class F >
  GenericException (A x, B y, C z, D a, E b, F c)

  6 arguments
- template<class A , class B , class C , class D , class E , class F , class G >
  GenericException (A x, B y, C z, D a, E b, F c, G d)

  7 arguments:
- template<class A , class B , class C , class D , class E , class F , class G , class H >
  GenericException (A x, B y, C z, D a, E b, F c, G d, H e)

  8 arguments:
- template<class A , class B , class C , class D , class E , class F , class G , class H , class I >
  GenericException (A x, B y, C z, D a, E b, F c, G d, H e, I f)

  9 arguments:
- template<class A , class B , class C , class D , class E , class F , class G , class H , class I , class J >
  GenericException (A x, B y, C z, D a, E b, F c, G d, H e, I f, J g)

  10 arguments:

7.9.1 Detailed Description

A simple exception class. Instantiate it with 1 to 6 parameters. All parameters must be printable, and will actually be printed.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 GenericException::GenericException ( ) `[inline]`

O argument.

7.9.2.2 template<class A > GenericException::GenericException (A *x*) `[inline]`

1 argument

7.9.2.3 template<class A , class B > GenericException::GenericException (A *x*, B *y*) `[inline]`

2 arguments

7.9.2.4 template<class A , class B , class C > GenericException::GenericException (A *x*, B *y*, C *z*) `[inline]`

3 arguments

7.9.2.5 template<class A , class B , class C , class D > GenericException::GenericException (A *x*, B *y*, C *z*, D *a*) `[inline]`

4 arguments

7.9.2.6 template<class A , class B , class C , class D , class E > GenericException::GenericException (A *x*, B *y*, C *z*, D *a*, E *b*) `[inline]`

5 arguments

7.9.2.7 template<class A , class B , class C , class D , class E , class F > GenericException::GenericException (A *x*, B *y*, C *z*, D *a*, E *b*, F *c*) `[inline]`

6 arguments

7.9.2.8 template<class A , class B , class C , class D , class E , class F , class G > GenericException::GenericException (A *x*, B *y*, C *z*, D *a*, E *b*, F *c*, G *d*) `[inline]`

7 arguments:

7.9.2.9 template<class A , class B , class C , class D , class E , class F , class G , class H > GenericException::GenericException (A *x*, B *y*, C *z*, D *a*, E *b*, F *c*, G *d*, H *e*) `[inline]`

8 arguments:

7.9.2.10 template<class A , class B , class C , class D , class E , class F , class G , class H , class I > GenericException::GenericException (A *x,* B *y,* C *z,* D *a,* E *b,* F *c,* G *d,* H *e,* I *f*) [inline]

9 arguments:

7.9.2.11 template<class A , class B , class C , class D , class E , class F , class G , class H , class I , class J > GenericException::GenericException (A *x,* B *y,* C *z,* D *a,* E *b,* F *c,* G *d,* H *e,* I *f,* J *g*) [inline]

10 arguments:

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

- common/include/GenericException.hpp

7.10 odes::logger Class Reference

For logging events.

`#include <logger.hpp>`

Classes

- struct event

Public Member Functions

- logger ()

  constructor
- ∼logger ()

  destructor
- void clear ()

  reset.
- void put (double _time, int _step, eventType _TheEvent, double _value)
- void print (eventType E=all)

Private Attributes

- list< event > L

7.10.1 Detailed Description

For logging events.

For logging events (change of time steps, rejected steps, and so on).

7.10.2 Constructor & Destructor Documentation

7.10.2.1 odes::logger::logger ( ) [inline]

constructor

7.10.2.2 odes::logger::∼logger ( ) [inline]

destructor

7.10.3 Member Function Documentation

7.10.3.1 void odes::logger::clear ( ) [inline]

reset.

7.10.3.2 void odes::logger::print (eventType *E* = all) [inline]

print

Parameters

E	eventype, for filtering results.

7.10.3.3 void odes::logger::put (double *_time,* int *_step,* eventType *_TheEvent,* double *_value*) [inline]

put an event

Parameters

_time	actual time
_step	number
_TheEvent	event type (see struct event).
_value	some value like new time step.

7.10.4 Member Data Documentation

7.10.4.1 list<event> odes::logger::L [private]

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

- common/include/logger.hpp

7.11 odes::Matrices< full, Hessengerg, n, nsub, nsup > Class Template Reference

full matrices.

```
#include <Matrices.hpp>
```

Public Types

- typedef fortranArray< n > MatrixReal
- typedef fortranComplexArray< n > MatrixComplex

Protected Member Functions

- Matrices ()
- void decomr (double fac1, MatrixReal &Jac)

- void decomc (double alpha, double beta, const MatrixReal &Jac)
- void solvereal (fortranVectorF< n > &Z, const MatrixReal &Jac)
- void solvecomplex (fortranVectorF< n > &Zr, fortranVectorF< n > &Zi, const fortranArray< n > &Jac)
- int lbegin (int i) const

  begining of line i.
- int lend (int i) const

  end of line i.

Protected Attributes

- bool calhes

Private Attributes

- int ipivr [n]
- int ipivc [n]
- MatrixReal E1
- MatrixComplex E2R
- fortranVectorF< 2 ∗n > Z2N

7.11.1 Detailed Description

template<bool full, bool Hessengerg, int n, int nsub, int nsup>class odes::Matrices< full, Hessengerg, n, nsub, nsup >

full matrices.

7.11.2 Member Typedef Documentation

7.11.2.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> typedef fortranComplexArray<n> odes::Matrices< full, Hessengerg, n, nsub, nsup >::MatrixComplex

7.11.2.2 template<bool full, bool Hessengerg, int n, int nsub, int nsup> typedef fortranArray<n> odes::Matrices< full, Hessengerg, n, nsub, nsup >::MatrixReal

7.11.3 Constructor & Destructor Documentation

7.11.3.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> odes::Matrices< full, Hessengerg, n, nsub, nsup >::Matrices ( ) [inline], [protected]

7.11.4 Member Function Documentation

7.11.4.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub, nsup >::decomc (double *alpha*, double *beta*, const MatrixReal & *Jac*) [inline], [protected]

build and factorize "complex" matrix

Parameters

alpha	: we add alpha∗I to the real part of the Jacobian.

beta	: we add alpha∗I to the imaginary part of the Jacobian.
Jac	the jacobian.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.11.4.2 template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub, nsup >::decomr (double *fac1*, MatrixReal & *Jac*) [inline], [protected]

build and factorize "real" matrix

Parameters

fac1	: we add fac1∗I to the Jacobian.
Jac	the jacobian.

7.11.4.3 template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lbegin (int *i*) const [inline], [protected]

begining of line i.

7.11.4.4 template<bool full, bool Hessengerg, int n, int nsub, int nsup> int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lend (int *i*) const [inline], [protected]

end of line i.

7.11.4.5 template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvecomplex (fortranVectorF< n > & *Zr*, fortranVectorF< n > & *Zi*, const fortranArray< n > & *Jac*) [inline], [protected]

solve "imaginary" part.

Parameters

Zr	IN/OUT: RHS, real part (IN); result,real part (OUT).
Zi	IN/OUT: RHS, imag. part (IN); result,imag part (OUT).
Jac	the Jacobian matrix.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.11.4.6 template<bool full, bool Hessengerg, int n, int nsub, int nsup> void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvereal (fortranVectorF< n > & *Z*, const MatrixReal & *Jac*) [inline], [protected]

solve "real" part.

Parameters

Z	IN/OUT: 2nd member (IN); result (OUT).
Jac	the Jacobian matrix.

7.11.5 Member Data Documentation

7.11.5.1 template<bool full, bool Hessengerg, int n, int nsub, int nsup> bool odes::Matrices< full, Hessengerg, n, nsub, nsup >::calhes [protected]

7.11.5.2 template< bool full, bool Hessengerg, int n, int nsub, int nsup > MatrixReal odes::Matrices< full, Hessengerg, n, nsub, nsup >::E1 [private]

7.11.5.3 template< bool full, bool Hessengerg, int n, int nsub, int nsup > MatrixComplex odes::Matrices< full, Hessengerg, n, nsub, nsup >::E2R [private]

7.11.5.4 template< bool full, bool Hessengerg, int n, int nsub, int nsup > int odes::Matrices< full, Hessengerg, n, nsub, nsup >::ipivc[n] [private]

7.11.5.5 template< bool full, bool Hessengerg, int n, int nsub, int nsup > int odes::Matrices< full, Hessengerg, n, nsub, nsup >::ipivr[n] [private]

7.11.5.6 template< bool full, bool Hessengerg, int n, int nsub, int nsup > fortranVectorF< 2*n > odes::Matrices< full, Hessengerg, n, nsub, nsup >::Z2N [private]

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

- common/include/Matrices.hpp

7.12 odes::Matrices< false, false, n, nsub, nsup > Class Template Reference

`#include <Matrices.hpp>`

Public Types

- typedef
  fortranRectangularArray< n,
  nsub, nsup > MatrixReal
- typedef
  fortranRectangularComplexArray
  < n, nsub, nsup > MatrixComplex

Protected Member Functions

- Matrices ()

  constructor.
- void decomr (double fac1, const MatrixReal &Jac)

  see full matrix case.
- void decomc (double alpha, double beta, const MatrixReal &Jac)

  see full matrix case.
- void solvereal (fortranVectorF< n > &Z, const MatrixReal &Jac)

  see full matrix case.
- void solvecomplex (fortranVectorF< n > &Zr, fortranVectorF< n > &Zi, const MatrixReal &Jac)

  see full matrix case.
- int lbegin (int i) const

  begining of line i.
- int lend (int i) const

  end of line i.

Protected Attributes

- bool calhes

Private Attributes

- int ipivr [n]
- int ipivc [n]
- MatrixReal E1
- MatrixComplex E2R
- fortranVectorF< 2 *n > Z2N

Static Private Attributes

- static const int ldab =2*nsub+nsup+1

7.12.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< false, false, n, nsub, nsup >

banded matrices, with nsub subdiagonals (nsub>=0) and nsup (>=0) super diagonals. storage is adapted to lapack banded matrices routines.

7.12.2 Member Typedef Documentation

7.12.2.1 template<int n, int nsub, int nsup> typedef **fortranRectangularComplexArray**<n,nsub,nsup> **odes::Matrices**< false, false, n, nsub, nsup >::**MatrixComplex**

7.12.2.2 template<int n, int nsub, int nsup> typedef **fortranRectangularArray**<n,nsub,nsup> **odes::Matrices**< false, false, n, nsub, nsup >::**MatrixReal**

7.12.3 Constructor & Destructor Documentation

7.12.3.1 template<int n, int nsub, int nsup> **odes::Matrices**< false, false, n, nsub, nsup >::**Matrices** ( ) `[inline]`, `[protected]`

constructor.

7.12.4 Member Function Documentation

7.12.4.1 template<int n, int nsub, int nsup> void **odes::Matrices**< false, false, n, nsub, nsup >::decomc (double *alpha,* double *beta,* const **MatrixReal** & *Jac*) `[inline]`, `[protected]`

see full matrix case.

define the ivdep pragma for different compilers.

7.12.4.2 template<int n, int nsub, int nsup> void **odes::Matrices**< false, false, n, nsub, nsup >::decomr (double *fac1,* const **MatrixReal** & *Jac*) `[inline]`, `[protected]`

see full matrix case.

7.12.4.3 template<int n, int nsub, int nsup> int **odes::Matrices**< false, false, n, nsub, nsup >::lbegin (int *i*) const `[inline]`, `[protected]`

begining of line i.

7.12.4.4 template< int n, int nsub, int nsup > int **odes::Matrices**< false, false, n, nsub, nsup >::lend (int *i*) const [inline], [protected]

end of line i.

7.12.4.5 template< int n, int nsub, int nsup > void **odes::Matrices**< false, false, n, nsub, nsup >::solvecomplex (**fortranVectorF**< n > & *Zr*, **fortranVectorF**< n > & *Zi*, const **MatrixReal** & *Jac*) [inline], [protected]

see full matrix case.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.12.4.6 template< int n, int nsub, int nsup > void **odes::Matrices**< false, false, n, nsub, nsup >::solvereal (**fortranVectorF**< n > & *Z*, const **MatrixReal** & *Jac*) [inline], [protected]

see full matrix case.

7.12.5 Member Data Documentation

7.12.5.1 template< int n, int nsub, int nsup > bool **odes::Matrices**< false, false, n, nsub, nsup >::calhes [protected]

7.12.5.2 template< int n, int nsub, int nsup > **MatrixReal odes::Matrices**< false, false, n, nsub, nsup >::E1 [private]

7.12.5.3 template< int n, int nsub, int nsup > **MatrixComplex odes::Matrices**< false, false, n, nsub, nsup >::E2R [private]

7.12.5.4 template< int n, int nsub, int nsup > int **odes::Matrices**< false, false, n, nsub, nsup >::ipivc[n] [private]

7.12.5.5 template< int n, int nsub, int nsup > int **odes::Matrices**< false, false, n, nsub, nsup >::ipivr[n] [private]

7.12.5.6 template< int n, int nsub, int nsup > const int **odes::Matrices**< false, false, n, nsub, nsup >::ldab =2∗nsub+nsup+1 [static], [private]

7.12.5.7 template< int n, int nsub, int nsup > **fortranVectorF**<2∗n> **odes::Matrices**< false, false, n, nsub, nsup >::Z2N [private]

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

- common/include/Matrices.hpp

7.13 odes::Matrices< false, true, n, nsub, nsup > Class Template Reference

Banded matrices, Hessenberg=true: error.

`#include <Matrices.hpp>`

Public Types

- typedef fortranArray< n > MatrixReal
- typedef fortranComplexArray< n > MatrixComplex

Protected Member Functions

- Matrices ()

  constructor.
- void decomr (double fac1, const MatrixReal &Jac)
- void decomc (double alpha, double beta, const MatrixReal &Jac)
- void solvereal (fortranVectorF< n > &Z, const MatrixReal &Jac)
- void solvecomplex (fortranVectorF< n > &Zr, fortranVectorF< n > &Zi, const fortranArray< n > &Jac)
- int lbegin (int i) const
- int lend (int i) const

Protected Attributes

- bool calhes

7.13.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< false, true, n, nsub, nsup >

Banded matrices, Hessenberg=true: error.

7.13.2 Member Typedef Documentation

7.13.2.1 template<int n, int nsub, int nsup> typedef **fortranComplexArray<n> odes::Matrices**< false, true, n, nsub, nsup >::**MatrixComplex**

7.13.2.2 template<int n, int nsub, int nsup> typedef **fortranArray<n> odes::Matrices**< false, true, n, nsub, nsup >::**MatrixReal**

7.13.3 Constructor & Destructor Documentation

7.13.3.1 template<int n, int nsub, int nsup> **odes::Matrices**< false, true, n, nsub, nsup >::**Matrices** ( ) `[inline]`, `[protected]`

constructor.

7.13.4 Member Function Documentation

7.13.4.1 template<int n, int nsub, int nsup> void **odes::Matrices**< false, true, n, nsub, nsup >::decomc (double *alpha,* double *beta,* const **MatrixReal** & *Jac*) `[inline]`, `[protected]`

7.13.4.2 template<int n, int nsub, int nsup> void **odes::Matrices**< false, true, n, nsub, nsup >::decomr (double *fac1,* const **MatrixReal** & *Jac*) `[inline]`, `[protected]`

7.13.4.3 template<int n, int nsub, int nsup> int **odes::Matrices**< false, true, n, nsub, nsup >::lbegin (int *i*) const `[inline]`, `[protected]`

7.13.4.4 template<int n, int nsub, int nsup> int **odes::Matrices**< false, true, n, nsub, nsup >::lend (int *i*) const `[inline]`, `[protected]`

7.13.4.5 template<int n, int nsub, int nsup> void **odes::Matrices**< false, true, n, nsub, nsup >::solvecomplex (**fortranVectorF**< n > & *Zr,* **fortranVectorF**< n > & *Zi,* const **fortranArray**< n > & *Jac*) `[inline]`, `[protected]`

7.13.4.6 template<int n, int nsub, int nsup> void **odes::Matrices**< false, true, n, nsub, nsup >::solvereal (**fortranVectorF**< n > & *Z,* const **MatrixReal** & *Jac*) [inline], [protected]

7.13.5 Member Data Documentation

7.13.5.1 template<int n, int nsub, int nsup> bool **odes::Matrices**< false, true, n, nsub, nsup >::calhes [protected]

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

- common/include/Matrices.hpp

7.14 odes::Matrices< true, true, n, nsub, nsup > Class Template Reference

Full matrices, Hessenberg=true.

`#include <Matrices.hpp>`

Public Types

- typedef fortranArray< n > MatrixReal
- typedef fortranComplexArray< n > MatrixComplex

Protected Member Functions

- Matrices ()

  constructor.
- ∼Matrices ()
- void decomr (double fac1, MatrixReal &Jac)
- void solvereal (fortranVectorF< n > &Z, const MatrixReal &Jac)
- void decomc (double alpha, double beta, const MatrixReal &Jac)
- void solvecomplex (fortranVectorF< n > &Zr, fortranVectorF< n > &Zi, const fortranArray< n > &Jac)
- int Ibegin (int i) const
- int Iend (int i) const

Protected Attributes

- bool calhes

Private Attributes

- double ∗ work
- int lwork
- int ipr [n]
- int ipi [n]
- double tau [n]
- MatrixReal E1
- MatrixReal E2R
- MatrixReal E2I
- fortranVectorF< n > Cr
- fortranVectorF< n > Ci
- fortranVectorF< n > Fr
- fortranVectorF< n > Fi

7.14.1 Detailed Description

template<int n, int nsub, int nsup>class odes::Matrices< true, true, n, nsub, nsup >

Full matrices, Hessenberg=true.

7.14.2 Member Typedef Documentation

7.14.2.1 template<int n, int nsub, int nsup> typedef fortranComplexArray<n> odes::Matrices< true, true, n, nsub, nsup >::MatrixComplex

7.14.2.2 template<int n, int nsub, int nsup> typedef fortranArray<n> odes::Matrices< true, true, n, nsub, nsup >::MatrixReal

7.14.3 Constructor & Destructor Documentation

7.14.3.1 template<int n, int nsub, int nsup> odes::Matrices< true, true, n, nsub, nsup >::Matrices ( ) [inline], [protected]

constructor.

7.14.3.2 template<int n, int nsub, int nsup> odes::Matrices< true, true, n, nsub, nsup >::∼Matrices ( ) [inline], [protected]

7.14.4 Member Function Documentation

7.14.4.1 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::decomc (double *alpha,* double *beta,* const MatrixReal & *Jac*) [inline], [protected]

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.14.4.2 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::decomr (double *fac1,* MatrixReal & *Jac*) [inline], [protected]

7.14.4.3 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lbegin (int *i*) const [inline], [protected]

7.14.4.4 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lend (int *i*) const [inline], [protected]

7.14.4.5 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::solvecomplex (fortranVectorF< n > & *Zr,* fortranVectorF< n > & *Zi,* const fortranArray< n > & *Jac*) [inline], [protected]

7.14.4.6 template<int n, int nsub, int nsup> void odes::Matrices< true, true, n, nsub, nsup >::solvereal (fortranVectorF< n > & *Z,* const MatrixReal & *Jac*) [inline], [protected]

7.14.5 Member Data Documentation

7.14.5.1 template<int n, int nsub, int nsup> bool odes::Matrices< true, true, n, nsub, nsup >::calhes [protected]

7.14.5.2 template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Ci [private]

7.14.5.3 template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Cr [private]

7.14.5.4 template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E1 [private]

7.14.5.5 template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E2I [private]

7.14.5.6 template<int n, int nsub, int nsup> MatrixReal odes::Matrices< true, true, n, nsub, nsup >::E2R [private]

7.14.5.7 template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Fi [private]

7.14.5.8 template<int n, int nsub, int nsup> fortranVectorF<n> odes::Matrices< true, true, n, nsub, nsup >::Fr [private]

7.14.5.9 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::ipi[n] [private]

7.14.5.10 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::ipr[n] [private]

7.14.5.11 template<int n, int nsub, int nsup> int odes::Matrices< true, true, n, nsub, nsup >::lwork [private]

7.14.5.12 template<int n, int nsub, int nsup> double odes::Matrices< true, true, n, nsub, nsup >::tau[n] [private]

7.14.5.13 template<int n, int nsub, int nsup> double* odes::Matrices< true, true, n, nsub, nsup >::work [private]

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

- common/include/Matrices.hpp

7.15 odes::Matrixtype< n, nsub, nsup > Struct Template Reference

```
#include <MatrixType.hpp>
```

Public Types

- typedef Matrices< Full, false,
  n, nsub, nsup >::MatrixReal Matrix

Static Public Attributes

- static const bool Full =(n-nsub)==1&&(n-nsup)==1

7.15.1 Member Typedef Documentation

7.15.1.1 template<int n, int nsub, int nsup> typedef Matrices<Full,false,n,nsub,nsup>::MatrixReal odes::Matrixtype< n, nsub, nsup >::Matrix

7.15.2 Member Data Documentation

7.15.2.1 template<int n, int nsub, int nsup> const bool odes::Matrixtype< n, nsub, nsup >::Full =(n-nsub)==1&&(n-nsup)==1 [static]

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

- common/include/MatrixType.hpp

7.16 odes::Radau5cc< Fonct > Class Template Reference

A C++ implementation of Radau5.

```
#include <Radau5cc.hpp>
```

Public Member Functions

- void setTestPolicy (bool _sameTestValue, double _atol[ ], double _rtol[ ])
- MatrixReal & Jacobian ()

  return a reference to the Jacobian.
- Fonct & rhs ()

  return a reference to the right-hand side object function.
- Radau5cc (bool _sameTestValue, double _atol[ ], double _rtol[ ])
- ∼Radau5cc ()

  destructor
- void performOnlyOneStep (bool val)
- void setRecomputeJacobianTreshold (double _thet)
- void setMaxIterationsNewton (int _nit)
- void setSafe (double _safe)
- void setFacrFacl (double _facr, double _facl)
- void setGustafssonTest (bool val)
- void setNmax (int _nmax)
- void setFnewt (double _fnewt)
- int getNstep () const

  get number of steps performed.
- int getNaccpt () const

  get number of steps accepted.
- int getNrejct () const

  get number of steps rejected.
- int getNJac () const

  get number of jacobian computed.
- int getNdec () const

  get number of matrix decompositions.
- double getLastTimeStep () const

  get last time step used.
- int getNewt () const

  get number of Newton iterations performed.
- int getNfoncCalled () const
- double getfirstAcceptedStep () const
- void operator() (double _h, double &x, double _xend, double y[ ])

Protected Member Functions

- void Jacobian (double t, fortranVector y, const fortranVector Fy)
- void doNewtonMatrices ()

  make Newton matrices (factorize them).
- void slvrad ()

  Linear system solution in Newton iterations.
- std::pair< bool, double > NewtonIterationTest ()
- int Newton (double x, fortranVector &Y)
- void doScal (const fortranVector &Y)
- double estrad (fortranVectorF< n > &y0, fortranVector &y, double t, bool first, bool reject)
- void seth (double x, double newh)

Protected Attributes

- Fonct F
- fortranVectorF< n > Z1
- fortranVectorF< n > Z2
- fortranVectorF< n > Z3
- fortranVectorF< n > F1
- fortranVectorF< n > F2
- fortranVectorF< n > F3
- fortranVectorF< n > Z1T
- fortranVectorF< n > Z2T
- fortranVectorF< n > Z3T

Private Types

- enum OutNewtonLoop { converged, willNotConverge, didNotConverge }
- typedef Matrices< MatrixFull,
  Hessenberg, n, ninf, nsup >
  ::MatrixReal MatrixReal
- typedef fortranComplexArray< n > MatrixComplex

Private Member Functions

- void decomr (double fac1, MatrixReal &Jac)
- void decomc (double alpha, double beta, const MatrixReal &Jac)
- void solvereal (fortranVectorF< n > &Z, const MatrixReal &Jac)
- void solvecomplex (fortranVectorF< n > &Zr, fortranVectorF< n > &Zi, const fortranArray< n > &Jac)
- int lbegin (int i) const

  begining of line i.
- int lend (int i) const

  end of line i.
- ct_assert (full||(!full &&!Hessenberg))

Private Attributes

- double SQ6
- double C1
- double C2
- double C1M1
- double C2M1
- double C1MC2
- double DD1
- double DD2
- double DD3
- double U1
- double ALPH
- double BETA
- double CNO
- double T11
- double T12
- double T13
- double T21
- double T22
- double T23
- double T31
- double TI11
- double TI12
- double TI13
- double TI21
- double TI22
- double TI23
- double TI31
- double TI32
- double TI33
- double uround
- double faccon
- double fnewt
- double tolst
- double xph
- double firstAcceptedStep
- double thet
- double dynold
- double h
- double hhfac
- double theta
- double dyno
- double safe
- double cfac
- double facr
- double facl
- double thqold
- double hacc
- double erracc
- double hold
- double xold
- double xend
- double posneg
- double hmaxn

- double hmax
- double hopt
- double quot1
- double quot2
- bool sameTestValue
- bool startn
- bool first
- bool caljac
- bool reject
- bool gustafssonTest
- bool last
- bool onlyOneStep
- fortranVectorF< n > atol
- fortranVectorF< n > rtol
- fortranVectorF< n > CONT1
- fortranVectorF< n > scal
- fortranVectorF< n > save1
- fortranVectorF< n > save2
- fortranVectorF< n > save3
- MatrixReal Jac
- int naccpt
- int nit
- int newt
- int nmax
- int nrejct
- int njac
- int nstep
- int ndec
- int nfonccalled
- bool calhes

Static Private Attributes

- static const int n =Fonct::n
- static const int ninf =Fonct::nsub
- static const int nsup =Fonct::nsup
- static const bool MatrixFull =(n-ninf)==1&&(n-nsup)==1
- static const bool Hessenberg =Fonct::Hessenberg
- static const bool ComputeJacobianNumerically

7.16.1 Detailed Description

template<class Fonct>class odes::Radau5cc< Fonct >

A C++ implementation of Radau5.

Class Radau5cc

A C++ implementation of (most part of) Hairer & Wanner Radau5. program.

7.16.2 Member Typedef Documentation

7.16.2.1 typedef fortranComplexArray<n> odes::Matrices< full, Hessengerg, n, nsub, nsup >::MatrixComplex [inherited]

7.16.2.2 template<class Fonct > typedef Matrices<MatrixFull,Hessenberg,n,ninf,nsup>::MatrixReal odes::Radau5cc< Fonct >::MatrixReal [private]

7.16.3 Member Enumeration Documentation

7.16.3.1 template<class Fonct > enum odes::Radau5cc::OutNewtonLoop [private]

Enumerator

converged

willNotConverge

didNotConverge

7.16.4 Constructor & Destructor Documentation

7.16.4.1 template<class Fonct > odes::Radau5cc< Fonct >::Radau5cc (bool *_sameTestValue,* double *_atol[ ],* double *_rtol[ ]*) [inline]

constructor

Parameters

_sameTestValue	if true error tolerance are scalar
_atol	absolute tolerance
_rtol	relative tolerance

Note

even if the tolerance are scalar, _atol and _rtol are arrays.

7.16.4.2 template<class Fonct > odes::Radau5cc< Fonct >::∼Radau5cc ( ) [inline]

destructor

7.16.5 Member Function Documentation

7.16.5.1 compat< full, Hessenberg >::ct_assert (full|| *!full &&!Hessenberg*) [inherited]

7.16.5.2 void odes::Matrices< full, Hessengerg, n, nsub, nsup >::decomc (double *alpha,* double *beta,* const MatrixReal & *Jac*) [inline], [protected], [inherited]

build and factorize "complex" matrix

Parameters

alpha	: we add alpha∗I to the real part of the Jacobian.
beta	: we add alpha∗I to the imaginary part of the Jacobian.

Jac	the jacobian.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.16.5.3 void **odes::Matrices**< full, Hessengerg, **n**, nsub, **nsup** >::decomr (double *fac1*, **MatrixReal** & *Jac*) [inline], [protected], [inherited]

build and factorize "real" matrix

Parameters

fac1	: we add fac1∗I to the Jacobian.
Jac	the jacobian.

7.16.5.4 template<class Fonct > void **odes::Radau5cc**< Fonct >::doNewtonMatrices ( ) [inline], [protected]

make Newton matrices (factorize them).

7.16.5.5 template<class Fonct > void **odes::Radau5cc**< Fonct >::doScal (const **fortranVector** & *Y*) [inline], [protected]

(re) compute scal

Parameters

Y	

7.16.5.6 template<class Fonct > double **odes::Radau5cc**< Fonct >::estrad (**fortranVectorF**< **n** > & *y0*, **fortranVector** & *y*, double *t*, bool *first*, bool *reject*) [inline], [protected]

error estimation.

Parameters

y0	
y	
t	
first	
reject	

7.16.5.7 template<class Fonct > double **odes::Radau5cc**< Fonct >::getfirstAcceptedStep ( ) const [inline]

7.16.5.8 template<class Fonct > double **odes::Radau5cc**< Fonct >::getLastTimeStep ( ) const [inline]

get last time step used.

7.16.5.9 template<class Fonct > int **odes::Radau5cc**< Fonct >::getNaccpt ( ) const [inline]

get number of steps accepted.

7.16.5.10 template<class Fonct > int odes::Radau5cc< Fonct >::getNdec ( ) const [inline]

get number of matrix decompositions.

7.16.5.11 template<class Fonct > int odes::Radau5cc< Fonct >::getNewt ( ) const [inline]

get number of Newton iterations performed.

7.16.5.12 template<class Fonct > int odes::Radau5cc< Fonct >::getNfoncCalled ( ) const [inline]

get number of function call.

Note

> Z! if jacobian is computed numerically, we have n call for it. else, an estimation of the cost must take account of the cost of the computation of the exact Jacobian.

7.16.5.13 template<class Fonct > int odes::Radau5cc< Fonct >::getNJac ( ) const [inline]

get number of jacobian computed.

7.16.5.14 template<class Fonct > int odes::Radau5cc< Fonct >::getNrejct ( ) const [inline]

get number of steps rejected.

7.16.5.15 template<class Fonct > int odes::Radau5cc< Fonct >::getNstep ( ) const [inline]

get number of steps performed.

7.16.5.16 int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lbegin (int *i*) const [inline], [protected], [inherited]

begining of line i.

7.16.5.17 int odes::Matrices< full, Hessengerg, n, nsub, nsup >::lend (int *i*) const [inline], [protected], [inherited]

end of line i.

7.16.5.18 template<class Fonct > void odes::Radau5cc< Fonct >::Jacobian (double *t*, fortranVector *y*, const fortranVector *Fy*) [inline], [protected]

compute (numerically) jacobian of Fonct

Parameters

t	current time (IN)
y	current value of unknowns (IN)
Fy	computed values of F(t,y,..) (IN)

7.16.5.19 template<class Fonct > MatrixReal& odes::Radau5cc< Fonct >::Jacobian ( ) [inline]

return a reference to the Jacobian.

7.16.5.20 template<class Fonct > int odes::Radau5cc< Fonct >::Newton (double *x,* fortranVector & *Y*) [inline], [protected]

perform simplified Newton iterations.

Parameters

x	time
Y	current value

Note

we return the reason for which we break iterations.

7.16.5.21 template<class Fonct > std::pair<bool,double> odes::Radau5cc< Fonct >::NewtonIterationTest ( ) [inline], [protected]

Test simplified Newton iterations convergence.

Note

return a pair made ok: (true or false) depending if the convergence. a suggested new value for the time step h.

7.16.5.22 template<class Fonct > void odes::Radau5cc< Fonct >::operator() (double *_h,* double & *x,* double *_xend,* double *y[]*) [inline]

make one time step.

Parameters

_h	time step
x	actual time
_xend	we integrate from x to _xend
y[]	current unknowns values.

7.16.5.23 template<class Fonct > void odes::Radau5cc< Fonct >::performOnlyOneStep (bool *val*) [inline]

operator will perform only one step, and return,whatever the value of time x.

Parameters

val	set onlyOneStep to val

Note

default is false, ie operator() will integrate up to _xend (see operator()).

7.16.5.24 template<class Fonct > Fonct& odes::Radau5cc< Fonct >::rhs ( ) [inline]

return a reference to the right-hand side object function.

7.16.5.25 template<class Fonct > void odes::Radau5cc< Fonct >::setFacrFacl (double *_facr,* double *_facl*) [inline]

change parameters for step size reduction.

Parameters

_facr	
_facl	

Note

new step h is chose so that _facl<= h/hold<=facr

7.16.5.26 template<class Fonct > void odes::Radau5cc< Fonct >::setFnewt (double *_fnewt*) [inline]

change stopping criterion for Newton iterations.

Parameters

_fnewt	new value.

7.16.5.27 template<class Fonct > void odes::Radau5cc< Fonct >::setGustafssonTest (bool *val*) [inline]

perform Gustafsson test? (default yes).

Parameters

val	boolean.

7.16.5.28 template<class Fonct > void odes::Radau5cc< Fonct >::seth (double *x*, double *newh*) [inline], [protected]

set value of h.

Parameters

x	current value of time.
newh	newvalue of h.

Note

can be used to keep an history of h modifications.

7.16.5.29 template<class Fonct > void odes::Radau5cc< Fonct >::setMaxIterationsNewton (int *_nit*) [inline]

7.16.5.30 template<class Fonct > void odes::Radau5cc< Fonct >::setNmax (int *_nmax*) [inline]

change the maximum number of allowed steps.

Parameters

_nmax	new value.

7.16.5.31 template<class Fonct > void odes::Radau5cc< Fonct >::setRecomputeJacobianTreshold (double *_thet*) [inline]

7.16.5.32 template<class Fonct > void odes::Radau5cc< Fonct >::setSafe (double *_safe*) [inline]

change safety factor for step size reduction

Parameters

_safe	new value

7.16.5.33 template<class Fonct > void odes::Radau5cc< Fonct >::setTestPolicy (bool *_sameTestValue,* double *_atol[ ],* double *_rtol[ ]*) [inline]

set the error test policy

Parameters

_sameTestValue	== true iff atol and rtol are scalar
_atol	
_rtol	

7.16.5.34 template<class Fonct > void odes::Radau5cc< Fonct >::slvrad ( ) [inline], [protected]

Linear system solution in Newton iterations.

7.16.5.35 void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvecomplex (fortranVectorF< n > & *Zr,* fortranVectorF< n > & *Zi,* const fortranArray< n > & *Jac*) [inline], [protected], [inherited]

solve "imaginary" part.

Parameters

Zr	IN/OUT: RHS, real part (IN); result,real part (OUT).
Zi	IN/OUT: RHS, imag. part (IN); result,imag part (OUT).
Jac	the Jacobian matrix.

define the ivdep pragma for different compilers.

define the ivdep pragma for different compilers.

7.16.5.36 void odes::Matrices< full, Hessengerg, n, nsub, nsup >::solvereal (fortranVectorF< n > & *Z,* const MatrixReal & *Jac*) [inline], [protected], [inherited]

solve "real" part.

Parameters

Z	IN/OUT: 2nd member (IN); result (OUT).
Jac	the Jacobian matrix.

7.16.6 Member Data Documentation

7.16.6.1 template<class Fonct > double odes::Radau5cc< Fonct >::ALPH [private]

7.16.6.2 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::atol [private]

7.16.6.3 template<class Fonct > double odes::Radau5cc< Fonct >::BETA [private]

7.16.6.4 template<class Fonct > double odes::Radau5cc< Fonct >::C1 [private]

7.16.6.5 template<class Fonct > double odes::Radau5cc< Fonct >::C1M1 [private]

7.16.6.6 template<class Fonct > double odes::Radau5cc< Fonct >::C1MC2 [private]

7.16.6.7 template<class Fonct > double odes::Radau5cc< Fonct >::C2 [private]

7.16.6.8 template<class Fonct > double odes::Radau5cc< Fonct >::C2M1 [private]

7.16.6.9 bool odes::Matrices< full, Hessengerg, n, nsub, nsup >::calhes [protected], [inherited]

7.16.6.10 template<class Fonct > bool odes::Radau5cc< Fonct >::caljac [private]

7.16.6.11 template<class Fonct > double odes::Radau5cc< Fonct >::cfac [private]

7.16.6.12 template<class Fonct > double odes::Radau5cc< Fonct >::CNO [private]

7.16.6.13 template<class Fonct > const bool odes::Radau5cc< Fonct >::ComputeJacobianNumerically [static], [private]

Initial value:

```
=
      Fonct::ComputeJacobianNumerically
```

7.16.6.14 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::CONT1 [private]

7.16.6.15 template<class Fonct > double odes::Radau5cc< Fonct >::DD1 [private]

7.16.6.16 template<class Fonct > double odes::Radau5cc< Fonct >::DD2 [private]

7.16.6.17 template<class Fonct > double odes::Radau5cc< Fonct >::DD3 [private]

7.16.6.18 template<class Fonct > double odes::Radau5cc< Fonct >::dyno [private]

7.16.6.19 template<class Fonct > double odes::Radau5cc< Fonct >::dynold [private]

7.16.6.20 template<class Fonct > double odes::Radau5cc< Fonct >::erracc [private]

7.16.6.21 template<class Fonct > Fonct odes::Radau5cc< Fonct >::F [protected]

7.16.6.22 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F1 [protected]

7.16.6.23 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F2 [protected]

7.16.6.24 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::F3 [protected]

7.16.6.25 template<class Fonct > double odes::Radau5cc< Fonct >::faccon [private]

7.16.6.26 template<class Fonct > double odes::Radau5cc< Fonct >::facl [private]

7.16.6.27 template<class Fonct > double odes::Radau5cc< Fonct >::facr [private]

7.16.6.28 template<class Fonct > bool odes::Radau5cc< Fonct >::first [private]

7.16.6.29 template<class Fonct > double odes::Radau5cc< Fonct >::firstAcceptedStep [private]

7.16.6.30 template<class Fonct > double odes::Radau5cc< Fonct >::fnewt [private]

7.16.6.31 template< class Fonct > bool odes::Radau5cc< Fonct >::gustafssonTest [private]

7.16.6.32 template< class Fonct > double odes::Radau5cc< Fonct >::h [private]

7.16.6.33 template< class Fonct > double odes::Radau5cc< Fonct >::hacc [private]

7.16.6.34 template< class Fonct > const bool odes::Radau5cc< Fonct >::Hessenberg =Fonct::Hessenberg [static], [private]

7.16.6.35 template< class Fonct > double odes::Radau5cc< Fonct >::hhfac [private]

7.16.6.36 template< class Fonct > double odes::Radau5cc< Fonct >::hmax [private]

7.16.6.37 template< class Fonct > double odes::Radau5cc< Fonct >::hmaxn [private]

7.16.6.38 template< class Fonct > double odes::Radau5cc< Fonct >::hold [private]

7.16.6.39 template< class Fonct > double odes::Radau5cc< Fonct >::hopt [private]

7.16.6.40 template< class Fonct > MatrixReal odes::Radau5cc< Fonct >::Jac [private]

7.16.6.41 template< class Fonct > bool odes::Radau5cc< Fonct >::last [private]

7.16.6.42 template< class Fonct > const bool odes::Radau5cc< Fonct >::MatrixFull =(n-ninf)==1&&(n-nsup)==1 [static], [private]

7.16.6.43 template< class Fonct > const int odes::Radau5cc< Fonct >::n =Fonct::n [static], [private]

7.16.6.44 template< class Fonct > int odes::Radau5cc< Fonct >::naccpt [private]

7.16.6.45 template< class Fonct > int odes::Radau5cc< Fonct >::ndec [private]

7.16.6.46 template< class Fonct > int odes::Radau5cc< Fonct >::newt [private]

7.16.6.47 template< class Fonct > int odes::Radau5cc< Fonct >::nfonccalled [private]

7.16.6.48 template< class Fonct > const int odes::Radau5cc< Fonct >::ninf =Fonct::nsub [static], [private]

7.16.6.49 template< class Fonct > int odes::Radau5cc< Fonct >::nit [private]

7.16.6.50 template< class Fonct > int odes::Radau5cc< Fonct >::njac [private]

7.16.6.51 template< class Fonct > int odes::Radau5cc< Fonct >::nmax [private]

7.16.6.52 template< class Fonct > int odes::Radau5cc< Fonct >::nrejct [private]

7.16.6.53 template< class Fonct > int odes::Radau5cc< Fonct >::nstep [private]

7.16.6.54 template< class Fonct > const int odes::Radau5cc< Fonct >::nsup =Fonct::nsup [static], [private]

7.16.6.55 template< class Fonct > bool odes::Radau5cc< Fonct >::onlyOneStep [private]

7.16.6.56 template< class Fonct > double odes::Radau5cc< Fonct >::posneg [private]

7.16.6.57 template< class Fonct > double odes::Radau5cc< Fonct >::quot1 [private]

7.16.6.58 template<class Fonct > double odes::Radau5cc< Fonct >::quot2 `[private]`

7.16.6.59 template<class Fonct > bool odes::Radau5cc< Fonct >::reject `[private]`

7.16.6.60 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::rtol `[private]`

7.16.6.61 template<class Fonct > double odes::Radau5cc< Fonct >::safe `[private]`

7.16.6.62 template<class Fonct > bool odes::Radau5cc< Fonct >::sameTestValue `[private]`

7.16.6.63 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::save1 `[private]`

7.16.6.64 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::save2 `[private]`

7.16.6.65 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::save3 `[private]`

7.16.6.66 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::scal `[private]`

7.16.6.67 template<class Fonct > double odes::Radau5cc< Fonct >::SQ6 `[private]`

7.16.6.68 template<class Fonct > bool odes::Radau5cc< Fonct >::startn `[private]`

7.16.6.69 template<class Fonct > double odes::Radau5cc< Fonct >::T11 `[private]`

7.16.6.70 template<class Fonct > double odes::Radau5cc< Fonct >::T12 `[private]`

7.16.6.71 template<class Fonct > double odes::Radau5cc< Fonct >::T13 `[private]`

7.16.6.72 template<class Fonct > double odes::Radau5cc< Fonct >::T21 `[private]`

7.16.6.73 template<class Fonct > double odes::Radau5cc< Fonct >::T22 `[private]`

7.16.6.74 template<class Fonct > double odes::Radau5cc< Fonct >::T23 `[private]`

7.16.6.75 template<class Fonct > double odes::Radau5cc< Fonct >::T31 `[private]`

7.16.6.76 template<class Fonct > double odes::Radau5cc< Fonct >::thet `[private]`

7.16.6.77 template<class Fonct > double odes::Radau5cc< Fonct >::theta `[private]`

7.16.6.78 template<class Fonct > double odes::Radau5cc< Fonct >::thqold `[private]`

7.16.6.79 template<class Fonct > double odes::Radau5cc< Fonct >::TI11 `[private]`

7.16.6.80 template<class Fonct > double odes::Radau5cc< Fonct >::TI12 `[private]`

7.16.6.81 template<class Fonct > double odes::Radau5cc< Fonct >::TI13 `[private]`

7.16.6.82 template<class Fonct > double odes::Radau5cc< Fonct >::TI21 `[private]`

7.16.6.83 template<class Fonct > double odes::Radau5cc< Fonct >::TI22 `[private]`

7.16.6.84 template<class Fonct > double odes::Radau5cc< Fonct >::TI23 `[private]`

7.16.6.85 template<class Fonct > double odes::Radau5cc< Fonct >::TI31 `[private]`

7.16.6.86 template<class Fonct > double odes::Radau5cc< Fonct >::TI32 [private]

7.16.6.87 template<class Fonct > double odes::Radau5cc< Fonct >::TI33 [private]

7.16.6.88 template<class Fonct > double odes::Radau5cc< Fonct >::tolst [private]

7.16.6.89 template<class Fonct > double odes::Radau5cc< Fonct >::U1 [private]

7.16.6.90 template<class Fonct > double odes::Radau5cc< Fonct >::uround [private]

7.16.6.91 template<class Fonct > double odes::Radau5cc< Fonct >::xend [private]

7.16.6.92 template<class Fonct > double odes::Radau5cc< Fonct >::xold [private]

7.16.6.93 template<class Fonct > double odes::Radau5cc< Fonct >::xph [private]

7.16.6.94 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z1 [protected]

7.16.6.95 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z1T [protected]

7.16.6.96 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z2 [protected]

7.16.6.97 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z2T [protected]

7.16.6.98 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z3 [protected]

7.16.6.99 template<class Fonct > fortranVectorF<n> odes::Radau5cc< Fonct >::Z3T [protected]

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

- Radau5-NG/include/Radau5cc.hpp

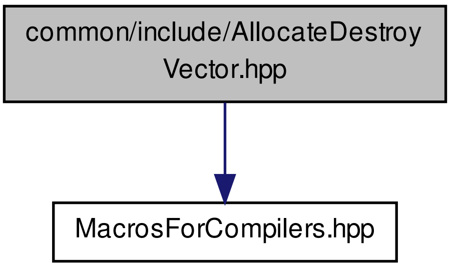
Chapter 8

File Documentation

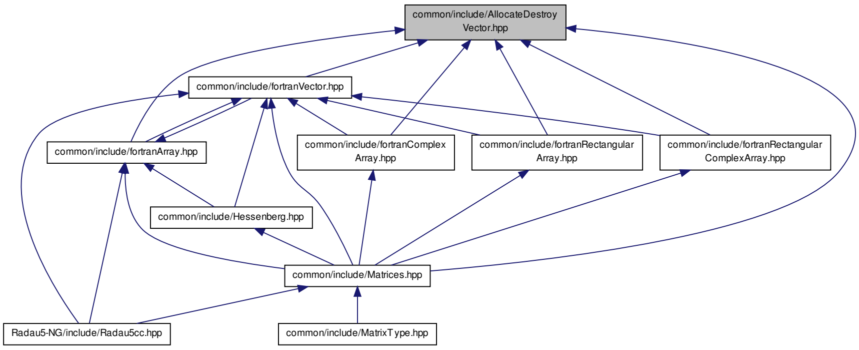
8.1 common/include/AllocateDestroyVector.hpp File Reference

```
#include "MacrosForCompilers.hpp"
```

Include dependency graph for AllocateDestroyVector.hpp:



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


common/include/AllocateDestroy
Vector.hpp
common/include/fortranVector.hpp
common/include/fortranArray.hpp
common/include/fortranComplex
Array.hpp
common/include/fortranRectangular
Array.hpp
common/include/fortranRectangular
ComplexArray.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
Radau5-NG/include/Radau5cc.hpp
common/include/MatrixType.hpp

Macros

- #define ASSUME_ALIGNED(lvalueptr)

Functions

- double ∗ allocDoubleArray (int size)

  Define different methods to allocate arrays.
- void destroyDoubleArray (double ∗x)

8.1.1 Macro Definition Documentation

8.1.1.1 #define ASSUME_ALIGNED(*lvalueptr*)

8.1.2 Function Documentation

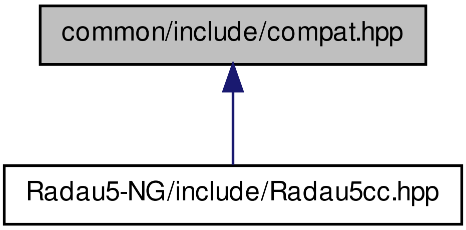
8.1.2.1 double∗ allocDoubleArray (int *size*)

Define different methods to allocate arrays.

8.1.2.2 void destroyDoubleArray (double ∗ *x*)

8.2 common/include/compat.hpp File Reference

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



Classes

- struct compat< full, Hessenberg >

Macros

- #define ASSERT_CONCAT_(a, b) a##b
- #define ASSERT_CONCAT(a, b) ASSERT_CONCAT_(a, b)
- #define ct_assert(e) enum { ASSERT_CONCAT(assert_line_, __LINE__) = 1/(!!(e)) }

8.2.1 Macro Definition Documentation

8.2.1.1 #define ASSERT_CONCAT(*a,* *b*) ASSERT_CONCAT_(a, b)

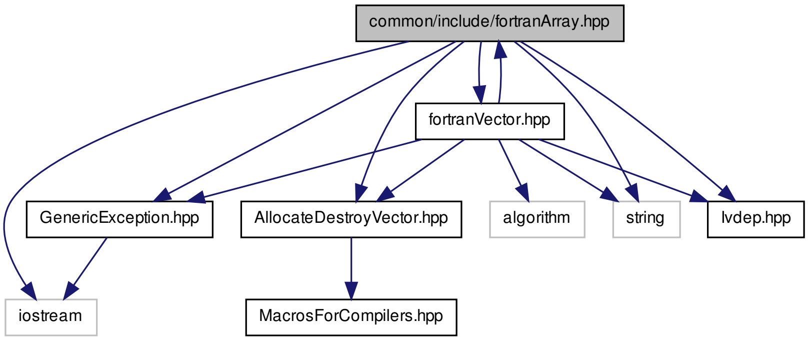
8.2.1.2 #define ASSERT_CONCAT_(*a,* *b*) a##b

8.2.1.3 #define ct_assert(*e*) enum { ASSERT_CONCAT(assert_line_, __LINE__) = 1/(!!(e)) }

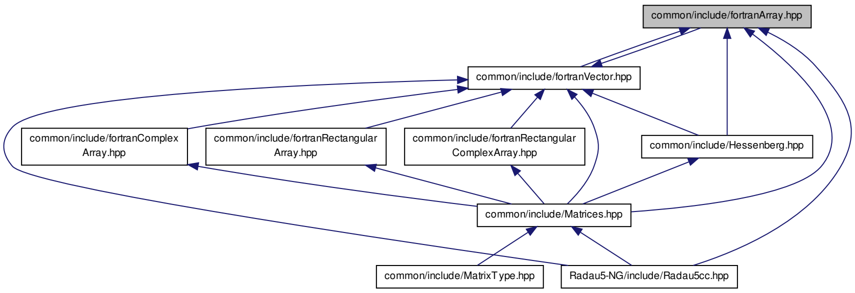
8.3 common/include/fortranArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include <string>
#include <iostream>
#include "AllocateDestroyVector.hpp"
#include "Ivdep.hpp"
```

Include dependency graph for fortranArray.hpp:



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


common/include/fortranArray.hpp
common/include/fortranVector.hpp
common/include/fortranComplex Array.hpp
common/include/fortranRectangular Array.hpp
common/include/fortranRectangular ComplexArray.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
common/include/MatrixType.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::fortranArray< n >

    Array of double.

Namespaces

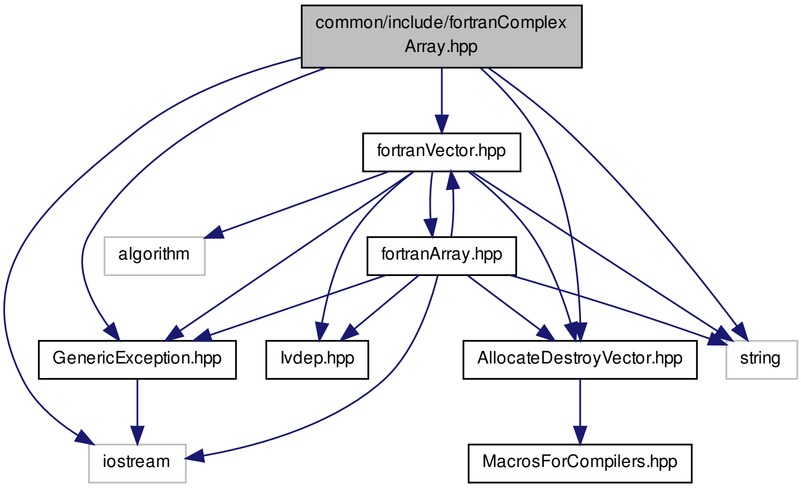
- odes

    Compute all matrix operations.

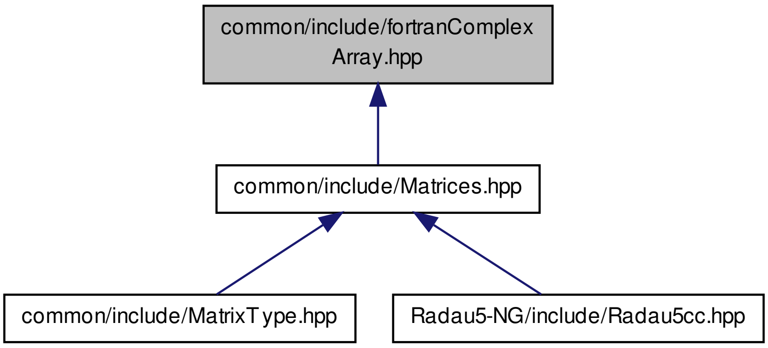
8.4 common/include/fortranComplexArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include <string>
#include <iostream>
#include "AllocateDestroyVector.hpp"
```

Include dependency graph for fortranComplexArray.hpp:



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


common/include/fortranComplex Array.hpp
common/include/Matrices.hpp
common/include/MatrixType.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::fortranComplexArray< n >

    Array of complex..

Namespaces

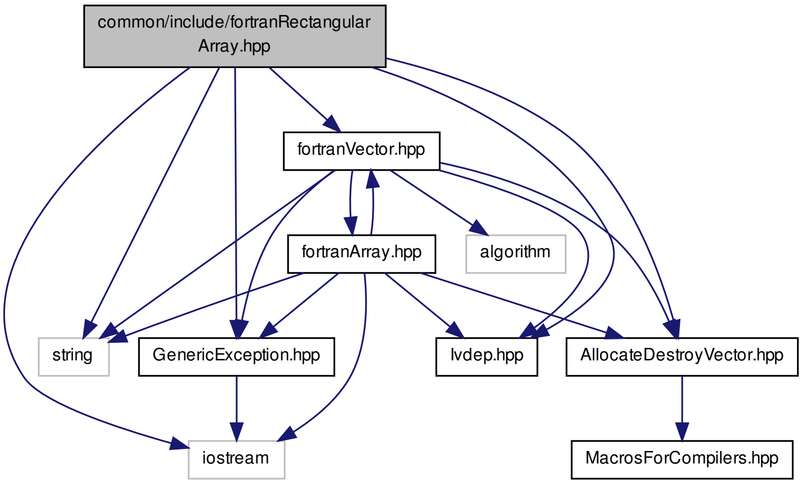
- odes

    Compute all matrix operations.

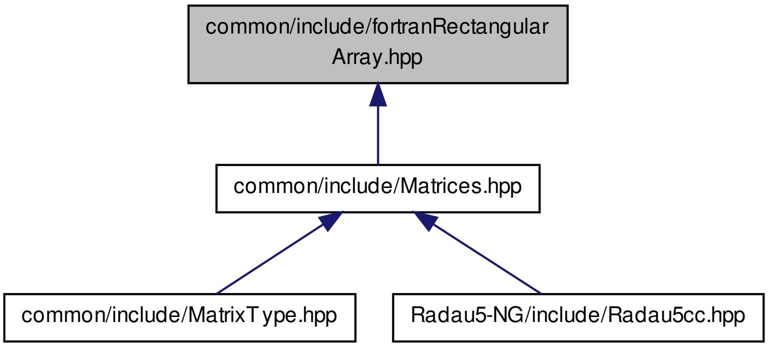
8.5 common/include/fortranRectangularArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include "AllocateDestroyVector.hpp"
#include <string>
#include <iostream>
#include "Ivdep.hpp"
```

Include dependency graph for fortranRectangularArray.hpp:



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


common/include/fortranRectangular Array.hpp
common/include/Matrices.hpp
common/include/MatrixT ype.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::fortranRectangularArray< n, kl, ku >

  Banded Array of double.

Namespaces

- odes

  Compute all matrix operations.

Macros

- #define MAX(x, y) ((x)>(y)?(x):(y))
- #define MIN(x, y) ((x)<(y)?(x):(y))

8.5.1 Macro Definition Documentation

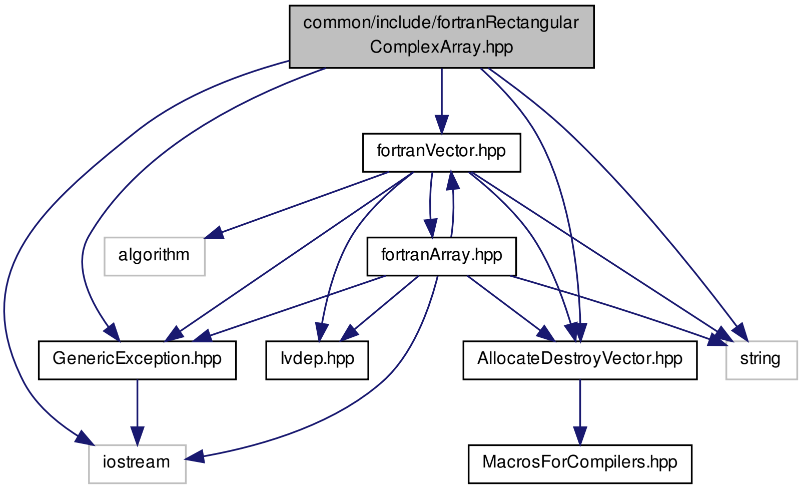
8.5.1.1 #define MAX(*x*, *y*) ((x)>(y)?(x):(y))

8.5.1.2 #define MIN(*x*, *y*) ((x)<(y)?(x):(y))

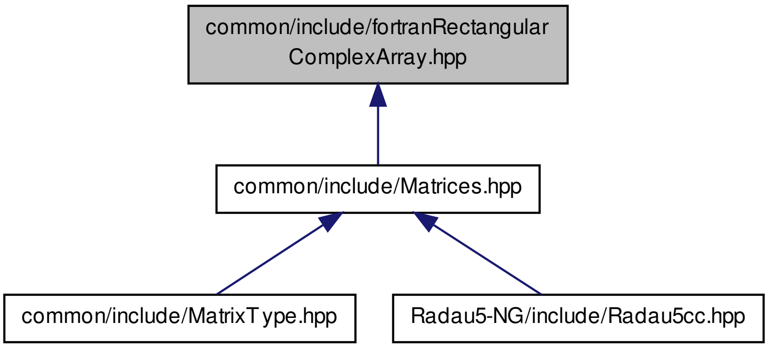
8.6 common/include/fortranRectangularComplexArray.hpp File Reference

```
#include "fortranVector.hpp"
#include "GenericException.hpp"
#include "AllocateDestroyVector.hpp"
#include <string>
#include <iostream>
```

Include dependency graph for fortranRectangularComplexArray.hpp:



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


common/include/fortranRectangular ComplexArray.hpp
common/include/Matrices.hpp
common/include/MatrixType.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::fortranRectangularComplexArray< n, kl, ku >

  Banded Array of complex..

Namespaces

- odes

  Compute all matrix operations.

Macros

- #define MAX(x, y) ((x)>(y)?(x):(y))
- #define MIN(x, y) ((x)<(y)?(x):(y))

8.6.1 Macro Definition Documentation

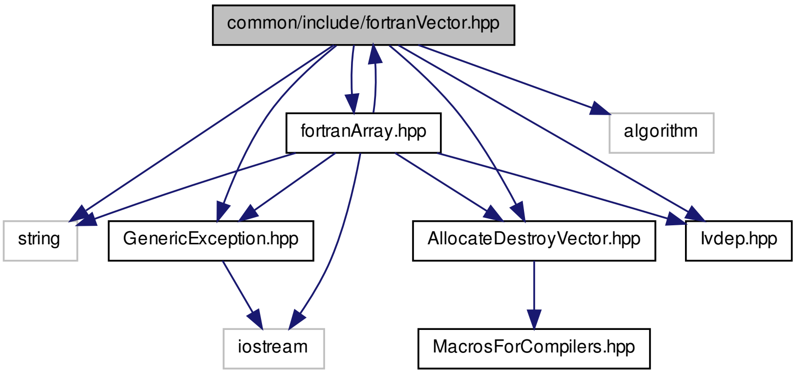
8.6.1.1 #define MAX(*x*, *y*) ((x)>(y)?(x):(y))

8.6.1.2 #define MIN(*x*, *y*) ((x)<(y)?(x):(y))

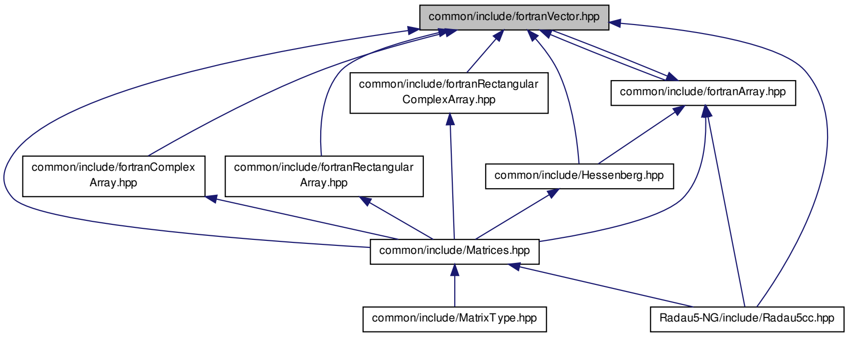
8.7 common/include/fortranVector.hpp File Reference

```
#include "GenericException.hpp"
#include "fortranArray.hpp"
#include "AllocateDestroyVector.hpp"
#include <algorithm>
#include <string>
#include "Ivdep.hpp"
```

Include dependency graph for fortranVector.hpp:



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


common/include/fortranVector.hpp
common/include/fortranRectangular ComplexArray.hpp
common/include/fortranArray.hpp
common/include/fortranComplex Array.hpp
common/include/fortranRectangular Array.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
common/include/MatrixType.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::fortranVector

  Vector class (of doubles).
- class odes::fortranVectorF< n >

  Vector class (of doubles).

Namespaces

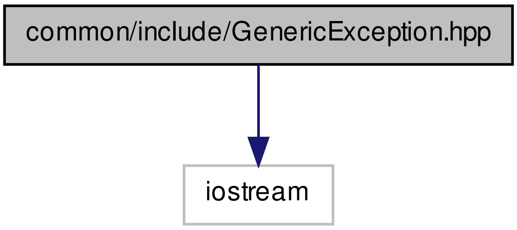
- odes

  Compute all matrix operations.

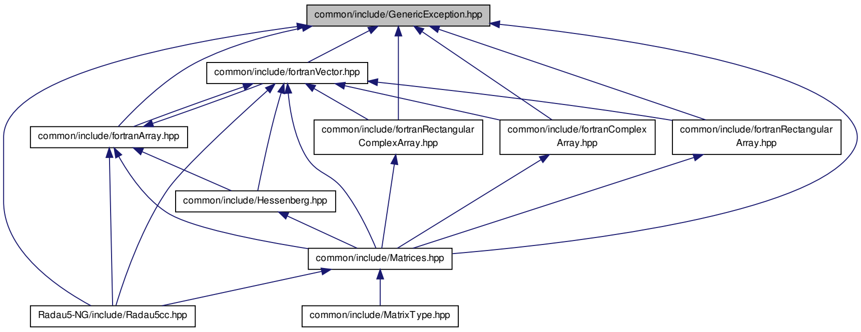
8.8 common/include/GenericException.hpp File Reference

```
#include <iostream>
```

Include dependency graph for GenericException.hpp:



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


common/include/GenericException.hpp
common/include/fortranVector.hpp
common/include/fortranArray.hpp
common/include/fortranRectangular ComplexArray.hpp
common/include/fortranComplex Array.hpp
common/include/fortranRectangular Array.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
Radau5-NG/include/Radau5cc.hpp
common/include/MatrixType.hpp

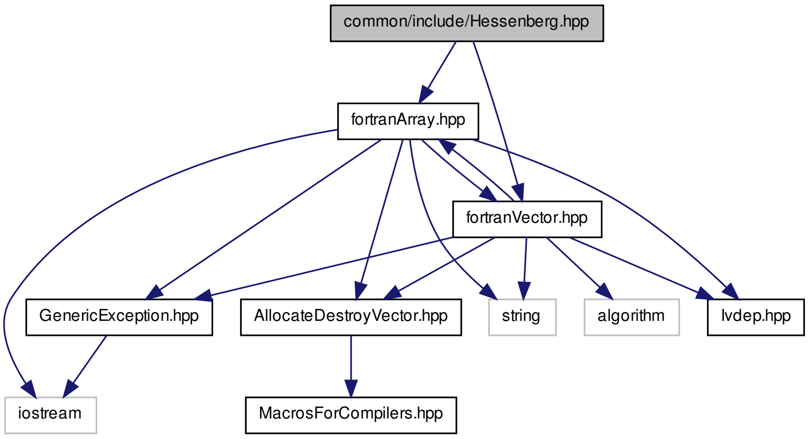
Classes

- class GenericException

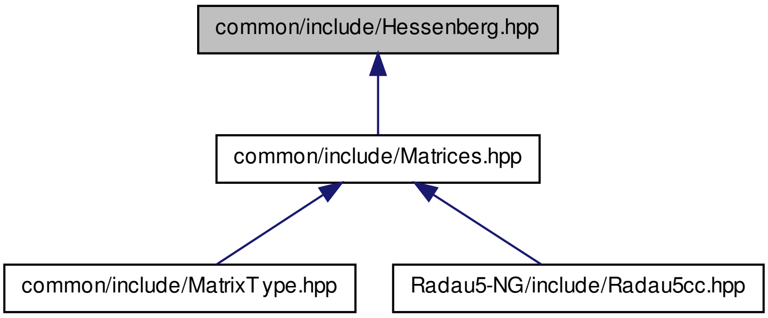
8.9 common/include/Hessenberg.hpp File Reference

```
#include "fortranArray.hpp"
#include "fortranVector.hpp"
```

Include dependency graph for Hessenberg.hpp:



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


common/include/Hessenberg.hpp
common/include/Matrices.hpp
common/include/MatrixType.hpp
Radau5-NG/include/Radau5cc.hpp

Macros

- #define MIN(x, y) ((x)<(y)?(x):(y))
- #define ABS(a) (((a) >= (0.0)) ? (a) : (-a))

Functions

- template<int n>
  int dech (fortranArray< n > &A, int ip[])

  Triangularization by Gaussian elimination of a Hessenberk Matrix.

- template<int n>
  void solh (const fortranArray< n > &A, const int ip[], fortranVector &B)
- template<int n>
  int dechc (fortranArray< n > &Ar, fortranArray< n > &Ai, int ip[])

  Triangularization by Gaussian elimination of a complex.
- template<int n>
  void solhc (const fortranArray< n > &Ar, const fortranArray< n > &Ai, const int ip[], fortranVector &Br, fortranVector &Bi)
- template<int n>
  void householder (const fortranArray< n > &A, const double tau[], const fortranVector &X, fortranVector &Y, bool direct)

8.9.1 Macro Definition Documentation

8.9.1.1 #define ABS(*a*) (((a) >= (0.0)) ? (a) : (-a))

8.9.1.2 #define MIN(*x, y*) ((x)<(y)?(x):(y))

8.9.2 Function Documentation

8.9.2.1 template<int n> int dech (**fortranArray**< n > & *A,* int *ip[]*)

Triangularization by Gaussian elimination of a Hessenberk Matrix.

All what is necessary to compute with Hessenberg matrices (but not the reduction of a full matrix to Hessenberg form (see Matrices.hpp). Triangularization by Gaussian elimination of a Hessenberk Matrix with lower Bandwidth ==1.

Parameters

A	the matrix (IN/OUT).
ip	index of pivots.

Note

> return 0 ii everything went well, otherwise the rank wher the matrix A was found singular.
> this is a C++ transcription of DECH (in H.&W. fortran code decsol).

8.9.2.2 template<int n> int dechc (**fortranArray**< n > & *Ar,* **fortranArray**< n > & *Ai,* int *ip[]*)

Triangularization by Gaussian elimination of a complex.

Triangularization by Gaussian elimination of a Hessenberk Matrix with lower Bandwidth ==1. The matrix, here is complex, but treated as a pair of 2 real matrices. Hessenberg Matrix.

Note

> pure transciption of H. & W. fortran code.

Parameters

Ar	the matrix,real part (IN/OUT).
Ai	the matrix,imaginary part (IN/OUT).

ip	index of pivots.

Note

> return 0 ii everything went well, otherwise the rank wher the matrix A was found singular.
> this is a C++ transcription of DECHC (in H.&W. fortran code decsol.

8.9.2.3 template<int n> void householder (const **fortranArray**< n > & *A*, const double *tau[ ]*, const **fortranVector** & *X*, **fortranVector** & *Y*, bool *direct*)

Given a Matrix A, in Hessenberg form, as computed by DGEHRD (lapack) with ilo=1 and ihi=n, and a vector X, compute: Y=(Product of Householder reflexions) X.

Parameters

A	matrix computed in DGEHRD (with ilo=1 and ihi=n).
tau	parameters computed in DGEHRD.
X	(IN).
Y	result.
direct	if true apply Householder refelexions from 1 to n-2 otherwise from n-2 down to 1.

Note

> that we use dgehrd (in Matrices.hpp) to reduce the Jacobian matrix to Hessenberg form. We must adapt the computation to the output of this routine.

8.9.2.4 template<int n> void solh (const **fortranArray**< n > & *A*, const int *ip[ ]*, **fortranVector** & *B*)

Solution of a linear system where the matrix A was computed in dech.

Parameters

A	the matrix computed in dech (IN)
ip	the pivots obtained in dech. (IN)
B	the RHS on entry, the solution at the end (IN/OUT).

Note

> this is a transcription of the SOLH routine in H.& W. decsol.

8.9.2.5 template<int n> void solhc (const **fortranArray**< n > & *Ar*, const **fortranArray**< n > & *Ai*, const int *ip[ ]*, **fortranVector** & *Br*, **fortranVector** & *Bi*)

Solution of a linear system where the matrix A was computed in dech.

Parameters

Ar	matrix computed in dechc (IN)
Ai	matrix computed in dechc (IN)
ip	the pivots obtained in dech. (IN)
Br	the RHS on entry (real part) the solution at the end (IN/OUT).

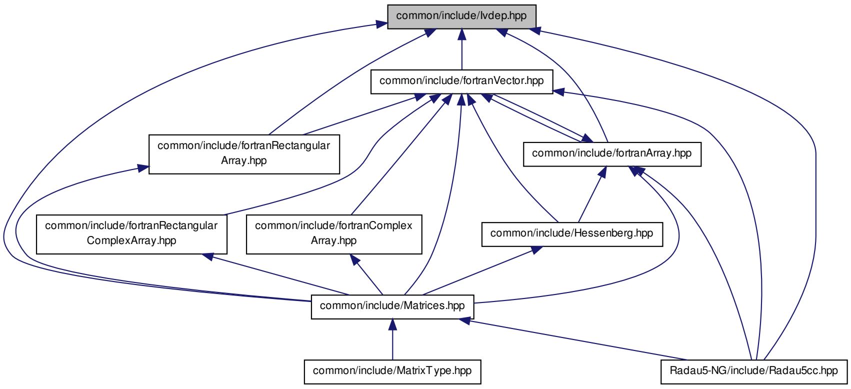
Bi	the RHS on entry (imag. part) the solution at the end (IN/OUT).

Note

this is a C++ transcription of SOLHC (in H.&W. fortran code decsol)

8.10 common/include/lvdep.hpp File Reference

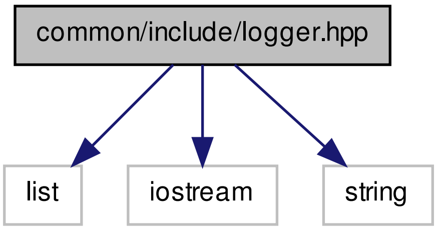
This graph shows which files directly or indirectly include this file:


common/include/lvdep.hpp
common/include/fortranVector.hpp
common/include/fortranRectangular Array.hpp
common/include/fortranArray.hpp
common/include/fortranRectangular ComplexArray.hpp
common/include/fortranComplex Array.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
common/include/MatrixT ype.hpp
Radau5-NG/include/Radau5cc.hpp

8.11 common/include/logger.hpp File Reference

```
#include <list>
#include <iostream>
#include <string>
```

Include dependency graph for logger.hpp:



Classes

- class odes::logger

  For logging events.
- struct odes::logger::event

Namespaces

- odes

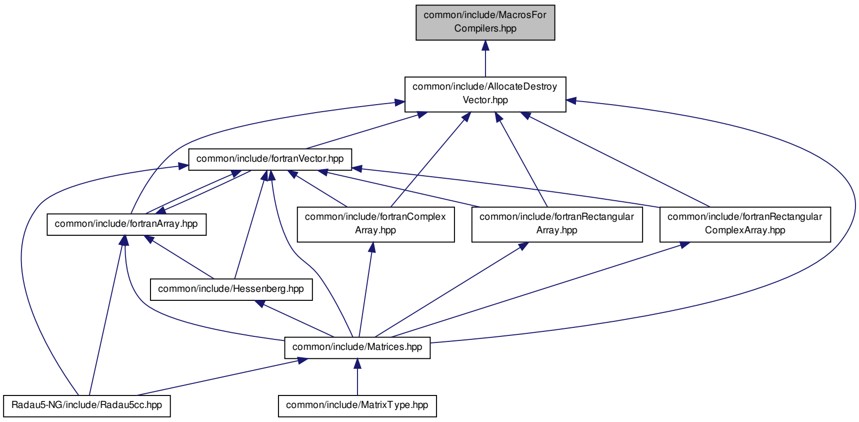
  Compute all matrix operations.

Enumerations

- enum odes::eventType {
  odes::success =0, odes::rejectedStep, odes::NewtonFailed, odes::NewtonWillNotConverge,
  odes::changedH, odes::all }

8.12 common/include/MacrosForCompilers.hpp File Reference

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


common/include/MacrosForCompilers.hpp
common/include/AllocateDestroyVector.hpp
common/include/fortranVector.hpp
common/include/fortranArray.hpp
common/include/fortranComplexArray.hpp
common/include/fortranRectangularArray.hpp
common/include/fortranRectangularComplexArray.hpp
common/include/Hessenberg.hpp
common/include/Matrices.hpp
Radau5-NG/include/Radau5cc.hpp
common/include/MatrixType.hpp

Macros

- #define BLOCK_FACTOR 8∗sizeof(double)

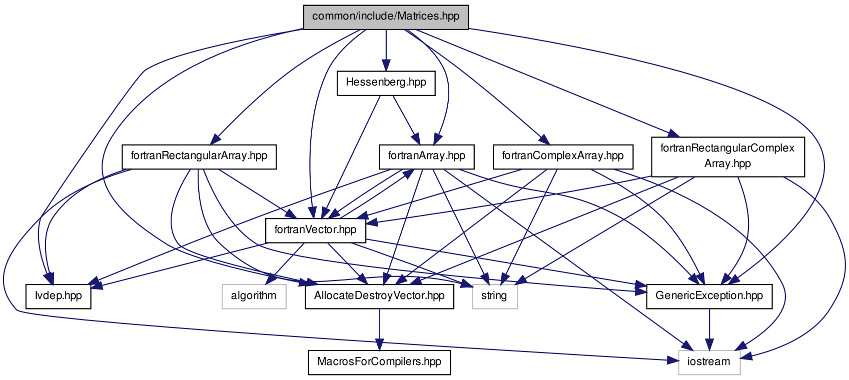
8.12.1 Macro Definition Documentation

8.12.1.1 #define BLOCK_FACTOR 8∗sizeof(double)

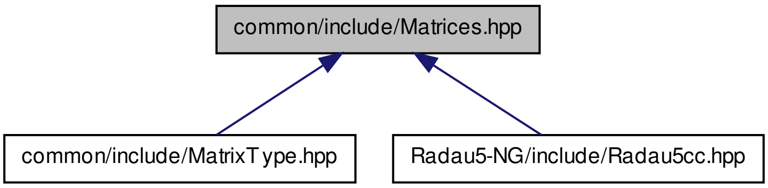
8.13 common/include/Matrices.hpp File Reference

```
#include "fortranArray.hpp"
#include "fortranComplexArray.hpp"
#include "fortranRectangularArray.hpp"
#include "fortranRectangularComplexArray.hpp"
#include "fortranVector.hpp"
#include "AllocateDestroyVector.hpp"
#include "Hessenberg.hpp"
#include "GenericException.hpp"
#include "Ivdep.hpp"
```

Include dependency graph for Matrices.hpp:



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


common/include/Matrices.hpp
common/include/MatrixT ype.hpp
Radau5-NG/include/Radau5cc.hpp

Classes

- class odes::Matrices< full, Hessengerg, n, nsub, nsup >

  full matrices.
- class odes::Matrices< false, false, n, nsub, nsup >
- class odes::Matrices< true, true, n, nsub, nsup >

  Full matrices, Hessenberg=true.
- class odes::Matrices< false, true, n, nsub, nsup >

  Banded matrices, Hessenberg=true: error.

Namespaces

- odes

  Compute all matrix operations.

Macros

- #define MAX(x, y) ((x)>(y)?(x):(y))
- #define MIN(x, y) ((x)<(y)?(x):(y))

8.13.1 Macro Definition Documentation

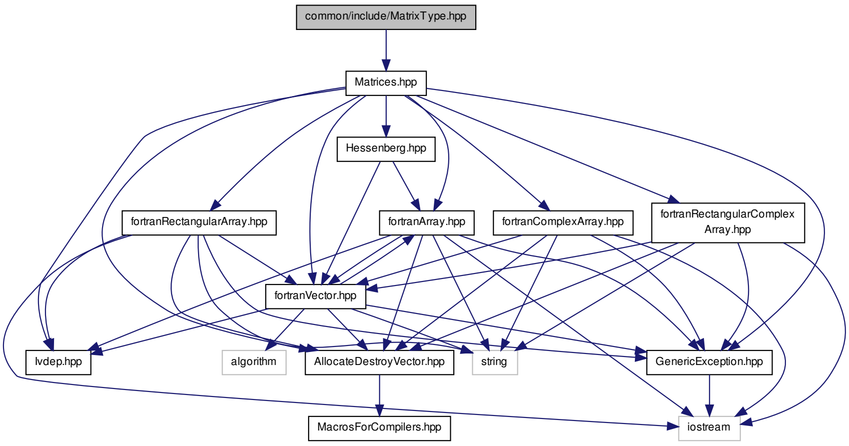
8.13.1.1 #define MAX(*x,* *y*) ((x)>(y)?(x):(y))

8.13.1.2 #define MIN(*x,* *y*) ((x)<(y)?(x):(y))

8.14 common/include/MatrixType.hpp File Reference

```
#include "Matrices.hpp"
```

Include dependency graph for MatrixType.hpp:



Classes

- struct odes::Matrixtype< n, nsub, nsup >

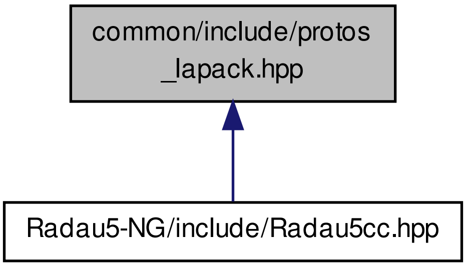
Namespaces

- odes

  Compute all matrix operations.

8.15 common/include/protos_lapack.hpp File Reference

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



Namespaces

- odes

  Compute all matrix operations.

Functions

- void odes::dgetrf_ (int ∗n, int ∗m, double ∗a, int ∗lda, int ∗ipiv, int ∗info)

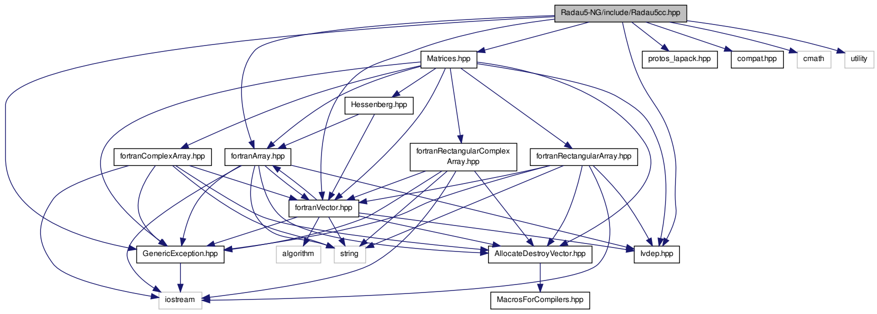
  prototypes for lapack routines.
- void odes::zgetrf_ (int ∗n, int ∗m, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void odes::dgetrs_ (const char ∗s, int ∗N, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void odes::zgetrs_ (const char ∗s, int ∗N, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void odes::dgbtrf_ (int ∗n, int ∗m, int ∗k1, int ∗k2, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void odes::zgbtrf_ (int ∗n, int ∗m, int ∗k1, int ∗k2, double ∗a, int ∗lda, int ∗ipiv, int ∗info)
- void odes::dgbtrs_ (const char ∗s, int ∗N, int ∗k1, int ∗k2, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void odes::zgbtrs_ (const char ∗s, int ∗N, int ∗k1, int ∗k2, int ∗NRHS, double ∗A, int ∗LDA, int ∗IPIV, double ∗B, int ∗LDB, int ∗INFO)
- void odes::dlarnv_ (int ∗idist, int iseed[ ], int ∗n, double ∗x)
- void odes::dgehrd_ (int ∗n, int ∗ilo, int ∗ihi, double ∗a, int ∗lda, double tau[ ], double work[ ], int ∗lwork, int ∗info)
- void odes::dorghr_ (int ∗n, int ∗ilo, int ∗ihi, double ∗a, int ∗lda, double tau[ ], double work[ ], int ∗lwork, int ∗info)

8.16 Mainpage/mainpage.dox File Reference

8.17 Radau5-NG/include/Radau5cc.hpp File Reference

```
#include "GenericException.hpp"
#include "fortranArray.hpp"
#include "fortranVector.hpp"
#include "protos_lapack.hpp"
#include "Matrices.hpp"
#include "compat.hpp"
#include <cmath>
#include <utility>
#include "Ivdep.hpp"
```

Include dependency graph for Radau5cc.hpp:



Classes

- class odes::Radau5cc< Fonct >

  A C++ implementation of Radau5.

Namespaces

- odes

  Compute all matrix operations.

Macros

- #define MAX(x, y) ((x)>(y)?(x):(y))
- #define MIN(x, y) ((x)<(y)?(x):(y))
- #define ABS(a) (((a) >= (0.0)) ? (a) : (-a))
- #define SIGN(a) (((a) >= (0.0)) ? (1.) : (-1.))

8.17.1 Macro Definition Documentation

8.17.1.1 #define ABS(*a*) (((a) >= (0.0)) ? (a) : (-a))

8.17.1.2 #define MAX(*x,* *y*) ((x)>(y)?(x):(y))

8.17.1.3 #define MIN(*x,* *y*) ((x)<(y)?(x):(y))

8.17.1.4 #define SIGN(*a*) (((a) >= (0.0)) ? (1.) : (-1.))

Index