CAPDDDEs

0.2.1

Generated by Doxygen 1.8.17

Chapter 1

Nonrigorous, non-autonomous demo

- before compiling this demo, you need to compile extrenal libraries first, go to the root directory fo the project and read README.md there.
- When you have compiled the external libraries, you can return here and compile this demo
- to compile this demo you need to invoke in this directory:

  ``` make all ```
- it will create ./bin subdirectory

  ``` cd ./bin ```
- to run the program, simply type

  ``` ./draw-nonrig ```
- to see possible options of the program, type

  ``` ./draw-nonrig –help ```
- you may also try to run 'run-demos.sh' from this demo root directory:

  ``` bash run-demos.sh ```
- have a nice day!
  ```

Chapter 2

Exemplary programs

This folder contains exemplary programs to show how to use the library.

As an example consider **demo-elninio** folder. You can compile it by invoking `make all` inside the **demo-elninio** folder, or by invoking `make examples/demo-elninio` in the root directory of the repository.

The executables will be created in **demo-elninio**/**bin** folder, and you can execute `bash run-demos.sh` inside the **demo-elninio** folder to run some demonstrations. The output will go into **demo-elninio**/**bin** folder.

All other examples will provide similar functionality.

You can copy any example folder into ∗∗../devel∗∗ folder and experiment with it/modify it to your liking.

See documentation in ∗∗../devel∗∗ folder to learn how to make your own programs.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Namespace Documentation

6.1 capd Namespace Reference

6.1.1 Detailed Description

This file constitutes part of DDEs rigorous integration framework developed in PhD Thesis:

```
"Rigorous Integration of Delay Differential Equations", Jagiellonian University, 2015
```

When using it in any scientific work please consider citing my articles (preferred), PhD thesis and/or my web page. For the publications describing the source code please refer to http://scirsc.org/p/papers.

This work would not be possible without aid and expertise of people involved in developing CAPD library (Computer Assisted Proofs in Dynamics). Please refer to http://capd.ii.uj.edu.pl and consider citing also the CAPD library when using those codes in any scientific work.

Author: Robert Szczelina, PhD (former) Małopolska Center of Biotechnology, Jagiellonian University AND (currently) Faculty of Mathematics and Computer Science, Jagiellonian University email: robert.szczelina@uj.↩ edu.pl www: scirsc.org

This source code is provided under GNU GPL license (v.2 or whatever)

Chapter 7

Class Documentation

7.1 capd::ddeshelper::ArgumentParser Class Reference

`#include <ddeshelperlib.h>`

Public Member Functions

- **ArgumentParser** (ArgcType &argc, ArgvType &argv)
- std::string **getCommandLine** (std::string sep=" \\\n# ")
- std::string **getHelp** ()
- bool **parse** (std::string const &param, std::string const &help_s="")
- template<typename T >
  bool **parse** (std::string const &param, T &out, std::string const &help_s="")
- template<typename T >
  bool **parse** (std::string const &param, T &out, const char *const help_s)
- template<typename T >
  bool **parse** (std::string const &param, T &out, std::initializer_list< T > allowed_values, std::string const &help_s="")
- template<typename T , typename F >
  bool **parse** (std::string const &param, T &out, F predicate, std::string const &help_s="")

Friends

- template<typename T >
  ArgumentParser & operator<< (ArgumentParser &out, T const &item)

7.1.1 Detailed Description

This is for a nice parsing of arguments from command line

This is not so optimal ('m' * 'argc' * 'string comparison' operations) Where m is number of "parse" parameters. It could be done better, but I decided to keep it simple and produce decent help string automatically.

For now it only supports params in the following form: param=value and param You can do –param=value and –param if you wih For now it does not support multiple forms of params, ie. -f and –file meaning the same, itp.

Example use:

```
capd::ddeshelper::ArgumentParser args(argc, argv);
```

std::string outdir = "."; std::string prefix = "attractor"; int p = 32; int n = 3; std::string plot_mode = "poincare"; args.↩ parse("outdir=", outdir); args.parse("prefix=", prefix, "A prefix of all the filenames"); args.parse("p=", p); args.↩ parse("n=", n, "order of the method"); args.parse("plot=", plot_mode, {std::string("poincare"), std::string("xpx"), std::string("phasespace")}, "Kind of plot to produce and this is a very long explanation what happens in that parameter lorem ipsum sit dolor"); int itest = 5; std::string stest; args.parse("stest=", stest, {std::string("poincare"), std::string("xpx"), std::string("phasespace")}); args.parse("itest=", itest, {-1,0,1}); bool flag = args.parse(std↩ ::string("flag"));

TODO: (not important) add support for variation in params, eg. args.parse({"-h", "--help"}), itp. should be easy TODO: (not important) extract those from the project and send to github as a small library TODO: (not important) add option to handle separated values, ie. "--param data"

7.1.2 Friends And Related Function Documentation

7.1.2.1 operator<<

```
template<typename T >
ArgumentParser& operator<< (
            ArgumentParser & out,
            T const & item )  [friend]
```

outputs with a nice pad

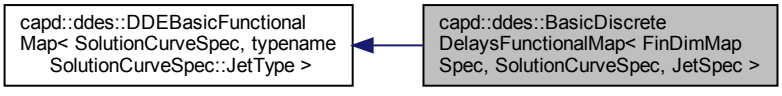
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/ddeshelperlib.↩ h

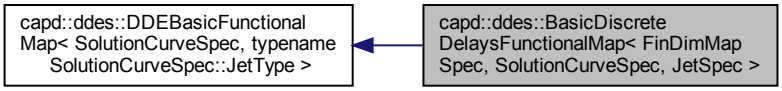
7.2 capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference

```
#include <BasicDiscreteDelaysFunctionalMap.h>
```

Inheritance diagram for capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >:



Collaboration diagram for capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >:



Classes

- struct rebind

Public Types

- typedef DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > **BaseClass**
- typedef BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > **Class**
- typedef SolutionCurveSpec **CurveType**
- typedef JetSpec **JetType**
- typedef FinDimMapSpec **MapType**
- typedef BaseClass::RealType **RealType**
- typedef BaseClass::TimePointType **TimePointType**
- typedef BaseClass::MatrixType **MatrixType**
- typedef BaseClass::VectorType **VectorType**
- typedef BaseClass::ScalarType **ScalarType**
- typedef BaseClass::size_type **size_type**
- typedef BaseClass::DataType **DataType**
- typedef BaseClass::ValueStorageType **ValueStorageType**
- typedef BaseClass::VariableStorageType **VariableStorageType**
- typedef BaseClass::JacobianStorageType **JacobianStorageType**
- typedef std::vector< TimePointType > DelayStorageType
- typedef DelayStorageType::const_iterator const_iterator
- typedef DelayStorageType::iterator iterator

Public Member Functions

- template<typename OtherDiscreteDelaysFunctionalMap >
  BasicDiscreteDelaysFunctionalMap (const OtherDiscreteDelaysFunctionalMap &orig)
- **BasicDiscreteDelaysFunctionalMap** (const BasicDiscreteDelaysFunctionalMap &orig)
- **BasicDiscreteDelaysFunctionalMap** (const TimePointType &tau)
- **BasicDiscreteDelaysFunctionalMap** (const MapType &f)
- **BasicDiscreteDelaysFunctionalMap** (const MapType &f, const TimePointType &tau)
- **BasicDiscreteDelaysFunctionalMap** (const MapType &f, std::vector< TimePointType > delays)
- **BasicDiscreteDelaysFunctionalMap** (const MapType &f, int n, TimePointType delays[])
- virtual ~BasicDiscreteDelaysFunctionalMap ()
- iterator begin ()
- iterator end ()
- const_iterator begin () const
- const_iterator end () const

- size_type imageDimension () const
- size_type dimension () const
- size_type delaysCount () const
- VectorType operator() (const TimePointType &t, const CurveType &x) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, size_type &out_admissible_order) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const

  computeDDECoefficients
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const=0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const

Protected Member Functions

- void **checkDimension** (size_type d) const
- template<typename AnyVector >
  void **checkDimension** (AnyVector const &v) const
- void **checkMapSignature** () const

Protected Attributes

- MapType **m_map**
- DelayStorageType **m_delays**

Additional Inherited Members

7.2.1 Detailed Description

template<typename FinDimMapSpec, typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType>
class capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >

A class to represent right hand side of the DDE of the form:

```
x' = f(t, x(t), x(t-tau_1), ..., x(t-tau_m))
```

NOTE: This is a demo version of what I want to achieve with the DDEs code interface I assume it should be similar to ODEs in original CAPD. I assume that any RHS map f of the equation x'(t) = f(x_\tau) should be representable with similar interface (i.e. the function accepts some SolutionCurve and can produce Jet at current time in the solution defined with the equation (function computeDDECoefficients) I do not assume anything about Solution↩ Curve except, it can produce Forward Taylor Jets of itself at a given point (that is used in the equation) and that it can return its value at a current time.

In the implementation below, I assume that I can ask about SOlutionCUrve and its Jet at $t-\tau$, t is current time. Then I construct an AD jet of x(t-\tau) and pass it to standard CAPD map.

NOTE: I do not know if integral equations (i.e. x'(t) = F(\int_{t-\tau}^t g(x(s)) ds) could be described in this manner... But I hope it could be done, at least if g is linear in x (e.g. g = Id). NOTE: in fact, functional map could have operators that are templates, but we assume that the argument curve might have some requirements when computing, for example it could determine only the delays at the specific grid points (because of the loss of regularity, we are not able to allways give jets over some general intervals, see new notes). Therefore, we supply the Map with appropriate CurveType that it can manipulate.

7.2.2 Member Typedef Documentation

7.2.2.1 const_iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef DelayStorageType::const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< Fin↩
DimMapSpec, SolutionCurveSpec, JetSpec >::const_iterator
```

iterator over this map will iterate over delays!

7.2.2.2 DelayStorageType

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef std::vector< TimePointType > capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDim↩
MapSpec, SolutionCurveSpec, JetSpec >::DelayStorageType
```

storage type for discrete number of delays

7.2.2.3 iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef DelayStorageType::iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMap↩
Spec, SolutionCurveSpec, JetSpec >::iterator
```

iterator over this map will iterate over delays!

7.2.3 Constructor & Destructor Documentation

7.2.3.1 ~BasicDiscreteDelaysFunctionalMap()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::~BasicDiscreteDelaysFunctionalMap ( )  [inline], [virtual]
```

standard

7.2.4 Member Function Documentation

7.2.4.1 begin() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::begin ( )  [inline]
```

iterator over delays

7.2.4.2 begin() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::begin ( ) const  [inline]
```

iterator over delays

7.2.4.3 collectComputationData()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::collectComputationData (
             const TimePointType & t0,
             const TimePointType & th,
             const RealType & dt,
             const CurveType & x,
             VariableStorageType & out_u,
             size_type & out_admissible_order ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h]

Implements capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

7.2.4.4 computeDDECoefficients() [1/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.2.4.5 computeDDECoefficients() [2/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.2.4.6 computeDDECoefficients() [3/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::computeDDECoefficients (
            const RealType & t0,
            const ValueStorageType & u,
            ValueStorageType & coeffs ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs. for k loop

Implements capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

7.2.4.7 computeDDECoefficients() [4/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

7.2.4.8 computeDDECoefficients() [5/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::computeDDECoefficients (
            const RealType & t0,
            const ValueStorageType & u,
            ValueStorageType & coeffs,
            JacobianStorageType & Du ) const  [inline], [virtual]
```

computeDDECoefficients

see Interface docs.

Implements capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

7.2.4.9 computeDDECoefficients() [6/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes $\frac{\partial coeffs[i]}{\partial u}$ (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

7.2.4.10 computeDDECoefficients() [7/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs.↩ resize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE↩ Coefficients(..., u, ...) call to pure virtual function.

7.2.4.11 computeDDECoefficients() [8/8]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
    Du[k] = \frac{\partial coeffs[k]}{\partial u}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

```
    Du[k][j] = \frac{\partial coeffs[k]}{\partial u[j]}
```

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

7.2.4.12 delaysCount()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::delaysCount ( ) const  [inline]
```

number of delays

7.2.4.13 dimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.2.4.14 end() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::end ( )  [inline]
```

iterator over delays

7.2.4.15 end() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::end ( ) const  [inline]
```

iterator over delays

7.2.4.16 imageDimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::imageDimension ( ) const  [inline]
```

output dimension of the internal map

7.2.4.17 operator()()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
VectorType capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::operator() (
            const TimePointType & t,
            const CurveType & x ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs.

Implements capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

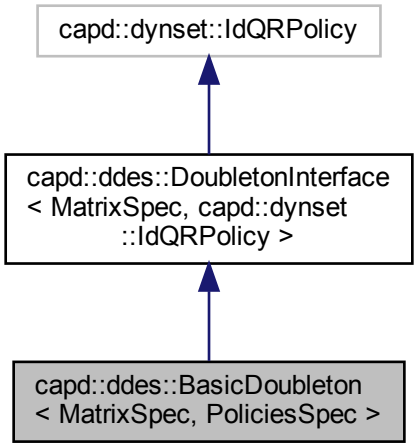
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Basic↩ DiscreteDelaysFunctionalMap.h

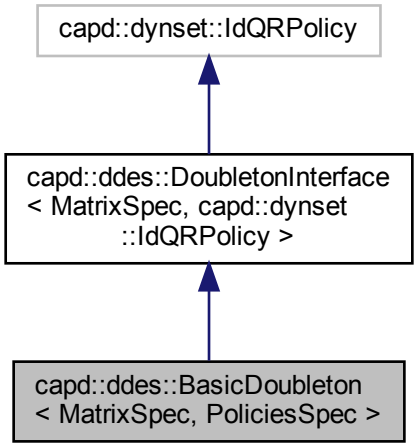
7.3 capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec > Class Template Reference

```
#include <BasicDoubleton.h>
```

Inheritance diagram for capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >:



Collaboration diagram for capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >:



Public Types

- typedef MatrixSpec **MatrixType**
- typedef MatrixType::RowVectorType **VectorType**
- typedef MatrixType::ScalarType **ScalarType**
- typedef MatrixType::size_type **size_type**
- typedef BasicDoubleton **Class**

- typedef DoubletonInterface< MatrixSpec > **BaseClass**
- typedef VectorType * **VectorTypePtr**
- typedef MatrixType * **MatrixTypePtr**
- typedef std::bitset< 5 > **OwnershipType**
- typedef PoliciesSpec **Policy**
- typedef PoliciesSpec **QRPolicy**

Public Member Functions

- Class & operator= (Class const &orig)
- BasicDoubleton ()
- BasicDoubleton (VectorType const &x, VectorType *common_r0=0, bool passOwnership=false)
- BasicDoubleton (Class const &orig)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, MatrixType const &Binv, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0, MatrixType const &B, Vector↩ Type const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0, MatrixType const &B, Matrix↩ Type const &Binv, VectorType const &r)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0)
- BasicDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0)
- BasicDoubleton (VectorType *x, MatrixType *C, VectorType *r0, MatrixType *B, VectorType *r)
- BasicDoubleton (size_type d, size_type N0=-1)
- virtual ~BasicDoubleton ()
- size_type storageN0 () const

  < *import dimension from interface*
- size_type storageDimension () const
- VectorType get_x () const
- MatrixType get_C () const
- VectorType get_r0 () const
- MatrixType get_B () const
- VectorType get_r () const
- BaseClass & set_x (VectorType const &x)
- BaseClass & set_C (MatrixType const &C)
- BaseClass & set_r0 (VectorType const &r0)
- BaseClass & set_B (MatrixType const &B)
- BaseClass & set_r (VectorType const &r)
- BaseClass & set_Cr0 (MatrixType const &C, VectorType const &r0)
- VectorType midPoint () const
- VectorType hull () const
- virtual bool common_x (VectorType const *x) const

  < *using the base class take_* method...*
- virtual bool common_C (MatrixType const *C) const
- virtual bool common_r0 (VectorType const *r0) const
- virtual bool common_B (MatrixType const *B) const
- virtual bool common_r (VectorType const *r) const
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- std::string show () const
- virtual void reinitialize (size_type d, size_type N0)
- virtual size_type dimension () const

- virtual Class & set_x (VectorType const &x)=0
- virtual Class & set_x (VectorType *x, bool passOwnership=false)
- virtual Class & set_C (MatrixType const &C)=0

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_C (MatrixType *C, bool passOwnership=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_r0 (VectorType const &r0)=0

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_r0 (VectorType *r0, bool passOwnership=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_B (MatrixType const &B)=0

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_B (MatrixType *B, bool passOwnership=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_r (VectorType const &r)=0

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_r (VectorType *r, bool passOwnership=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_Cr0 (MatrixType const &C, VectorType const &r0)=0

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_Cr0 (MatrixType *C, VectorType *r0, bool passOwnership1=false, bool pass↩ Ownership2=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_Binv (MatrixType *Binv, bool passOwnership=false)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual Class & set_Binv (MatrixType const &Binv)

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual VectorType * take_x ()

  < *using the base class set_*(pointer, ownershhip) method...*
- virtual MatrixType * take_C ()

  < *using the base class take_* method...*
- virtual VectorType * take_r0 ()

  < *using the base class take_* method...*
- virtual MatrixType * take_B ()

  < *using the base class take_* method...*
- virtual VectorType * take_r ()

  < *using the base class take_* method...*
- virtual MatrixType * take_Binv ()

  < *using the base class take_* method...*
- virtual Class & add (VectorType const &v)
- virtual Class & add (Class const &set)
- virtual Class & mul (ScalarType const &c)
- virtual Class & mulThenAdd (ScalarType const &c, Class const &set)

Protected Member Functions

- void sanityCheck (std::string const &what="") const
- void setupDimension (size_type d, size_type N0=0)
- void setupFromData (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- void setupFromOther (Class const &other)
- void setupFromOther (BaseClass const &other)

Protected Attributes

- VectorType **m_x**
- MatrixType **m_C**
- VectorType **m_r0**
- MatrixType **m_B**
- VectorType **m_r**

Additional Inherited Members

7.3.1 Detailed Description

template<typename MatrixSpec, typename PoliciesSpec = capd::dynset::IdQRPolicy>
class capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >

This class stores a set X \in \R^d of the form:

X = x + C * r0 + B * r

with $x \in \R^d$, $r0, r$ - closed intervals centered at 0, $r0 \subset \R^{N_0}$, $C \in Lin(\R^{N_0}, \R^d)$, $B \in Lin(\R^d, \R^d)$, $r \in \R^d$, B being a matrix easy to invert, e.g. B = ID (Interval form of the remainder) or B orthogonal (Doubleton set with QR decomposition).

NOTE: compare to the classic Lohner Doubleton Set in CAPD, where authors assume $N_0 = d$) NOTE: This is the most basic implementation, I use bare Vectors and Matrices and all the data is stored inside the set. For DDEs as described in notes it is somehow innefficient, as each component will contain its copy of r0 vector which in the computation in fact is common to all the components. Therefore we will store O(size of repr.) elements of size dim(r0) instead of O(1). SharedDoubleton is designed for this purpose, but is is highly complicated as of now (and I have some problems with memory leak - o be corrected later). I have decided to implement simple solution to test things without need to worry about the problems with dynamical memory allocation. The code is also simpler to read.

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

NOTE: THIS PARTICULAR IMPLEMENTATION does not use shared memory, everything is static (from the point of view of implkemnentation, afaik CAPD uses shared ptr inside the library) Its main purpose is testing, as it will be (probably) slower and (for sure) more memory consuming (2x at least) than a SharedDoubleton version. SharedDoubleton will be used in applications.

TODO: (NOT URGENT) implement QR policy in BinvB (now IdPolicy is hardcoded). This is not urgent, as I use SharedDoubleton in my computations.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 BasicDoubleton() [1/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton ( )  [inline]
```

default constructor makes a 1D point-set at 0

7.3.2.2 BasicDoubleton() [2/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            VectorType * common_r0 = 0,
            bool passOwnership = false )  [inline]
```

setup set with a given vector. You can pass common_r0 and passOwnership = false (default) if you want to set externally controlled r0 element. If passOwnership = true then the set will take responsibility for deallocating item.

7.3.2.3 BasicDoubleton() [3/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            Class const & orig )  [inline]
```

copies the original set (ownership is preserved)

7.3.2.4 BasicDoubleton() [4/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0,
            MatrixType const & B,
            VectorType const & r )  [inline]
```

setup this set with a given data, set owns everything

7.3.2.5 BasicDoubleton() [5/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0,
            MatrixType const & B,
            MatrixType const & Binv,
            VectorType const & r )  [inline]
```

setup this set with a given data, set owns everything

7.3.2.6 BasicDoubleton() [6/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0,
            MatrixType const & B,
            VectorType const & r )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.3.2.7 BasicDoubleton() [7/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0,
            MatrixType const & B,
            MatrixType const & Binv,
            VectorType const & r )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.3.2.8 BasicDoubleton() [8/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0 )  [inline]
```

setup this set with a given data, set is owner of everything

7.3.2.9 BasicDoubleton() [9/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0 )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.3.2.10 BasicDoubleton() [10/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            VectorType * x,
            MatrixType * C,
            VectorType * r0,
            MatrixType * B,
            VectorType * r )  [inline]
```

setup this set with a given data but the set does not own the data (user is responsible for deleting)

7.3.2.11 BasicDoubleton() [11/11]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::BasicDoubleton (
            size_type d,
            size_type N0 = -1 )  [inline]
```

setup this set as zero vector, but the structure of given dimensions. If second arg is < 0 then d is used instead.

7.3.2.12 ~BasicDoubleton()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::~BasicDoubleton ( )  [inline],
[virtual]
```

standard thing

7.3.3 Member Function Documentation

7.3.3.1 add() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add  [inline]
```

add a Set or Vector to this representation. It takes the representation into consideration.

7.3.3.2 add() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add  [inline]
```

add a Set or Vector to this representation. It takes the representation into consideration.

7.3.3.3 affineTransform()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::affineTransform (
            MatrixType const & M,
            VectorType const & v )  [inline], [virtual]
```

applies in a smart way to this set X the affine transform f(y) = M ∗ (X - v)

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.4 common_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::common_B (
            MatrixType const * B ) const  [inline], [virtual]
```

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.5 common_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::common_C (
            MatrixType const * C ) const  [inline], [virtual]
```

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.6 common_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::common_r (
            VectorType const * r ) const  [inline], [virtual]
```

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.7 common_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::common_r0 (
            VectorType const * r0 ) const  [inline], [virtual]
```

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.8 common_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::common_x (
            VectorType const * x ) const  [inline], [virtual]
```

< using the base class take_* method...

checked by object equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.9 dimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension  [inline]
```

by default dimension is equal to storage dimension

7.3.3.10 get_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_B ( ) const  [inline]
```

see interface

7.3.3.11 get_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_C ( ) const  [inline]
```

see interface

7.3.3.12 get_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_r ( ) const  [inline]
```

see interface

7.3.3.13 get_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_r0 ( ) const  [inline]
```

see interface

7.3.3.14 get_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::get_x ( ) const  [inline]
```

see interface

7.3.3.15 hull()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::hull ( ) const  [inline]
```

see interface

7.3.3.16 midPoint()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::midPoint ( ) const  [inline]
```

see interface

7.3.3.17 mul()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::mul  [inline]
```

multiply set by a scalar. Should take set structure into consideration.

7.3.3.18 mulThenAdd()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::mulThenAdd  [inline]
```

multiply set by a scalar, then add another set. Very simple basic implementation by first mul then add.

7.3.3.19 operator=()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
Class& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::operator= (
            Class const & orig )  [inline]
```

assign operator - it needs to deallocate memory if neccessery, then setup set anew

7.3.3.20 reinitialize()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::reinitialize (
            size_type d,
            size_type N0 )  [inline], [virtual]
```

see base class

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.21 sanityCheck()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::sanityCheck (
            std::string const & what = "" ) const  [inline], [protected]
```

helper function to check if the object represents sane data (all dimensions compatible and all pointers set if neccesary) should be called after constructing or manipulating object

7.3.3.22 set_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_B (
            MatrixType const & B )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.23 set_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_C (
            MatrixType const & C )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.24 set_Cr0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_Cr0 (
            MatrixType const & C,
            VectorType const & r0 )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.25 set_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_r (
            VectorType const & r )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.26 set_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_r0 (
            VectorType const & r0 )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.27 set_x() [1/3]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x  [inline]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

7.3.3.28 set_x() [2/3]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::set_x (
            VectorType const & x )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.3.3.29 set_x() [3/3]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

7.3.3.30 setupDimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::setupDimension (
            size_type d,
            size_type N0 = 0 )  [inline], [protected]
```

makes all vector and matrices to be of appropriate dimension

7.3.3.31 setupFromData()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::setupFromData (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0,
            MatrixType const & B,
            VectorType const & r ) [inline], [protected]
```

helper in setup

7.3.3.32 setupFromOther() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::setupFromOther (
            BaseClass const & other ) [inline], [protected]
```

helper in copying from base class (interface)

7.3.3.33 setupFromOther() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
void capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::setupFromOther (
            Class const & other ) [inline], [protected]
```

helper in copying

7.3.3.34 show()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
std::string capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::show ( ) const [inline]
```

show info on this set

7.3.3.35 storageDimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::storageDimension ( ) const
[inline]
```

see interface

7.3.3.36 storageN0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::storageN0 ( ) const [inline]
```

< import dimension from interface

see interface

7.3.3.37 translate()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >::translate (
            VectorType const & v )  [inline], [virtual]
```

applies in a smart way to this set X the transform f(y) = X - v

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Basic↩ Doubleton.h

7.4 capd::ddeshelper::BinaryData< T > Union Template Reference

Public Member Functions

- **BinaryData** (T v)

Public Attributes

- T **value**
- char **repr** [sizeof(T)]

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperCommon.h

7.5 capd::ddeshelper::CoordinateFrame< MatrixSpec, VectorSpec, ScalarSpec > Class Template Reference

```
#include <DDECoordinateFrameHelper.h>
```

Public Types

- typedef MatrixSpec **MatrixType**
- typedef VectorSpec **VectorType**
- typedef ScalarSpec **ScalarType**

Public Member Functions

- **CoordinateFrame** (VectorType x, VectorType sec, ScalarType secval, capd::poincare::CrossingDirection d, MatrixType C, MatrixType invC)
- **CoordinateFrame** (VectorType x, MatrixType C, MatrixType invC, capd::poincare::CrossingDirection d=capd::poincare::MinusPlus)
- **CoordinateFrame** (int M)
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- VectorType **inCoords** (VectorType &x)
- VectorType **inGlobal** (VectorType &x)
- VectorType **x0** ()
- MatrixType **C** ()
- MatrixType **invC** ()
- VectorType **sectionVector** ()
- ScalarType **sectionValue** ()

Public Attributes

- int **M**
- VectorType **reference**
- VectorType **vsection**
- ScalarType **secvalue**
- capd::poincare::CrossingDirection **crossingDirection**
- MatrixType **coords**
- MatrixType **inverseCoords**
- std::string **filepath**

Friends

- std::istream & **operator>>** (std::istream &in, CoordinateFrame &coords)
- bool **operator==** (CoordinateFrame &first, CoordinateFrame &second)
- bool **operator!=** (CoordinateFrame &first, CoordinateFrame &second)

7.5.1 Detailed Description

template<typename MatrixSpec, typename VectorSpec = typename MatrixSpec::VectorType, typename ScalarSpec = typename VectorSpec::ScalarType>
class capd::ddeshelper::CoordinateFrame< MatrixSpec, VectorSpec, ScalarSpec >

Internal class to hold a affine coordinate change on a (p,n)-fset.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/DDECoordinateFrameHelper.h

7.6 capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem Class Reference

`#include <DDEHelperRigorous.h>`

Public Member Functions

- **CoordinateSystem** (Vector x, Vector sec, Scalar secval, capd::poincare::CrossingDirection d, Matrix C, Matrix invC)
- **CoordinateSystem** (Vector x, Matrix C, Matrix invC, capd::poincare::CrossingDirection d=capd::poincare↩ ::MinusPlus)
- **CoordinateSystem** (int M)
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- Vector **x0** ()
- Matrix **C** ()
- Matrix **invC** ()
- Vector **sectionVector** ()
- Vector **sectionValue** ()

Public Attributes

- int **M**
- Vector **reference**
- Vector **vsection**
- Scalar **secvalue**
- capd::poincare::CrossingDirection **crossingDirection**
- Matrix **coords**
- Matrix **inverseCoords**
- std::string **filepath**

Friends

- std::istream & **operator>>** (std::istream &in, CoordinateSystem &coords)
- bool **operator==** (CoordinateSystem &first, CoordinateSystem &second)
- bool **operator!=** (CoordinateSystem &first, CoordinateSystem &second)

7.6.1 Detailed Description

template<typename EqSpec, int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11Rect2Policies>
class capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::CoordinateSystem

Internal class to hold a affine coordinate change on a (p,n)-fset.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperRigorous.h

7.7 capd::ddeshelper::CubeSet< VectorSpec > Class Template Reference

Classes

- class CubeSetIterator

Public Types

- typedef VectorSpec **VectorType**
- typedef VectorType::ScalarType **ScalarType**
- typedef VectorType::size_type **size_type**
- typedef std::vector< int > **CubeSignatureType**
- typedef std::set< CubeSignatureType > **CSContainerType**
- typedef CSContainerType::const_iterator **CubeSignatureConstIterator**
- typedef CSContainerType::iterator **CubeSignatureIterator**

Public Member Functions

- **CubeSet** (VectorType const &x0, VectorType const &r0)
- **CubeSet** (VectorType const &r0)
- CubeSet & **insert** (CubeSignatureType const &cube)
- int insert_cover (VectorType box, int max_cuts=-1)
- CubeSetIterator **begin** ()
- CubeSetIterator **end** ()
- CubeSignatureIterator **csbegin** ()
- CubeSignatureIterator **csend** ()
- CubeSignatureConstIterator **csbegin** () const
- CubeSignatureConstIterator **csend** () const
- VectorType **get_x0** () const
- VectorType **get_r0** () const
- VectorType **get_dx** (CubeSignatureType const &signature) const
- bool **operator==** (CubeSet const &other) const
- bool subcubeset (CubeSet const &other) const
- template<typename Iterator >
  CubeSet & **insert** (Iterator from, Iterator to)
- int **size** () const

Friends

- std::ostream & operator<< (std::ostream &out, CubeSet const &items)
- std::istream & operator>> (std::istream &in, CubeSet &items)

7.7.1 Member Function Documentation

7.7.1.1 insert_cover()

```
template<typename VectorSpec >
int capd::ddeshelper::CubeSet< VectorSpec >::insert_cover (
            VectorType box,
            int max_cuts = -1 )  [inline]
```

Max_cuts == -1 means we mean cut in all neccessary dimensions. This is default behaviour. returns the last index at what the cut was needed. If it is smaller than max_cuts requirement then we are ok with the set. If it is greater on equal, then we need to probably change the m_r0, as the box cannot be covered with the current small boxes.

7.7.1.2 subcubeset()

```
template<typename VectorSpec >
bool capd::ddeshelper::CubeSet< VectorSpec >::subcubeset (
            CubeSet< VectorSpec > const & other ) const  [inline]
```

this only check if x0's, r0's are equal (same cube structure), and that all cubes are in the other

7.7.2 Friends And Related Function Documentation

7.7.2.1 operator<<

```
template<typename VectorSpec >
std::ostream& operator<< (
            std::ostream & out,
            CubeSet< VectorSpec > const & items )  [friend]
```

this only outputs cube signatures and add to the set

7.7.2.2 operator>>

```
template<typename VectorSpec >
std::istream& operator>> (
            std::istream & in,
            CubeSet< VectorSpec > & items )  [friend]
```

this only reads cube signatures and adds to the set

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/DDECubeSet.h

7.8 capd::ddeshelper::CubeSet< VectorSpec >::CubeSetIterator Class Reference

Public Member Functions

- **CubeSetIterator** (CubeSetIterator const &orig)
- **operator VectorType** () const
- CubeSignatureType const & **operator*** () const
- CubeSetIterator & **operator++** ()
- CubeSetIterator **operator++** (int)

Friends

- class **CubeSet**
- bool **operator==** (CubeSetIterator const &lhs, CubeSetIterator const &rhs)
- bool **operator!=** (CubeSetIterator const &lhs, CubeSetIterator const &rhs)

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/DDECubeSet.h

7.9 capd::ddes::CubicIkeda< ScalarSpec, ParamSpec > Class Template Reference

`#include <SampleEqns.h>`

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **CubicIkeda** (ParamType a)
- **CubicIkeda** (CubicIkeda const &orig)
- **CubicIkeda** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- CubicIkeda & **operator=** (CubicIkeda const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **a**

7.9.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >

A model map of what I expect from a Map R^m -> R^n to be good for use with DDE codes.

7.9.2 Member Function Documentation

7.9.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::dimension ( )  [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.9.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline],
[static]
```

number of parameters to fully configure equation

7.9.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::CubicIkeda< ScalarSpec, ParamSpec >::imageDimension ( )  [inline],
[static]
```

output dimension of the internal map

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

7.10 capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > Class Template Reference

```
#include <FunctionalMap.h>
```

Public Types

- typedef SolutionCurveSpec **CurveType**
- typedef JetSpec **JetType**
- typedef CurveType::RealType **RealType**
- typedef CurveType::TimePointType **TimePointType**
- typedef CurveType::MatrixType **MatrixType**
- typedef CurveType::VectorType **VectorType**
- typedef CurveType::ScalarType **ScalarType**
- typedef CurveType::size_type **size_type**
- typedef CurveType::DataType **DataType**
- typedef std::vector< DataType > VariableStorageType
- typedef std::vector< VectorType > ValueStorageType
- typedef std::vector< std::vector< MatrixType > > JacobianStorageType

Public Member Functions

- virtual size_type imageDimension () const =0
- virtual VectorType operator() (const TimePointType &t0, const CurveType &x) const =0
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, size_type &out_admissible_order) const =0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const =0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const =0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual VectorType operator() (const CurveType &x) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual ~DDEBasicFunctionalMap ()

Static Public Member Functions

- static void convert (VariableStorageType const &u, ValueStorageType &v)
- static void deconvert (ValueStorageType const &u, VariableStorageType &v)

Protected Member Functions

- virtual void checkCurveDimension (CurveType const &x, std::string extra="") const

7.10.1 Detailed Description

template<typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType>
class capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >

- TODO: rewrite DOCS this is the interface of a Functional map, that can be evaluated only by "querying" the curve about its jets at various points in the past. it can produce jet at a given time t together with partial derivatives w.r.t. to internal variables from curve used in the computation by recurrent formula for a functional equation:

```
x'(t) = f(t, x)
```

where x is the solution curve in \R^d defined for times smaller than t (up to some maximal past time usually).

A good example of such a functional map is discrete delay case, e.g. F(t, x) = f(t, x(t), x(t-tau)), f : \R \times \R^{2d}->\R (easily extended to more delays). (Probably) the functionals that include integral over the past: F(t, x) = f(t, \int_{t-tau}^{t} x(s) ds) will also fall into this category.

The concrete implementations will be available later.

Implementation note: when implementing concrete class of this interface, do not forget to put using BaseClass↩ ::operator(); using BaseClass::computeDDECoefficients; in the public part of your class to gain access to default implementations of some functions.

TODO: (NOT URGENT, FUTURE, RETHINK) it would be better not to pass containers to computeDDECoefficients, but maybe iterators to given types? It could be used to compute in a given place, i.e. at an already created Jet?

7.10.2 Member Typedef Documentation

7.10.2.1 JacobianStorageType

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef std::vector< std::vector<MatrixType> > capd::ddes::DDEBasicFunctionalMap< Solution↩
CurveSpec, JetSpec >::JacobianStorageType
```

This type will store partial derivatives with respect to variables from input curve used in the computations. Each matrix will be of M(d, d) type.

7.10.2.2 ValueStorageType

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef std::vector< VectorType > capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, Jet↩
Spec >::ValueStorageType
```

similar to VariableStorageType but does not care about the internal representation. It will be used to store the Jet coefficients only in output (we do not need to care about the time point and we can freely change order (i.e. size of the ValueStorageType)

7.10.2.3 VariableStorageType

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
typedef std::vector< DataType > capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec
>::VariableStorageType
```

This type will store all variables from the input curve used to evaluate the map and to produce recurrently the jet at a given time. It will be stored in native SetType to allow usage of Lohner-type algorithms for controlling wrapping effect later. Each element of this collection would be a set in \R^d dimensional space.

7.10.3 Constructor & Destructor Documentation

7.10.3.1 ~DDEBasicFunctionalMap()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::~DDEBasicFunctionalMap
( )  [inline], [virtual]
```

virtual destructor for warning suppresion

7.10.4 Member Function Documentation

7.10.4.1 checkCurveDimension()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::checkCurve↩
Dimension (
            CurveType const & x,
            std::string extra = "" ) const  [inline], [protected], [virtual]
```

helper function to check. TODO: (NOT URGENT, FUTURE) make a switch to turn this off for speed.

7.10.4.2 collectComputationData()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::collectComputation↩
Data (
            const TimePointType & t0,
            const TimePointType & th,
            const RealType & dt,
            const CurveType & x,
            VariableStorageType & out_u,
            size_type & out_admissible_order ) const  [pure virtual]
```

Collects all data from curve necessary for the computation of the value. This does not collect ENCLOSURE DATA required in rigorous computations.

We use collection of DataType items as the output, we ASSUME that DataType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecesary computations (i.e. Jac Phi would be 0 and we would do $0 * C * r0$). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out_u a finite-dimensional collection of data that is used to compute the map Param: admissible_order will tell how high order of expansion is able to produce with data in u

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

7.10.4.3 computeDDECoefficients() [1/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
            const CurveType & curve,
            ValueStorageType & coeffs ) const  [inline], [virtual]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.10.4.4 computeDDECoefficients() [2/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
            const CurveType & curve,
            ValueStorageType & coeffs,
            VariableStorageType & u,
            JacobianStorageType & Du ) const  [inline], [virtual]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.10.4.5 computeDDECoefficients() [3/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
             const RealType & t0,
             const ValueStorageType & u,
             ValueStorageType & coeffs ) const  [pure virtual]
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

7.10.4.6 computeDDECoefficients() [4/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
             const RealType & t0,
             const ValueStorageType & u,
             ValueStorageType & coeffs,
             JacobianStorageType & Du ) const  [pure virtual]
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes $\frac{\partial coeffs[i]}{\partial u}$ (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

7.10.4.7 computeDDECoefficients() [5/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
             const TimePointType & t0,
             const CurveType & x,
             ValueStorageType & coeffs ) const  [inline], [virtual]
```

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs.↩ resize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE↩ Coefficients(..., u, ...) call to pure virtual function.

7.10.4.8 computeDDECoefficients() [6/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients (
             const TimePointType & t0,
             const CurveType & x,
             ValueStorageType & coeffs,
             VariableStorageType & u,
             JacobianStorageType & Du ) const  [inline], [virtual]
```

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
    Du[k] = \frac{\partial coeffs[k]}{\partial u}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

```
    Du[k][j] = \frac{\partial coeffs[k]}{\partial u[j]}
```

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

7.10.4.9 convert()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
static void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::convert (
             VariableStorageType const & u,
             ValueStorageType & v )  [inline], [static]
```

helper to convert from collection of sets to collection of vectors

7.10.4.10 deconvert()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
static void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::deconvert (
            ValueStorageType const & u,
            VariableStorageType & v )  [inline], [static]
```

helper to convert from collection of sets to collection of vectors NOTE: name is changed, as ValueStorageType might be VariableStorageType in some instances, and this would make compilation ambiguous.

7.10.4.11 imageDimension()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual size_type capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::image↩
Dimension ( ) const  [pure virtual]
```

output dimension of the map. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implemented in capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, typename SolutionCurveSpec::JetType >.

7.10.4.12 operator()() [1/2]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual VectorType capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::operator()
(
            const CurveType & x ) const  [inline], [virtual]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.10.4.13 operator()() [2/2]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual VectorType capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::operator()
(
            const TimePointType & t0,
            const CurveType & x ) const  [pure virtual]
```

computes value of the map at a given time point for a given solution curve. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implemented in capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Functional↩Map.h

7.11 capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec > Class Template Reference

`#include <DDEBasicPoincareMap.h>`

Public Types

- typedef DDEBasicPoincareMap< DynSysSpec, SectionSpec > **Class**
- typedef DynSysSpec **DynSysType**
- typedef SectionSpec **SectionType**
- typedef DynSysSpec::CurveType **CurveType**
- typedef CurveType::TimePointType **TimePointType**
- typedef CurveType::VectorType **VectorType**
- typedef CurveType::MatrixType **MatrixType**
- typedef CurveType::ScalarType **ScalarType**
- typedef CurveType::RealType **RealType**
- typedef DynSysType::JacobianStorageType **JacobianStorageType**
- typedef CurveType::size_type **size_type**

Public Member Functions

- **DDEBasicPoincareMap** (DDEBasicPoincareMap const &other)
- **DDEBasicPoincareMap** (DynSysType &dynsys, SectionType &section, CrossingDirection direction=Crossing↩ Direction::Both, int reqSteps=0, int maxSteps=-1, double binsearchEpsilon=DEFAULT_BINSEARCH_EP↩ SILON)
- void operator() (CurveType &curve, CurveType &on_section, RealType &out_approachTime)
- void operator() (CurveType &in_out_curve, CurveType &in_out_Pcurve, RealType &out_approachTime, VectorType &out_x, VectorType &out_Px, VectorType &out_fPx, MatrixType &out_V, MatrixType &out_DP)
- void setInitialV (CurveType &in_out_curve)
- void setInitialV (CurveType &in_out_curve, MatrixType const &V)
- void setCurrentV (CurveType &in_out_curve, MatrixType const &V)
- CurveType operator() (CurveType const &X)
- DDEBasicPoincareMap & **setDirection** (CrossingDirection direction)
- CrossingDirection **getDirection** ()
- DDEBasicPoincareMap & **setMaxSteps** (int maxSteps)
- int **getMaxSteps** ()
- int **getMaximumSteps** ()
- DDEBasicPoincareMap & **setRequiredSteps** (int requiredSteps)
- int **getRequiredSteps** ()
- void integrateUntilSectionCrossing (CurveType &curve, RealType &timeBeforeSection, SingleStepFn stepFn)
- void findCrossingTime (CurveType const &curve, CurveType &requested, RealType &crossingTime, int test↩ Direction)
- CrossingDirection detectCrossingDirection (CurveType const &curve)
- RealType **getLastEpsilonTime** () const
- RealType **getLastTimeBeforeSection** () const
- RealType **getLastReachTime** () const
- JacobianStorageType **getLastVariational** () const
- int **getLastStepsAfterSection** ()
- bool **isNormalizeVariational** () const
- Class & **setNormalizeVariational** (bool value)

Protected Member Functions

- void **checkSteps** ()
- void **extractVariationalMatrix** (CurveType const &curve, MatrixType &fullV, MatrixType &reducedV, std::vector< size_type > reducedShape, int p_howFar=-1) const
- void **extractVariationalMatrix** (CurveType const &curve, MatrixType &fullV) const

Protected Attributes

- DynSysType & **m_dynsys**
- SectionType & **m_section**
- CrossingDirection **m_direction**
- int **m_requiredSteps**
- int **m_maxSteps**
- int **m_steps**
- double **m_binsearchEpsilon**
- RealType **m_lastTimeBeforeSection**
- RealType **m_lastEpsilonTime**
- JacobianStorageType **m_variational**
- bool **m_normalizeVariational**
- std::vector< size_type > **m_storedInitialShape**

7.11.1 Detailed Description

template<typename DynSysSpec, typename SectionSpec>
class capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >

Implementation of Poincare Map for use with nonrigorous DDESolvers.

In theory should work with any compatible DDEDynSys and Section.

7.11.2 Member Function Documentation

7.11.2.1 detectCrossingDirection()

```
template<typename DynSysSpec , typename SectionSpec >
CrossingDirection capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::detectCrossing↩
Direction (
            CurveType const & curve )  [inline]
```

warning: might be time consuming, it (might) integrate the solution... it assumes curve is on section.

7.11.2.2 findCrossingTime()

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::findCrossingTime (
            CurveType const & curve,
            CurveType & requested,
            RealType & crossingTime,
            int testDirection )  [inline]
```

testDirection = -1 for before section testDirection = +1 for after section binsearch for now

7.11.2.3 integrateUntilSectionCrossing()

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::integrateUntilSectionCrossing
(
            CurveType & curve,
            RealType & timeBeforeSection,
            SingleStepFn stepFn )  [inline]
```

it extends the solution until the crossing with the section. Also, it takes into consideration the required number of steps (should be set-up before) It also moves away of section is initially curve crosses the section.

WARNING: it does not check for sanity of argument as of now, so it might cause runtime-errors if the proof is not well prepared.

7.11.2.4 operator()() [1/3]

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::operator() (
            CurveType & curve,
            CurveType & on_section,
            RealType & out_approachTime )  [inline]
```

in curve there is a curve that reaches first full step after the section in on_section there is the image on the section, as close to section as possible on_section must be initialized with the desired "length" (time interval) and "order" of the representation. on_section must be compatible (length, order) with the section

7.11.2.5 operator()() [2/3]

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::operator() (
            CurveType & in_out_curve,
            CurveType & in_out_Pcurve,
            RealType & out_approachTime,
            VectorType & out_x,
            VectorType & out_Px,
            VectorType & out_fPx,
            MatrixType & out_V,
            MatrixType & out_DP )  [inline]
```

For other parameters and base description, see the other operator()

This is more complicated than the case of ODEs, but for technical reasons it is most desirable to work with vector/matrix representations of the objects when dealing with variational equation on the coefficients.

NOTE: CurveType must be defined with DataType that allows to handle extra Matrix data. I have special structures for this. CHeck examples and helper classes. Thanks to C++ lazy evaluation of templates this also works on the basic data structure as long as you do not call the function in the program.

out_x is vector representation of input curve out_Px is vector representation of the image on the section (Pcurve) out_fPx is the value of the "Vector Field" of the equation as defined in the Banach space in practice, if c = Px is a curve in C([-tau, 0], \R^d), then out_fPx = c'. This is theoretically correct when c is on the C^1 solutions manifold, i.e. c(0^-) == f(c) where f is the r.h.s. of the DDE. This is true for any initial curve, if the reachTime > delay. See works for more explanation. out_V is the variational equation solution on the coefficients, i.e. d\varphi(t_p, x)/dx (in sense of the coefficients defining x) therefore it is of dimension out_x.dimension() x out_x.dimension() out_DP is the dP/dx (x) (derivative of P w.r.t. initial data, computed at point x) out_DP is obtained from out_V by some correction on the reach time t_p(x) in the neighbourhood of x. i.e. dP/dx (x) = d(\varphi(t_p(x), x))/dx = = \partial(\varphi)/\partial x (t_p(x), x) +

- \partial(\varphi)/\partial t (t_p(x), x) * d(t_p)/dx (x) = V + f(Px) * dt_p/dx (x) Please note that f(Px) is "vertical" and dt_p/dx (x) is "horizontal" vector so their product is a full matrix of desired dimension! We see that f(Px) is in out_fPx, V in out_V, then the last term is obtained by differentiation of the section condition equation:

s(\varphi(t_p(x), x)) == 0

so we get (* means scalar product in \R^M where needed, M = out_x.dimension())

\gradient s * (V + f(Px) * dt_p/dx (x)) == 0 so dt_p/dx (x) = - ({\gradient s} * V) / ({\gradient s} * f(Px))

Note, that ({\gradient s} * f(Px)) is simply a scalar, and it needs to be not equal 0 for the formula to make sense. This is usual notion of the section to be transversal to the flow at x.

Finally, note that x, Px, fPx can be used in .set_x() to a curve of a proper structure (important!) to get interesting data. The proper structure is stored in Pcurve, i.e. usually, one can do: curve_x = Pcurve; curve_x.set_x(out_x); // guaranteed: curve_x == curve.subcurve([-tau, 0]) curve_Px = Pcurve; curve_Px.set_x(out_Px); // guaranteed: curve_Px == Pcurve curve_fPx = Pcurve; curve_fPx.set_x(out_fPx); // guaranteed: curve_fPx == (Pcurve)'

IMPORTANT: curve_fPx will be of one order higher than simply trying to compute Pcurve.dt()! IMPORTANT: as .dt() loses one order. So Pcurve.dt() is not a proper way to get "force field" IMPORTANT: on curve_Px, as the dimensions will be different (orders different). IMPORTANT: Therefore, use curve_fPx to get the value of the "force field" on curve_Px.

7.11.2.6 operator()() [3/3]

```
template<typename DynSysSpec , typename SectionSpec >
CurveType capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::operator() (
            CurveType const & X )  [inline]
```

to be compatible with standard notion and CAPD interface, It makes following assumptions:

- X and PX will have the same structure (i.e. over same grid points and jets of the same order).

NOTE: you can retrieve information on reach time and epsilon step, but you cannot acquire the solution on the whole time. If you need solution over the full integration time, then use other operator().

7.11.2.7 setCurrentV()

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::setCurrentV (
            CurveType & in_out_curve,
            MatrixType const & V )  [inline]
```

Experimental...

7.11.2.8 setInitialV() [1/2]

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::setInitialV (
            CurveType & in_out_curve )  [inline]
```

helper function, sets to Id the Variational matrix w.r.t. initial coefficients in curve TODO: DRY

7.11.2.9 setInitialV() [2/2]

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEBasicPoincareMap< DynSysSpec, SectionSpec >::setInitialV (
            CurveType & in_out_curve,
            MatrixType const & V )  [inline]
```

sets initial V to a given Matrix. Matrix must be of a good shape TODO: DRY

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ BasicPoincareMap.h

7.12 capd::ddeshelper::DDECompareHelper< VectorSpec > Class Template Reference

```
#include <DDECompareHelper.h>
```

Public Types

- typedef VectorSpec **VectorType**
- typedef VectorType::size_type **size_type**
- typedef VectorType::ScalarType **ScalarType**

Public Member Functions

- **DDECompareHelper** (VectorType const &orig, VectorType const &other)
- void **printRelation** (int i, std::ostream &out) const
- void **printRelations** (std::ostream &out) const
- void **printSummary** (std::ostream &out) const
- std::string **inlineSummary** () const
- **operator std::string** () const
- int **getSubsetsCount** () const
- int **getSupsetsCount** () const
- int **getMissLeftCount** () const
- int **getMissRightCount** () const
- int **getMissCount** () const
- int **getIntersectionLeftCount** () const
- int **getIntersectionRightCount** () const
- int **getIntersectionsCount** () const
- int getUnknownsCount () const
- bool **isSubset** (size_type i) const
- bool **isSupset** (size_type i) const
- bool **isMissLeft** (size_type i) const
- bool **isMissRight** (size_type i) const
- bool **isMiss** (size_type i) const
- bool **isIntersectLeft** (size_type i) const
- bool **isIntersectRight** (size_type i) const
- bool **isIntersection** (size_type i) const
- bool **isUnknown** (size_type i) const

Friends

- std::ostream & **operator**<< (std::ostream &out, DDECompareHelper< VectorSpec > const &comp)

7.12.1 Detailed Description

template<typename VectorSpec>
class capd::ddeshelper::DDECompareHelper< VectorSpec >

allows to easily check how the other vector relates to the original.

7.12.2 Member Function Documentation

7.12.2.1 getUnknownsCount()

```
template<typename VectorSpec >
int capd::ddeshelper::DDECompareHelper< VectorSpec >::getUnknownsCount ( ) const  [inline]
```

this should always return 0, otherwise some serious errors happen

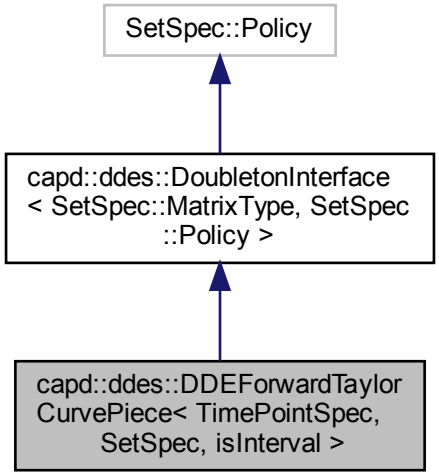
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/DDECompareHelper.h

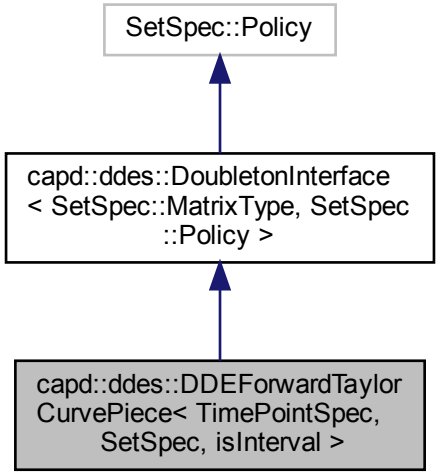
7.13 capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > Class Template Reference

`#include <DDEForwardTaylorCurvePiece.h>`

Inheritance diagram for capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >:



Collaboration diagram for capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >:



Public Types

- typedef SetSpec **DoubletonStorageType**
- typedef SetSpec **SetType**
- typedef SetType::MatrixType **MatrixType**
- typedef SetType::Policy **Policy**
- typedef SetType::QRPolicy **QRPolicy**
- typedef DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > **Class**
- typedef DoubletonInterface< MatrixType, Policy > **BaseClass**
- typedef SetType::VectorType **VectorType**
- typedef SetType::ScalarType **ScalarType**
- typedef SetType::size_type **size_type**
- typedef ScalarType **RealType**
- typedef TimePointSpec **TimePointType**
- typedef Class **JetType**
- typedef DoubletonStorageType * **iterator**
- typedef capd::ddes::GenericJet< TimePointType, VectorType, VectorType, MatrixType > **ExternalJetType**

Public Member Functions

- DDEForwardTaylorCurvePiece & operator= (DDEForwardTaylorCurvePiece const &orig)
- DDEForwardTaylorCurvePiece (TimePointType t0=TimePointType())
- DDEForwardTaylorCurvePiece (DDEForwardTaylorCurvePiece const &orig)
- DDEForwardTaylorCurvePiece (TimePointType t0, size_type dimension, size_type order)
- DDEForwardTaylorCurvePiece (TimePointType t0, size_type order, const VectorType &v, size_type N0=0)
- DDEForwardTaylorCurvePiece (TimePointType t0, size_type order, const VectorType &v, VectorType *r0, bool passOwnership=false)
- DDEForwardTaylorCurvePiece (TimePointType t0, VectorType const *it, VectorType const *itEnd)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs)
- DDEForwardTaylorCurvePiece (TimePointType t0, size_type order, SetType &value)
- DDEForwardTaylorCurvePiece (TimePointType t0, size_type order, const SetType &value)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs, std::vector< MatrixType > Cs, VectorType *r0, bool passR0Ownership, std::vector< MatrixType > Bs, std::vector< VectorType > rs, VectorType *Xi, bool passXiOwnership)
- DDEForwardTaylorCurvePiece (TimePointType t0, std::vector< VectorType > coeffs, std::vector< MatrixType > Cs, VectorType *r0, bool passR0Ownership, std::vector< MatrixType > Bs, std::vector< MatrixType > invBs, std::vector< VectorType > rs, VectorType *Xi, bool passXiOwnership)
- size_type dimension () const
- size_type order () const
- size_type storageDimension () const
- size_type storageN0 () const
- VectorType get_x () const
- MatrixType get_C () const
- MatrixType get_B () const
- VectorType get_r () const
- VectorType get_r0 () const
- BaseClass & set_x (VectorType const &x)
- BaseClass & set_C (MatrixType const &C)
- BaseClass & set_r0 (VectorType const &r0)
- BaseClass & set_Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set_Binv (MatrixType const &Binv)
- BaseClass & set_B (MatrixType const &B)
- BaseClass & set_r (VectorType const &r)
- virtual Class & mul (ScalarType const &c)

< *see Doubleton interface*

- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- ScalarType dot (ExternalJetType const &s) const
- DDEForwardTaylorCurvePiece dt (size_type n=1) const
- VectorType evalAtDelta (const RealType &delta_t) const
- VectorType eval (const RealType &t) const
- VectorType evalCoeffAtDelta (size_type n, const RealType &delta_t) const
- VectorType evalCoeff (size_type n, const RealType &t) const
- VectorType taylorAtDelta (const RealType &delta_t) const
- VectorType taylor (const RealType &t) const
- VectorType summaAtDelta (const RealType &delta_t) const
- VectorType summa (const RealType &t) const
- JetType jetAt (const TimePointType &t) const
- JetType jetAt (const RealType &t) const
- void taylorAtDelta (const RealType &delta_t, SetType &out) const
- void taylor (const RealType &t, SetType &out) const
- void evalAtDelta (const RealType &delta_t, SetType &out) const
- void eval (const RealType &t, SetType &out) const
- void evalCoeffAtDelta (size_type n, const RealType &delta_t, SetType &out) const
- void evalCoeff (size_type n, const RealType &t, SetType &out) const
- DDEForwardTaylorCurvePiece midCurve () const
- bool isMidCurve () const
- std::string show () const
- iterator beginJet ()
- iterator endJet ()
- iterator backJet ()
- const_iterator beginJet () const
- const_iterator endJet () const
- const_iterator backJet () const
- DoubletonStorageType & operator[ ] (size_type k)
- DoubletonStorageType const & operator[ ] (size_type k) const
- Class & setAsConstant (SetType &value)
- Class & setAsConstant (const SetType &value, VectorType ∗overwrite_r0=NULL, bool passOwnership=false)
- Class & setAsConstant (VectorType const &value, size_type N0=0)
- Class & setAsConstant (VectorType const &value, VectorType ∗r0, bool passOwnership=false)
- Class & setT0 (TimePointType const &t0)
- TimePointType getT0 () const
- TimePointType t0 () const
- void set_Xi (VectorType const &xi)
- void set_Xi (VectorType ∗xi, bool passOwnership=false)
- VectorType get_Xi () const
- VectorType ∗ take_Xi ()
- VectorType ∗ take_r0 ()
- Class & set_r0 (VectorType ∗r0, bool passOwnership=false)
- virtual ∼DDEForwardTaylorCurvePiece ()
- virtual void reinitialize (size_type d, size_type N0)
- virtual VectorType makeStorage_x () const
- virtual MatrixType makeStorage_C () const

  < *see Doubleton interface*
- virtual VectorType makeStorage_r0 () const

  < *see Doubleton interface*
- virtual MatrixType makeStorage_B () const

  < *see Doubleton interface*

- virtual VectorType makeStorage_r () const
  < *see Doubleton interface*
- virtual VectorType hull () const
  < *see Doubleton interface*
- virtual VectorType midPoint () const
  < *see Doubleton interface*
- virtual ScalarType dot (VectorType const &v) const
  < *see Doubleton interface*

Static Public Member Functions

- static std::string badge ()

Public Attributes

- const typedef DoubletonStorageType * **const_iterator**

Protected Member Functions

- void dimCheck (const VectorType *const v)
- void dimCheckC (const MatrixType *const C)
- void dimCheckB (const MatrixType *const B)
- void deallocateJet ()
- void deallocateXi ()
- void deallocateR0 ()
- void deallocate ()
- void allocateJet ()
- void allocateXi ()
- void allocateR0 (size_type N0=0)
- void allocate (size_type N0=0)
- void reallocateJet ()
- void reallocateXi ()
- void reallocateR0 (size_type N0=0)
- void reallocate (size_type N0=0)
- void updateCommonR0 ()
- template<typename IteratorType >
  void setupFromData (IteratorType it, IteratorType itEnd)
- template<typename XItSpec , typename CItSpec , typename BItSpec , typename RItSpec >
  void setupFromData (XItSpec xit, XItSpec xend, CItSpec Cit, CItSpec Cend, VectorType *r0, BItSpec Bit, BItSpec Bend, BItSpec invBit, BItSpec invBend, RItSpec rit, RItSpec rend, VectorType *Xi)

Protected Attributes

- TimePointType **m_t0**
- size_type **m_dimension**
- size_type **m_order**
- DoubletonStorageType * **m_jet_at_t0**
- VectorType * **m_r0**
- bool **m_r0_owner**
- VectorType * **m_Xi**
- bool **m_Xi_owner**

Friends

- std::ostream & **operator**<< (std::ostream &out, DDEForwardTaylorCurvePiece const &jet)
- std::istream & **operator**>> (std::istream &in, DDEForwardTaylorCurvePiece &jet)

7.13.1 Detailed Description

template<typename TimePointSpec, typename SetSpec, bool isInterval = capd::TypeTraits<typename SetSpec::MatrixType::ScalarType>::isInterval>
class capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >

A class to represent a part of some curve x over $[t0, t1)$ given by:

$(J^{[N]}_{t0}x)(t) + (n+1) * \int_{t0}^{t} \xi(s) (t0-s)^{N} ds$

where $\xi(s) \in R \subset \mathbb{R}^d$

in the code below: d = dimension n = order

If the interval version is used, then the class can represent sets of Curves, so that $J^{[N]}_{t0} \in x + A * r0 + Q * r$, (x is here a middle point of the set and Jet is interpreted as a collection of coefficients). $r0$ is some vector of any given dimension, compatible with A. We denote the dimension of $r0$ as $N0$. Q is a diagonal block matrix, so that Q_ii is orthonormal $d \times d$ matrix and and Q is 0 elsewhere. Vector r is of dimension $d * n$. we store matrices row-wise, ie in the code below m_A[k] represents a $d \times N0$ matrix and m_Q[k] is Q_kk a $d \times d$ matrix. Finally, we store r and x again component-wise, so that in code below m_r[k] and $m_x[k]$ is a d-dimensional vector. Therefore we have: $(J^{[N]}_{t0}x)_{[k]} \in m_x[k] + m_A[k] * r0 + m_Q[k] * m_r[k]$

NOTE: we store m_r0 as a one $N0$-dimensional vector, but for optimization purposes we hold the variable as a pointer - to allow sharing of this vector among many instances (important in DDESolutionCurve)

TODO: (NOT URGENT) extract abstract interface that has eval, taylor, summa, etc? TODO: (NOT URGENT) base interface is already done as GenericJet, it has taylor (or eval). TODO: (NOT URGENT) This should inherit Jet structure from there and add summa and taylor, and reload eval.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 DDEForwardTaylorCurvePiece() [1/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0 = TimePointType() )
```

creates an empty curve of order 0 and dimension 0, by default at the point $t_0=0$

7.13.2.2 DDEForwardTaylorCurvePiece() [2/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > const & orig )
```

copy constructor. if orig has a shared r0 and/or Xi then it will share it with the copy (this) Otherwise, it assures that the r0 / Xi is copied accordingly and the jet updated if necessary.

7.13.2.3 DDEForwardTaylorCurvePiece() [3/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            size_type dimension,
            size_type order )
```

makes a constant function of the vector value 0

7.13.2.4 DDEForwardTaylorCurvePiece() [4/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            size_type order,
            const VectorType & v,
            size_type N0 = 0 )
```

makes a constant function of the vector value v

7.13.2.5 DDEForwardTaylorCurvePiece() [5/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            size_type order,
            const VectorType & v,
            VectorType * r0,
            bool passOwnership = false )
```

makes a constant function of the vector value v

7.13.2.6 DDEForwardTaylorCurvePiece() [6/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            VectorType const * it,
            VectorType const * itEnd )
```

fills vector with data. Determine order from the size of data.

7.13.2.7 DDEForwardTaylorCurvePiece() [7/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            std::vector< VectorType > coeffs )
```

fills vector with data. Determine order from the size of data.

7.13.2.8 DDEForwardTaylorCurvePiece() [8/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            size_type order,
            SetType & value )
```

makes a curve piece of constant function, located at t0, of a given order and of given value. The r0 will be taken from the sets r0 (and no ownership transfer occurs).

7.13.2.9 DDEForwardTaylorCurvePiece() [9/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            size_type order,
            const SetType & value )
```

makes a curve piece of constant function, located at t0, of a given order and of given value. The r0 will be taken from the sets r0 (as copy).

7.13.2.10 DDEForwardTaylorCurvePiece() [10/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            std::vector< VectorType > coeffs,
            std::vector< MatrixType > Cs,
            VectorType * r0,
            bool passR0Ownership,
            std::vector< MatrixType > Bs,
            std::vector< VectorType > rs,
            VectorType * Xi,
            bool passXiOwnership )
```

almost complete constructor, setups everything from scratch (computes invB)

7.13.2.11 DDEForwardTaylorCurvePiece() [11/11]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::DDEForwardTaylorCurvePiece
(
            TimePointType t0,
            std::vector< VectorType > coeffs,
            std::vector< MatrixType > Cs,
            VectorType * r0,
            bool passR0Ownership,
            std::vector< MatrixType > Bs,
            std::vector< MatrixType > invBs,
            std::vector< VectorType > rs,
            VectorType * Xi,
            bool passXiOwnership )
```

most complete constructor, setups everything from scratch

7.13.2.12 ~DDEForwardTaylorCurvePiece()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::~DDEForwardTaylorCurveP
( )  [inline], [virtual]
```

standard thing

7.13.3 Member Function Documentation

7.13.3.1 affineTransform()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
BaseClass& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::affineTransform (
            MatrixType const & M,
            VectorType const & v )  [inline]
```

see doubleton interface

7.13.3.2 allocate()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocate (
            size_type N0 = 0 )  [inline], [protected]
```

technical, manages allocation of resources

7.13.3.3 allocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocateJet
( )  [inline], [protected]
```

technical, manages deallocation of resources

7.13.3.4 allocateR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocateR0
(
            size_type N0 = 0 )  [inline], [protected]
```

technical, manages allocation of resources

7.13.3.5 allocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::allocateXi
( )  [inline], [protected]
```

technical, manages allocation of resources

7.13.3.6 backJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::backJet
( )  [inline]
```

iterator to the last element in coefficients. Standard thing in C++

7.13.3.7 backJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::backJet ( ) const  [inline]
```

iterator to the last element in coefficients. Standard thing in C++

7.13.3.8 badge()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
static std::string capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval
>::badge ( )  [inline], [static]
```

should be one word

7.13.3.9 beginJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::begin↩
Jet ( )  [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

7.13.3.10 beginJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::beginJet ( ) const  [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

7.13.3.11 deallocate()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate
( )  [inline], [protected]
```

technical, manages deallocation of resources

7.13.3.12 deallocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate↩
Jet ( )  [inline], [protected]
```

technical, manages deallocation of resources

7.13.3.13 deallocateR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate↩
R0 ( )  [inline], [protected]
```

technical, manages deallocation of resources

7.13.3.14 deallocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::deallocate↩
Xi ( )  [inline], [protected]
```

technical, manages deallocation of resources

7.13.3.15 dimCheck()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dimCheck (
            const VectorType *const v )  [inline], [protected]
```

checks if the dimension of v is compatible with current set

7.13.3.16 dimCheckB()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dimCheckB (
            const MatrixType *const B )  [inline], [protected]
```

checks if the dimension of B is compatible with current set B matrix

7.13.3.17 dimCheckC()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dimCheckC (
            const MatrixType *const C )  [inline], [protected]
```

checks if the dimension of C matrix is compatible with current set C matrix (might be swapped)

7.13.3.18 dimension()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dimension
( ) const  [inline]
```

dimension of the value space of the Jet (i.e. $J : \R \to \R^{dimension}$)

7.13.3.19 dot()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
ScalarType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::dot (
            ExternalJetType const & s ) const  [inline]
```

computes dot product of this with JetType (does not take Xi part into consideration!)

This is a simple dot product in vector space. Consider implementing some other like integral (but it should be equivalent more or less becouse of polynomials)

7.13.3.20 dt()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >::dt (
            size_type n = 1 ) const
```

returns Forward Taylor Jet of the derivative of order n w.r.t. time, n must be smaller than the order

7.13.3.21 endJet() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::endJet
( )  [inline]
```

iterator to the past the order place. Standard thing in C++

7.13.3.22 endJet() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
const_iterator capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::endJet ( ) const  [inline]
```

iterator to the past the order place. Standard thing in C++

7.13.3.23 eval() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::eval
(
            const RealType & t ) const  [inline]
```

evaluates the jet at time t (please note that it is true time, not some small h w.r.t. jet.t0().

7.13.3.24 eval() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::eval (
            const RealType & t,
            SetType & out ) const  [inline]
```

evaluates the jet at time t0 + delta_t, delta_t should be positive! Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.25 evalAtDelta() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::evalAtDelta (
            const RealType & delta_t ) const
```

evaluates the jet at time t0 + delta_t , delta_t should be positive!

7.13.3.26 evalAtDelta() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::evalAtDelta
(
            const RealType & delta_t,
            SetType & out ) const
```

evaluates the jet at time t0 + delta_t, delta_t should be positive. Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.27 evalCoeff() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::evalCoeff (
            size_type n,
            const RealType & t ) const  [inline]
```

helper. Evaluates n-th derivative (more precisely j^[n] = j^(n)/n! - it is used in the algorithm most often) of this jet at t. It should be faster than jet.dt(n).eval(t)/n!

7.13.3.28 evalCoeff() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::evalCoeff (
            size_type n,
            const RealType & t,
            SetType & out ) const  [inline]
```

evaluates the jet at time t0 + delta_t, delta_t should be positive! Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.29 evalCoeffAtDelta() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::evalCoeffAtDelta (
            size_type n,
            const RealType & delta_t ) const
```

helper. Evaluates n-th derivative (more precisely j^[n] = j^(n)/n! - it is used in the algorithm most often) of this jet at t. It should be faster than jet.dt(n).evalAtDelta(t)/n!. The delta_t should be positive!

7.13.3.30 evalCoeffAtDelta() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::evalCoeff↩
AtDelta (
            size_type n,
            const RealType & delta_t,
            SetType & out ) const
```

evaluates the jet at time t0 + delta_t, delta_t should be positive!

- Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.31 get_B()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::MatrixType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::get_B
```

see doubleton interface

7.13.3.32 get_C()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::MatrixType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::get_C
```

see doubleton interface

7.13.3.33 get_r()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::get_r
```

see doubleton interface

7.13.3.34 get_r0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::get_r0
```

see doubleton interface

7.13.3.35 get_x()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::get_x
```

see doubleton interface

7.13.3.36 get_Xi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::get↩
_Xi ( ) const  [inline]
```

returns a value of Xi

7.13.3.37 getT0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
TimePointType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::getT0 ( ) const  [inline]
```

returns time at which this jet is located

7.13.3.38 isMidCurve()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
bool capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::isMidCurve
( ) const  [inline]
```

returns true, if the diamater of Taylor part is 0

7.13.3.39 jetAt() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
JetType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::jetAt (
            const RealType & t ) const  [inline]
```

returns jet at t for a function represented by this. It is rigorous, so it returns validated estimates (overestimates)

7.13.3.40 jetAt() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
JetType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::jetAt (
            const TimePointType & t ) const  [inline]
```

returns jet at t for a function represented by this. It is rigorous, so it returns validated estimates (overestimates)

7.13.3.41 makeStorage_x()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x
[inline]
```

makes a vector that can store x part

7.13.3.42 midCurve()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >::midCurve
```

returns jet with Taylor part of diameter 0 and the same Xi

7.13.3.43 mul()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::mul (
            ScalarType const & c )  [inline], [virtual]
```

< see Doubleton interface

multiply set by a scalar. Should take set structure into consideration.

7.13.3.44 operator=()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > & capd::ddes::DDEForwardTaylorCurvePiece<
TimePointSpec, SetSpec, isInterval >::operator= (
            DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval > const & orig )
```

copy operator. Works as in case of copy constructor (see doc there)

7.13.3.45 operator[]() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
DoubletonStorageType& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, is↩
Interval >::operator[] (
            size_type k )  [inline]
```

returns n-th order Taylor coefficient.

7.13.3.46 operator[]() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
DoubletonStorageType const& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec,
isInterval >::operator[] (
            size_type k ) const  [inline]
```

returns n-th order Taylor coefficient.

7.13.3.47 order()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::order
( ) const  [inline]
```

order of the Taylor part of the Jet

7.13.3.48 reallocate()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::reallocate
(
            size_type N0 = 0 )  [inline], [protected]
```

technical, manages re-allocation of resources

7.13.3.49 reallocateJet()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::reallocate↩
Jet ( )  [inline], [protected]
```

technical, manages allocation of resources

7.13.3.50 reallocateR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::reallocate↩
R0 (
             size_type N0 = 0 )  [inline], [protected]
```

technical, manages re-allocation of resources

7.13.3.51 reallocateXi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::reallocate↩
Xi ( )  [inline], [protected]
```

technical, manages re-allocation of resources

7.13.3.52 reinitialize()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
virtual void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::reinitialize (
             size_type d,
             size_type N0 )  [inline], [virtual]
```

see base class

7.13.3.53 set_B()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_B (
             MatrixType const & B )
```

see doubleton interface

WARNING: we assume additionaly, that B has a block-diagonal form, with d x d diagonal blocks (d = dimension()).
the code will raise exception if off-block-diagonal element is found non zero.

7.13.3.54 set_Binv()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_Binv (
             MatrixType const & Binv )
```

see doubleton interface

WARNING: we assume additionaly, that B has a block-diagonal form, with d x d diagonal blocks (d = dimension()).
the code will raise exception if off-block-diagonal element is found non zero.

7.13.3.55 set_C()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_C (
            MatrixType const & C )
```

see doubleton interface

7.13.3.56 set_Cr0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_Cr0 (
            MatrixType const & C,
            VectorType const & r0 )
```

see doubleton interface

7.13.3.57 set_r()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_r (
            VectorType const & r )
```

see doubleton interface

7.13.3.58 set_r0() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::set_r0 (
            VectorType * r0,
            bool passOwnership = false )  [inline]
```

set the estimate on the r0 part of the set, by default id does not pass ownership. Updates jet pointers! Throws exception if r0 is of bad dimension.

7.13.3.59 set_r0() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_r0 (
            VectorType const & r0 )
```

see doubleton interface

7.13.3.60 set_x()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::BaseClass & capd::ddes::DDEForwardTaylorCur
TimePointSpec, SetSpec, isInterval >::set_x (
            VectorType const & x )
```

see doubleton interface

7.13.3.61 set_Xi() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::set_Xi (
            VectorType * xi,
            bool passOwnership = false )  [inline]
```

set the estimate on the \xi part of the jet, by default id does not pass ownership. Throws exception if Xi is of bad dimension.

7.13.3.62 set_Xi() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::set_Xi (
            VectorType const & xi )  [inline]
```

set the estimate on the \xi part of the jet, makes object to own it.

7.13.3.63 setAsConstant() [1/4]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setAs↩
Constant (
            const SetType & value,
            VectorType * overwrite_r0 = NULL,
            bool passOwnership = false )  [inline]
```

set this vector to represent a constant function f(t) = value for all t > t0, the r0 vector is taken from the value set.

7.13.3.64 setAsConstant() [2/4]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setAs↩
Constant (
            SetType & value )  [inline]
```

set this vector to represent a constant function f(t) = value for all t > t0, the r0 vector is taken from the value set.

7.13.3.65 setAsConstant() [3/4]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setAs↩
Constant (
            VectorType const & value,
            size_type N0 = 0 )  [inline]
```

set this vector to represent a constant function f(t) = value for all t > t0

7.13.3.66 setAsConstant() [4/4]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setAs↩
Constant (
            VectorType const & value,
            VectorType * r0,
            bool passOwnership = false )  [inline]
```

set this vector to represent a constant function f(t) = value for all t > t0

7.13.3.67 setT0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
Class& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setT0 (
            TimePointType const & t0 )  [inline]
```

set the base time of the jet

7.13.3.68 setupFromData() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
template<typename IteratorType >
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setupFrom↩
Data (
            IteratorType it,
            IteratorType itEnd )  [inline], [protected]
```

setup data from a generic form of Iterator (to collection of Vectors)

7.13.3.69 setupFromData() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
template<typename XItSpec , typename CItSpec , typename BItSpec , typename RItSpec >
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::setupFrom↩
Data (
            XItSpec xit,
            XItSpec xend,
            CItSpec Cit,
            CItSpec Cend,
            VectorType * r0,
            BItSpec Bit,
            BItSpec Bend,
            BItSpec invBit,
            BItSpec invBend,
            RItSpec rit,
            RItSpec rend,
            VectorType * Xi )  [inline], [protected]
```

Setup set from a rich set of data. Please note that you pass r0 and Xi as pointers, but not pass ownership flags You need to manage flags manually, outside the procedure. This is highly internal function, not meant to be used by the end-user.

7.13.3.70 show()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
std::string capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::show
```

more verbose output, more human-friendly. Not rigorous-friendly.

7.13.3.71 storageDimension()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::storage↩
Dimension ( ) const  [inline]
```

DoubletonStorageInterface: returns a dimension of the Jet as a Vector (sequence) to store all the coefficients (without Xi part)

7.13.3.72 storageN0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
size_type capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::storage↩
N0 ( ) const  [inline]
```

DoubletonStorageInterface: returns a dimension of the r0 vector

7.13.3.73 summa()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::summa
(
            const RealType & t ) const  [inline]
```

evaluates the integral of Xi part of the jet at t

7.13.3.74 summaAtDelta()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::summaAtDelta (
            const RealType & delta_t ) const
```

evaluates the integral of Xi part of the jet at t0 + delta_t

7.13.3.75 t0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
TimePointType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::t0
( ) const  [inline]
```

naming convention. See getT0()

7.13.3.76 take_r0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType* capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::take_r0 ( )  [inline], [virtual]
```

returns r0 and forgets that object is owner (user is responsible for deallocating r0)

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.13.3.77 take_Xi()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType* capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::take_Xi ( )  [inline]
```

returns Xi and forgets that object is owner (user is responsible for deallocating Xi)

7.13.3.78 taylor() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
VectorType capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::taylor (
            const RealType & t ) const  [inline]
```

evaluates the Taylor part of the jet at t

7.13.3.79 taylor() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::taylor (
            const RealType & t,
            SetType & out ) const  [inline]
```

evaluates the Taylor part of the jet at t, delta_t should be positive.

- Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.80 taylorAtDelta() [1/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::VectorType capd::ddes::DDEForwardTaylorCurv
TimePointSpec, SetSpec, isInterval >::taylorAtDelta (
            const RealType & delta_t ) const
```

evaluates the Taylor part of the jet at t0 + delta_t

7.13.3.81 taylorAtDelta() [2/2]

```
template<typename TimePointSpec , typename SetSpec , bool isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::taylorAt↩
Delta (
            const RealType & delta_t,
            SetType & out ) const
```

evaluates the Taylor part of the jet at t0 + delta_t, delta_t should be positive. Note: out should be set representing [0,0]^d. Procedure do not test for it!

7.13.3.82 translate()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
BaseClass& capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >↩
::translate (
            VectorType const & v )  [inline]
```

see doubleton interface

7.13.3.83 updateCommonR0()

```
template<typename TimePointSpec , typename SetSpec , bool isInterval = capd::TypeTraits<typename
SetSpec::MatrixType::ScalarType>::isInterval>
void capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSpec, isInterval >::update↩
CommonR0 ( )  [inline], [protected]
```

technical, takes care that all jets share the same r0

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ ForwardTaylorCurvePiece.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ ForwardTaylorCurvePiece.hpp

7.14 capd::ddes::DDEJetSection< CurveSpec, isInterval > Class Template Reference

Public Types

- typedef CurveSpec **CurveType**
- typedef CurveType::size_type **size_type**
- typedef CurveType::TimePointType **TimePointType**
- typedef CurveType::VectorType **VectorType**
- typedef CurveType::MatrixType **MatrixType**
- typedef CurveType::ScalarType **ScalarType**
- typedef GenericJet< TimePointType, VectorType, VectorType, MatrixType, isInterval > **JetType**
- typedef std::vector< JetType > **JetStorageType**
- typedef JetStorageType::const_iterator **const_iterator**
- typedef JetStorageType::iterator **iterator**
- typedef JetStorageType::const_reverse_iterator **const_reverse_iterator**
- typedef JetStorageType::reverse_iterator **reverse_iterator**

Public Member Functions

- DDEJetSection (size_type d, size_type i, ScalarType c=0.)
- DDEJetSection (size_type d, size_type p, size_type n, VectorType const &vec, ScalarType c)
- ScalarType **operator()** (CurveSpec const &curve) const
- VectorType getGradient (CurveSpec curve) const
- **operator VectorType** () const
- size_type **storageDimension** () const
- size_type **dimension** () const
- DDEJetSection & **set_c** (ScalarType c)
- ScalarType **get_c** () const
- DDEJetSection & **set_s** (VectorType s)
- VectorType const & **get_s** () const
- DDEJetSection & **extend** (JetType const &jet)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- iterator **begin** ()
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const

Protected Attributes

- ScalarType **m_c**
- VectorType **m_s**
- JetStorageType **m_jets**
- VectorType **m_orig_s**

7.14.1 Constructor & Destructor Documentation

7.14.1.1 DDEJetSection() [1/2]

```
template<typename CurveSpec , bool isInterval = capd::TypeTraits<typename CurveSpec::Scalar↩
Type>::isInterval>
capd::ddes::DDEJetSection< CurveSpec, isInterval >::DDEJetSection (
            size_type d,
            size_type i,
            ScalarType c = 0.  )  [inline]
```

makes a section checking value at t=0 of a given direction, i.e. coordinate section: x(0)_i == c, where x(0) is d dimensional.

7.14.1.2 DDEJetSection() [2/2]

```
template<typename CurveSpec , bool isInterval = capd::TypeTraits<typename CurveSpec::Scalar↩
Type>::isInterval>
capd::ddes::DDEJetSection< CurveSpec, isInterval >::DDEJetSection (
            size_type d,
            size_type p,
            size_type n,
            VectorType const & vec,
            ScalarType c )  [inline]
```

makes a section with p Jets of order n each and expected scalar value c. vector vec must be d $*$ (1 + p $*$ (n+1)) dimensional and will propagate the jets.

7.14.2 Member Function Documentation

7.14.2.1 getGradient()

```
template<typename CurveSpec , bool isInterval = capd::TypeTraits<typename CurveSpec::Scalar↩
Type>::isInterval>
VectorType capd::ddes::DDEJetSection< CurveSpec, isInterval >::getGradient (
            CurveSpec curve ) const  [inline]
```

This returns gradient of the section d/dx s(x) evaluated at x = j(curve)

NOTE: Since in our case section is a hypersurface, the gradient is constant and equal (z(s), j(s)), extended to the structure of curve, se next NOTE.

NOTE: this function uses structure in curve to deliver higher order jet elements needed for computations. They will be set simply to 0. Therefore, it would be possible to multiply as a Vectors, i.e. we can simply do the vector-vector dot product:

getGradient(curve) . (z(curve), j(curve))

without worrying about (z(s), j(s)) having bad dimensions.

Note: curve passed by copy, as I need it inside

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDEJet↩Section.h

7.15 capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec > Class Template Reference

Public Types

- typedef FunctionalMapSpec **FunctionalMapType**
- typedef FunctionalMapType::CurveType **SolutionCurveType**
- typedef FunctionalMapType::CurveType **CurveType**
- typedef FunctionalMapType::JetType **JetType**
- typedef FunctionalMapType::DataType **DataType**
- typedef FunctionalMapType::VectorType **VectorType**
- typedef FunctionalMapType::MatrixType **MatrixType**
- typedef FunctionalMapType::TimePointType **TimePointType**
- typedef FunctionalMapType::RealType **RealType**
- typedef FunctionalMapType::size_type **size_type**
- typedef FunctionalMapType::VariableStorageType **VariableStorageType**
- typedef FunctionalMapType::JacobianStorageType **JacobianStorageType**
- typedef FunctionalMapType::ValueStorageType **ValueStorageType**

Public Member Functions

- **DDENonrigorousTaylorSolver** (DDENonrigorousTaylorSolver const &solver)
- **DDENonrigorousTaylorSolver** (FunctionalMapType const &map, size_type maxOrder=20)
- void operator() (CurveType &in_out_curve)
- void operator() (CurveType &in_out_curve, JacobianStorageType &D)
- void oneStep (TimePointType const &in_t0, TimePointType const &in_th, RealType const &in_h, ValueStorageType const &in_u, ValueStorageType &out_Phi_coeffs_t0, JacobianStorageType &out_JacPhi_coeffs_t0, ValueStorageType &out_Phi_z, JacobianStorageType &out_JacPhi_z)
- void oneStep (TimePointType const &in_t0, TimePointType const &in_th, RealType const &in_h, ValueStorageType const &in_u, ValueStorageType &out_Phi_coeffs_t0, ValueStorageType &out_Phi_z)
- FunctionalMapType const & **getMap** () const
- FunctionalMapType & **getMap** ()

7.15.1 Member Function Documentation

7.15.1.1 oneStep() [1/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >::oneStep (
            TimePointType const & in_t0,
            TimePointType const & in_th,
            RealType const & in_h,
            ValueStorageType const & in_u,
            ValueStorageType & out_Phi_coeffs_t0,
            JacobianStorageType & out_JacPhi_coeffs_t0,
            ValueStorageType & out_Phi_z,
            JacobianStorageType & out_JacPhi_z )  [inline]
```

this function produces approximation to the solution after time h = distance between grid points in the input curve. The var out_Phi_coeffs_t0 will hold the Jet coefficients at point t0, then out_JacPhi_coeffs_t0 will be the Jacobian of coeffs w.r.t. variables in out_u. Then out_Phi_x[0] is the value at t = t0+h computed by the Taylor method, i.e. evaluation of out_Phi_coeffs_t0(t0+h). out_JacPhi_x[0] is like in Taylor method for ODEs.

It is designed to be used internally. Operators() and epsilonShift() use this.

7.15.1.2 oneStep() [2/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >::oneStep (
            TimePointType const & in_t0,
            TimePointType const & in_th,
            RealType const & in_h,
            ValueStorageType const & in_u,
            ValueStorageType & out_Phi_coeffs_t0,
            ValueStorageType & out_Phi_z )  [inline]
```

same as the other oneStep, but without computing JacPhi

7.15.1.3 operator()() [1/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >::operator() (
            CurveType & in_out_curve )  [inline]
```

a nice operator form of the one step solver

7.15.1.4 operator()() [2/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDENonrigorousTaylorSolver< FunctionalMapSpec >::operator() (
            CurveType & in_out_curve,
            JacobianStorageType & D )  [inline]
```

a nice operator form of the one step solver NOTE IMPORTANT: we assume that DataType of Jet can hold value and Matrix we provide candidate for this: VectorWithJacobianData you cannot use this function with the basic data structure for SolutionCurve this is also motivated with the optimization in computation of rhs (if we do not use all past data, but only e.g. single discrete delay) the VectorWithJacobianData is to bind value of a variable with the derivative of coefficients w.r.t. initial data. Please check examples on how to use this in your code. TODO: (FUTURE): make this more user friendly...

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ NonrigorousTaylorSolver.h

7.16 capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec > Class Template Reference

```
#include <DDEPiecewisePolynomialCurve.h>
```

Classes

- struct rebind

Public Types

- typedef GridSpec **GridType**
- typedef JetSpec **JetType**
- typedef JetType **CurvePieceType**
- typedef DDEPiecewisePolynomialCurve< GridType, JetType > **Class**
- typedef JetType::DataType **DataType**
- typedef JetType::MatrixType **MatrixType**
- typedef JetType::VectorType **VectorType**
- typedef JetType::ScalarType **ScalarType**
- typedef ScalarType **RealType**
- typedef GridType::TimePointType **TimePointType**
- typedef JetType::size_type **size_type**
- typedef std::deque< CurvePieceType * > **PiecesStorageType**
- typedef PiecesStorageType::iterator **iterator**
- typedef PiecesStorageType::const_iterator **const_iterator**
- typedef PiecesStorageType::reverse_iterator **reverse_iterator**
- typedef PiecesStorageType::const_reverse_iterator **const_reverse_iterator**

Public Member Functions

- virtual std::string badge () const
- DDEPiecewisePolynomialCurve (const DDEPiecewisePolynomialCurve &orig)
- Class & operator= (const Class &orig)
- DDEPiecewisePolynomialCurve (const GridType &grid, size_type d=0)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, size_type d=0)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, const DataType &value)
- DDEPiecewisePolynomialCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size_type order, const DataType &value)
- size_type length () const
- Class subcurve (size_type index1, size_type index2) const
- Class subcurve (size_type index1) const
- Class subcurve (TimePointType const &t0, TimePointType const &t1) const
- Class subcurve (TimePointType const &t0) const
- reverse_iterator rbegin ()
- const_reverse_iterator rbegin () const
- reverse_iterator rend ()
- const_reverse_iterator rend () const
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- iterator at (TimePointType const &t)
- const_iterator at (TimePointType const &t) const
- GridType const & grid () const
- size_type dimension () const
- size_type storageDimension () const
- Class & setCurrentTime (TimePointType const &t0)
- TimePointType getCurrentTime () const
- TimePointType currentTime () const
- TimePointType t0 () const
- TimePointType getT0 () const
- Class & setT0 (TimePointType const &t0)
- TimePointType pastTime () const
- VectorType get_x () const
- Class & set_x (VectorType const &x)
- Class & affineTransform (MatrixType const &M, VectorType const &v)
- Class & translate (VectorType const &v)
- Class & add (VectorType const &v)
- Class & add (Class const &set)
- Class & mulThenAdd (ScalarType const &c, Class const &set)
- Class & mul (ScalarType const &c)
- Class & operator*= (ScalarType const &c)
- size_type pointToIndex (TimePointType const &t) const
- CurvePieceType & **getPiece** (TimePointType const &t) const
- CurvePieceType & getPiece (size_type const &index) const
- VectorType **eval** (TimePointType t) const
- VectorType eval (RealType t) const
- void **eval** (TimePointType t, DataType &out) const
- void **eval** (RealType t, DataType &out) const
- Class **dt** (DataType const &valueAtCurrent, size_type n=1) const
- template<typename FunctionalSpec >
  Class **dt** (FunctionalSpec const &f, size_type n=1) const
- Class **increasedOrder** (size_type r=1)

- Class **decreasedOrder** (size_type r=1)
- Class & **addPiece** (CurvePieceType const &newPiece)
- Class & addPiece (CurvePieceType ∗newPiece)
- Class & addPastPiece (const CurvePieceType &newPiece)
- Class & addPastPiece (CurvePieceType ∗newPiece)
- Class & setValueAtCurrent (const DataType &value)
- DataType & getValueAtCurrent ()
- const DataType & getValueAtCurrent () const
- JetType jet (RealType t0, RealType t1) const
- JetType jet (TimePointType t0, TimePointType t1) const
- JetType jet (TimePointType t0) const
- JetType ∗ jetPtr (TimePointType t0) const
- DataType j (size_type i, size_type k) const
- DataType j (TimePointType t0, size_type k) const
- JetType const & j (TimePointType t0) const
- JetType & j (TimePointType t0)
- size_type jetOrderAt (TimePointType t0)
- size_type jetCommonMaximalOrder ()
- const GridType & getGrid () const
- const TimePointType getStep () const
- std::string show () const
- template<typename DynSysSpec >
  Class & extend (DynSysSpec &solver)
- template<typename DynSysSpec >
  void epsilonShift (DynSysSpec const &solver, RealType const &epsilon, Class &out_result) const
- template<typename DynSysSpec >
  void epsilonShift (DynSysSpec const &solver, TimePointType const &at_t0, RealType const &epsilon, Class &out_result) const
- ScalarType dot (VectorType const &v) const
- template<typename JetSection >
  ScalarType dot (JetSection const &v) const
- Class & clear ()
- virtual ∼DDEPiecewisePolynomialCurve ()

Protected Member Functions

- void deallocatePieces ()
- template<typename IteratorSpec >
  void copyPieces (IteratorSpec from, IteratorSpec to)
- void **copyPieces** (iterator from, iterator to)
- void **copyPieces** (const_iterator from, const_iterator to)
- void writeTo (std::ostream &out) const
- void readFrom (std::istream &out) const

Protected Attributes

- const GridType & **m_grid**
- TimePointType **m_t_current**
- VectorType **m_lastEnclosure**
- DataType **m_valueAtCurrent**
- size_type **m_dimension**
- PiecesStorageType **m_pieces**

Friends

- std::ostream & operator<< (std::ostream &out, Class const &curve)
- std::istream & operator>> (std::istream &in, Class const &curve)

7.16.1 Detailed Description

template<typename GridSpec, typename JetSpec>
class capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >

TODO: docs

7.16.2 Constructor & Destructor Documentation

7.16.2.1 DDEPiecewisePolynomialCurve() [1/5]

```
template<typename GridSpec , typename JetSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::DDEPiecewisePolynomialCurve (
            const DDEPiecewisePolynomialCurve< GridSpec, JetSpec > & orig )  [inline]
```

copy constructor, standard thing

7.16.2.2 DDEPiecewisePolynomialCurve() [2/5]

```
template<typename GridSpec , typename JetSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::DDEPiecewisePolynomialCurve (
            const GridType & grid,
            size_type d = 0 )  [inline]
```

makes a Curve that is d dimensional, over given grid, located initially at TimePoint 0

7.16.2.3 DDEPiecewisePolynomialCurve() [3/5]

```
template<typename GridSpec , typename JetSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::DDEPiecewisePolynomialCurve (
            const GridType & grid,
            const TimePointType & t0,
            size_type d = 0 )  [inline]
```

makes a Curve that is d dimensional, over given grid starting at a given TimePoint

7.16.2.4 DDEPiecewisePolynomialCurve() [4/5]

```
template<typename GridSpec , typename JetSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::DDEPiecewisePolynomialCurve (
            const GridType & grid,
            const TimePointType & t0,
            const DataType & value )  [inline]
```

creates a Curve that is a single point of given value at time point t0

7.16.2.5 DDEPiecewisePolynomialCurve() [5/5]

```
template<typename GridSpec , typename JetSpec >
capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::DDEPiecewisePolynomialCurve (
            const GridType & grid,
            const TimePointType & t0,
            const TimePointType & t1,
            size_type order,
            const DataType & value )  [inline]
```

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration).

7.16.2.6 ~DDEPiecewisePolynomialCurve()

```
template<typename GridSpec , typename JetSpec >
virtual capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::~DDEPiecewisePolynomialCurve
( )  [inline], [virtual]
```

standard C++ thing

7.16.3 Member Function Documentation

7.16.3.1 add() [1/2]

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::add (
            Class const & set )  [inline]
```

todo: docs

7.16.3.2 add() [2/2]

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::add (
            VectorType const & v )  [inline]
```

todo: docs

7.16.3.3 addPastPiece() [1/2]

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::addPastPiece (
            const CurvePieceType & newPiece )  [inline]
```

same as addPiece but adds in the past (before existing pieces)

7.16.3.4 addPastPiece() [2/2]

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::addPastPiece (
            CurvePieceType * newPiece )  [inline]
```

adds directly by pointer, faster than by copying from reference. The curve will be resp. for deleting.

7.16.3.5 addPiece()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::addPiece (
            CurvePieceType * newPiece )  [inline]
```

adds directly by pointer, faster than by copying from reference. The curve will be resp. for deleting.

7.16.3.6 affineTransform()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::affineTransform (
            MatrixType const & M,
            VectorType const & v )  [inline]
```

todo: docs

7.16.3.7 at() [1/2]

```
template<typename GridSpec , typename JetSpec >
iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::at (
            TimePointType const & t )  [inline]
```

todo: docs

7.16.3.8 at() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::at (
            TimePointType const & t ) const  [inline]
```

todo: docs

7.16.3.9 badge()

```
template<typename GridSpec , typename JetSpec >
virtual std::string capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::badge ( )
const  [inline], [virtual]
```

for identifying in exceptions and output eventual derived classes

7.16.3.10 begin() [1/2]

```
template<typename GridSpec , typename JetSpec >
iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::begin ( )  [inline]
```

todo: docs

7.16.3.11 begin() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::begin ( ) const
[inline]
```

todo: docs

7.16.3.12 clear()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::clear ( )  [inline]
```

clears the Curve, removing all pieces and setting it to be a point of value 0. at current t0

7.16.3.13 copyPieces()

```
template<typename GridSpec , typename JetSpec >
template<typename IteratorSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::copyPieces (
            IteratorSpec from,
            IteratorSpec to )  [inline], [protected]
```

low level, does not care about changing time points, does not clear m_pieces, etc. just copies from to

7.16.3.14 currentTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::currentTime ( )
const  [inline]
```

todo: docs

7.16.3.15 deallocatePieces()

```
template<typename GridSpec , typename JetSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::deallocatePieces ( )  [inline],
[protected]
```

safely deallocates pointers in m_pieces and clears the container

7.16.3.16 dimension()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::dimension ( ) const
[inline]
```

todo: docs

7.16.3.17 dot() [1/2]

```
template<typename GridSpec , typename JetSpec >
template<typename JetSection >
ScalarType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::dot (
            JetSection const & v ) const  [inline]
```

computes dot taking appropriate jets into account. TODO: only template with a given type of section (based on JetType)?

7.16.3.18 dot() [2/2]

```
template<typename GridSpec , typename JetSpec >
ScalarType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::dot (
            VectorType const & v ) const  [inline]
```

computes dot taking appropriate number of first components from Curve

7.16.3.19 end() [1/2]

```
template<typename GridSpec , typename JetSpec >
iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::end ( )  [inline]
```

todo: docs

7.16.3.20 end() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::end ( ) const
[inline]
```

todo: docs

7.16.3.21 epsilonShift() [1/2]

```
template<typename GridSpec , typename JetSpec >
template<typename DynSysSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::epsilonShift (
            DynSysSpec const & solver,
            RealType const & epsilon,
            Class & out_result ) const  [inline]
```

it computes epsilon shift of in_curve and store it in out_result. out_result must be setup so that all desired Time↩ Points are presented and Jets must be initialized to a desired order. Most probable scenario: out_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in_curve starting at t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out_result at -tau, .., 0, respectively! The t0 is taken from in_curve.t0() in this procedure. For a general t0 see other function.

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance (?)

7.16.3.22 epsilonShift() [2/2]

```
template<typename GridSpec , typename JetSpec >
template<typename DynSysSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::epsilonShift (
            DynSysSpec const & solver,
            TimePointType const & at_t0,
            RealType const & epsilon,
            Class & out_result ) const  [inline]
```

it computes epsilon shift of in_curve and store it in out_result. out_result must be setup so that all desired Time↩ Points are presented and Jets must be initialized to a desired order. Most probable scenario: out_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in_curve starting at at_t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out_result at -tau, .., 0, respectively!

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

7.16.3.23 eval()

```
template<typename GridSpec , typename JetSpec >
VectorType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::eval (
            RealType t ) const  [inline]
```

WARNING: THIS FUNCTION IS NOT RIGOROUS FOR INTERVALS! DO NOT USE IN PROOFS WARNING: C↩ OMPUTE RIGOROUSLY ONLY USING eval(TimePointType t) VERSION

7.16.3.24 extend()

```
template<typename GridSpec , typename JetSpec >
template<typename DynSysSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::extend (
            DynSysSpec & solver )  [inline]
```

Naming convention. To be a pair of extend <-> epsilonShift See move() for more docs.

7.16.3.25 get_x()

```
template<typename GridSpec , typename JetSpec >
VectorType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::get_x ( ) const
[inline]
```

todo: docs

7.16.3.26 getCurrentTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getCurrentTime ( )
const  [inline]
```

todo: docs

7.16.3.27 getGrid()

```
template<typename GridSpec , typename JetSpec >
const GridType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getGrid ( )
const  [inline]
```

returns grid object for this curve

7.16.3.28 getPiece()

```
template<typename GridSpec , typename JetSpec >
CurvePieceType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getPiece (
            size_type const & index ) const  [inline]
```

it assumes that index is already processed (see pointToIndex())

7.16.3.29 getStep()

```
template<typename GridSpec , typename JetSpec >
const TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getStep ( )
const  [inline]
```

returns the grid step for this curve

7.16.3.30 getT0()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getT0 ( ) const
[inline]
```

todo: docs

7.16.3.31 getValueAtCurrent() [1/2]

```
template<typename GridSpec , typename JetSpec >
DataType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getValueAtCurrent ( )
[inline]
```

get the value at current

7.16.3.32 getValueAtCurrent() [2/2]

```
template<typename GridSpec , typename JetSpec >
const DataType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::getValueAt↩
Current ( ) const  [inline]
```

get the value at current (const object)

7.16.3.33 grid()

```
template<typename GridSpec , typename JetSpec >
GridType const& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::grid ( ) const
[inline]
```

todo: docs

7.16.3.34 j() [1/4]

```
template<typename GridSpec , typename JetSpec >
DataType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::j (
            size_type i,
            size_type k ) const  [inline]
```

k-th coeff at grid point i

7.16.3.35 j() [2/4]

```
template<typename GridSpec , typename JetSpec >
JetType& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::j (
            TimePointType t0 )  [inline]
```

jet at grid point t0 (reference)

7.16.3.36 j() [3/4]

```
template<typename GridSpec , typename JetSpec >
JetType const& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::j (
            TimePointType t0 ) const  [inline]
```

jet at grid point t0 (reference)

7.16.3.37 j() [4/4]

```
template<typename GridSpec , typename JetSpec >
DataType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::j (
            TimePointType t0,
            size_type k ) const  [inline]
```

k-th coeff at grid point t0

7.16.3.38 jet() [1/3]

```
template<typename GridSpec , typename JetSpec >
JetType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jet (
            RealType t0,
            RealType t1 ) const  [inline]
```

jet at grid point t0 and valid until t1 (copy, see j() for reference)

7.16.3.39 jet() [2/3]

```
template<typename GridSpec , typename JetSpec >
JetType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jet (
            TimePointType t0 ) const  [inline]
```

jet at grid point t0, and valid for t0 to t0 + grid_step (copy, see j() for reference)

7.16.3.40 jet() [3/3]

```
template<typename GridSpec , typename JetSpec >
JetType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jet (
            TimePointType t0,
            TimePointType t1 ) const  [inline]
```

jet at grid point t0 and valid until t1 (copy, see j() for reference)

7.16.3.41 jetCommonMaximalOrder()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jetCommonMaximalOrder
( )  [inline]
```

returns maximal common jet of all jets in this curve

7.16.3.42 jetOrderAt()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jetOrderAt (
            TimePointType t0 )  [inline]
```

returns order of jet available at point t. This is for optimization purposes (do not copy whole Jet just to know its order)

7.16.3.43 jetPtr()

```
template<typename GridSpec , typename JetSpec >
JetType* capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::jetPtr (
            TimePointType t0 ) const  [inline]
```

use with caution! Mainly for internal use for speed purposes

7.16.3.44 length()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::length ( ) const
[inline]
```

number of pieces

7.16.3.45 mul()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::mul (
            ScalarType const & c )  [inline]
```

todo: docs

7.16.3.46 mulThenAdd()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::mulThenAdd (
            ScalarType const & c,
            Class const & set )  [inline]
```

todo: docs

7.16.3.47 operator*=()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::operator*= (
            ScalarType const & c )  [inline]
```

todo: docs

7.16.3.48 operator=()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::operator= (
            const Class & orig )  [inline]
```

assign operator, standard thing

7.16.3.49 pastTime()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::pastTime ( ) const
[inline]
```

todo: docs

7.16.3.50 pointToIndex()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::pointToIndex (
            TimePointType const & t ) const  [inline]
```

returns index in the array of the Jets for a given TimePoint

7.16.3.51 rbegin() [1/2]

```
template<typename GridSpec , typename JetSpec >
reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rbegin ( )
[inline]
```

todo: docs

7.16.3.52 rbegin() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rbegin (
) const  [inline]
```

todo: docs

7.16.3.53 readFrom()

```
template<typename GridSpec , typename JetSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::readFrom (
            std::istream & out ) const  [protected]
```

internal function to make short impl. of >> operator and the long impl. in .hpp file (see comment in >> operator)

7.16.3.54 rend() [1/2]

```
template<typename GridSpec , typename JetSpec >
reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rend ( )  [inline]
```

todo: docs

7.16.3.55 rend() [2/2]

```
template<typename GridSpec , typename JetSpec >
const_reverse_iterator capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rend ( )
const  [inline]
```

todo: docs

7.16.3.56 set_x()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::set_x (
            VectorType const & x )  [inline]
```

todo: docs

7.16.3.57 setCurrentTime()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setCurrentTime (
            TimePointType const & t0 )  [inline]
```

todo: docs

7.16.3.58 setT0()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setT0 (
            TimePointType const & t0 )  [inline]
```

todo: docs

7.16.3.59 setValueAtCurrent()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::setValueAtCurrent (
            const DataType & value )  [inline]
```

set the value at a current time

7.16.3.60 show()

```
template<typename GridSpec , typename JetSpec >
std::string capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::show
```

show human readable representation

7.16.3.61 storageDimension()

```
template<typename GridSpec , typename JetSpec >
size_type capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::storageDimension ( )
const  [inline]
```

todo: docs

7.16.3.62 subcurve() [1/4]

```
template<typename GridSpec , typename JetSpec >
Class capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::subcurve (
            size_type index1 ) const  [inline]
```

Returns subcurve from idex1 to current time. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

7.16.3.63 subcurve() [2/4]

```
template<typename GridSpec , typename JetSpec >
Class capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::subcurve (
            size_type index1,
            size_type index2 ) const  [inline]
```

returns subcurve from index1 to index2 (in indexaction of internal data. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

7.16.3.64 subcurve() [3/4]

```
template<typename GridSpec , typename JetSpec >
Class capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::subcurve (
            TimePointType const & t0 ) const  [inline]
```

returns subcurve of the segment between t0 and currentTime()

7.16.3.65 subcurve() [4/4]

```
template<typename GridSpec , typename JetSpec >
Class capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::subcurve (
            TimePointType const & t0,
            TimePointType const & t1 ) const  [inline]
```

returns subcurve of the segment between t0 and t1

7.16.3.66 t0()

```
template<typename GridSpec , typename JetSpec >
TimePointType capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::t0 ( ) const
[inline]
```

todo: docs

7.16.3.67 translate()

```
template<typename GridSpec , typename JetSpec >
Class& capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::translate (
            VectorType const & v )  [inline]
```

todo: docs

7.16.3.68 writeTo()

```
template<typename GridSpec , typename JetSpec >
void capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::writeTo (
            std::ostream & out ) const  [protected]
```

internal function to make short impl. of << operator and the long impl. in .hpp file (see comment in << operator)

7.16.4 Friends And Related Function Documentation

7.16.4.1 operator<<

```
template<typename GridSpec , typename JetSpec >
std::ostream& operator<< (
            std::ostream & out,
            Class const & curve )  [friend]
```

friend operator must be inline, or the linker gets confused, so we use writeTo/readFrom

7.16.4.2 operator>>

```
template<typename GridSpec , typename JetSpec >
std::istream& operator>> (
            std::istream & in,
            Class const & curve )  [friend]
```

friend operator must be inline, or the linker gets confused, so we use writeTo/readFrom

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ PiecewisePolynomialCurve.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ PiecewisePolynomialCurve.hpp

7.17 capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec > Class Template Reference

```
#include <DDEPoincareMap.h>
```

Public Types

- typedef DynSysSpec **DynSysType**
- typedef SectionSpec **SectionType**
- typedef DynSysSpec::CurveType **CurveType**
- typedef CurveType::TimePointType **TimePointType**
- typedef CurveType::VectorType **VectorType**
- typedef CurveType::MatrixType **MatrixType**
- typedef CurveType::ScalarType **ScalarType**
- typedef CurveType::RealType **RealType**

Public Member Functions

- **DDEPoincareMap** (DDEPoincareMap const &other)
- **DDEPoincareMap** (DynSysType &dynsys, SectionType &section, CrossingDirection direction=Crossing↩ Direction::Both, int reqSteps=0, int maxSteps=-1, double binsearchEpsilon=DEFAULT_BINSEARCH_EPS↩ ILON)
- void operator() (CurveType &curve, CurveType &on_section, RealType &out_approachTime)
- CurveType operator() (CurveType const &X)
- DDEPoincareMap & **setDirection** (CrossingDirection direction)
- CrossingDirection **getDirection** ()
- DDEPoincareMap & **setMaxSteps** (int maxSteps)
- int **getMaxSteps** ()
- DDEPoincareMap & **setRequiredSteps** (int requiredSteps)
- int **getRequiredSteps** ()
- void integrateUntilSectionCrossing (CurveType &curve, RealType &timeBeforeSection, bool &isExtraStep↩ Needed)
- void findCrossingTime (CurveType const &curve, CurveType &requested, RealType &crossingTime, int test↩ Direction)
- CrossingDirection detectCrossingDirection (CurveType const &curve)
- RealType **getLastEpsilonTime** () const
- RealType **getLastTimeBeforeSection** () const
- RealType **getLastReachTime** () const

Protected Member Functions

- void **checkSteps** ()

Protected Attributes

- DynSysType & **m_dynsys**
- SectionType & **m_section**
- CrossingDirection **m_direction**
- int **m_requiredSteps**
- int **m_maxSteps**
- int **m_steps**
- double **m_binsearchEpsilon**
- RealType **m_lastTimeBeforeSection**
- RealType **m_lastEpsilonTime**

7.17.1 Detailed Description

template<typename DynSysSpec, typename SectionSpec>
class capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >

Implementation of Rigorous Poincare Map for use with DDESolvers.

In theory should work with any compatible DDEDynSys and Section.

7.17.2 Member Function Documentation

7.17.2.1 detectCrossingDirection()

```
template<typename DynSysSpec , typename SectionSpec >
CrossingDirection capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >::detectCrossing↩
Direction (
            CurveType const & curve )  [inline]
```

warning: might be time consuming, it (might) integrate the solution... it assumes curve is on section.

7.17.2.2 findCrossingTime()

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >::findCrossingTime (
            CurveType const & curve,
            CurveType & requested,
            RealType & crossingTime,
            int testDirection )  [inline]
```

testDirection = -1 for before section testDirection = +1 for after section binsearch for now

7.17.2.3 integrateUntilSectionCrossing()

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >::integrateUntilSectionCrossing (
            CurveType & curve,
            RealType & timeBeforeSection,
            bool & isExtraStepNeeded )  [inline]
```

it extends the solution until the crossing with the section. Also, it takes into consideration the required number of steps (should be set-up before) It also moves away of section is initially curve crosses the section.

WARNING: it does not check for sanity of argument as of now, so it might cause runtime-errors if the proof is not well prepared.

7.17.2.4 operator()() [1/2]

```
template<typename DynSysSpec , typename SectionSpec >
void capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >::operator() (
            CurveType & curve,
            CurveType & on_section,
            RealType & out_approachTime )  [inline]
```

in curve there is a curve that reaches just after the section in on_section there is the image on the section, as close to section as possible on_section must be initialized with the desired "length" (time interval) and "order" of the representation.

7.17.2.5 operator()() [2/2]

```
template<typename DynSysSpec , typename SectionSpec >
CurveType capd::ddes::DDEPoincareMap< DynSysSpec, SectionSpec >::operator() (
            CurveType const & X )  [inline]
```

to be compatible with standard notion and CAPD interface, It makes following assumptions:

- X and PX will have the same structure (i.e. over same grid points and jets of the same order).

NOTE: you can retrieve information on reach time and epsilon step, but you cannot acquire the solution on the whole time. If you need solution over the full integration time, then use other operator().

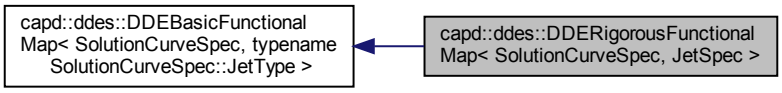
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ PoincareMap.h

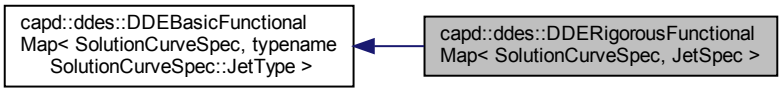
7.18 capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec > Class Template Reference

`#include <FunctionalMap.h>`

Inheritance diagram for capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >:



Collaboration diagram for capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >:



Public Types

- typedef DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec > **BaseClass**
- typedef BaseClass::CurveType **CurveType**
- typedef BaseClass::JetType **JetType**
- typedef BaseClass::RealType **RealType**
- typedef BaseClass::TimePointType **TimePointType**
- typedef BaseClass::MatrixType **MatrixType**
- typedef BaseClass::VectorType **VectorType**
- typedef BaseClass::ScalarType **ScalarType**
- typedef BaseClass::size_type **size_type**
- typedef BaseClass::DataType **DataType**
- typedef BaseClass::VariableStorageType **VariableStorageType**
- typedef BaseClass::ValueStorageType **ValueStorageType**
- typedef BaseClass::JacobianStorageType **JacobianStorageType**

Public Member Functions

- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, ValueStorageType &out_encl, size_type &out_admissible_order) const =0
- virtual ~DDERigorousFunctionalMap ()
- virtual size_type imageDimension () const=0
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, size_type &out_admissible_order) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const=0
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const=0
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const TimePointType &t0, const CurveType &x, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const CurveType &curve, ValueStorageType &coeffs, VariableStorageType &u, JacobianStorageType &Du) const

Static Public Member Functions

- static void convert (VariableStorageType const &u, ValueStorageType &v)

Protected Member Functions

- virtual void checkCurveDimension (CurveType const &x, std::string extra="") const

7.18.1 Detailed Description

template<typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType>
class capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >

TODO: rewrite DOCS this is the interface of a Functional map, that can be evaluated only by "querying" the curve about its jets at various points in the past. it can produce jet at a given time t together with partial derivatives w.r.t. to internal variables from curve used in the computation by recurrent formula for a functional equation:

```
x'(t) = f(t, x)
```

where x is the solution curve in \R^d defined for times smaller than t (up to some maximal past time usually).

A good example of such a functional map is discrete delay case, e.g. F(t, x) = f(t, x(t), x(t-tau)), f : \R \times \R^{2d}->\R (easily extended to more delays). (Probably) the functionals that include integral over the past: F(t, x) = f(t, \int_{t-tau}^{t} x(s) ds) will also fall into this category.

The concrete implementations will be available later.

Implementation note: when implementing concrete class of this interface, do not forget to put using BaseClass↩ ::operator(); using BaseClass::computeDDECoefficients; in the public part of your class to gain access to default implementations of some functions.

TODO: (NOT URGENT, FUTURE, RETHINK) it would be better not to pass containers to computeDDECoefficients, but maybe iterators to given types? It could be used to compute in a given place, i.e. at an already created Jet?

7.18.2 Constructor & Destructor Documentation

7.18.2.1 ~DDERigorousFunctionalMap()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >::~DDERigorousFunctionalMap
( )  [inline], [virtual]
```

virtual destructor for warning suppresion

7.18.3 Member Function Documentation

7.18.3.1 checkCurveDimension()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::checkCurve↩
Dimension  [inline], [protected]
```

helper function to check. TODO: (NOT URGENT, FUTURE) make a switch to turn this off for speed.

7.18.3.2 collectComputationData() [1/2]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::collectComputation↩
Data
```

Collects all data from curve necessary for the computation of the value. This does not collect ENCLOSURE DATA required in rigorous computations.

We use collection of DataType items as the output, we ASSUME that DataType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecesary computations (i.e. Jac Phi would be 0 and we would do $0 * C * r0$). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out_u a finite-dimensional collection of data that is used to compute the map Param: admissible_order will tell how high order of expansion is able to produce with data in u

7.18.3.3 collectComputationData() [2/2]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec >::collect↩
ComputationData (
            const TimePointType & t0,
            const TimePointType & th,
            const RealType & dt,
            const CurveType & x,
            VariableStorageType & out_u,
            ValueStorageType & out_encl,
            size_type & out_admissible_order ) const  [pure virtual]
```

Extra method needed by the rigorous code

Collects all data from curve necessary for the computation of the value. We use collection of DataType items as the output, we ASSUME that SetType elements are used to describe function by CurveType. Therefore, in the output, we could relate how inputs (VariableStorage) corresponds to outputs (ValueStorage) mainly by Jacobian of the map at Variables (JacobianStorageType).

Then, the main function that needs to be implemented in the Map are those depending on the VariableStorageType. Other can have default implementations and are for users convenience.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Note: dt might seem redundant here, but for some reasons it might be helpful, i.e. for epsilon steps (see paper) Note: dt should be by definition dt = th - t0. Note: This function is for optimization purposes. I could assume that the

whole Curve goes into computation, but then the computing cost would grow, as Curve grows in time. Also, if some coefficients are not used in rhs of the equation, we would carry out the unnecesary computations (i.e. Jac Phi would be 0 and we would do $0 * C * r0$). There is one problem with this however: in rigorous code we need to check if the section is not crossed between steps in Poincare map. Please see comments in JetSection / DDEPoincareMap classes.

Param: out_u a finite-dimensional collection of data that is used to compute the map Param: out_encl first entries corresponds to enclosures over [t0, t0+dt] of the variables in u, then, it might contain more entries, relevant to computation. For example higher order enclosures used in Taylor solver. Param: admissible_order will tell how high order of expansion is able to produce with data in u

7.18.3.4 computeDDECoefficients() [1/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.18.3.5 computeDDECoefficients() [2/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients  [inline]
```

this is for a current time in the solution. It provides basic implementation by call to the other function.

7.18.3.6 computeDDECoefficients() [3/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true. For non-developer users of the library it is possible to use versions of the function for specific CurveType.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

7.18.3.7 computeDDECoefficients() [4/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients
```

this is one of two computeDDECoefficients functions that needs to be implemented It takes as an input the proper set of variables representing input data to this map at time t0 (e.g. (appropriate subset of) output of collectComputationData()) and makes use of it to compute recursively the Taylor expansion of the solution (coeffs) at t0. It also computes $\frac{\partial coeffs[i]}{\partial u}$ (in Du). It assumes that u contain appropriate data in u for this specific point t0, the user must assure this is true.

THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. The user of the function should check if this is good size. The param out_admissible_order from collectComputationData() is used for this purpose.

7.18.3.8 computeDDECoefficients() [5/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients [inline]
```

computes recursively the Jet at time t for a given curve for a DDE of the form:

```
x'(t) = F(t, x)
```

Implementation note: the size of the ValueStorageType coeffs container is considered as a desired order. If order is 0 or order is too high for a given curve then the function should alter the order to highest possible. (use coeffs.↩ resize(order) for this purpose)

THERE IS DEFAULT IMPLEMENTATION OF THIS WITH collectComputationData() and computeDDE↩ Coefficients(..., u, ...) call to pure virtual function.

7.18.3.9 computeDDECoefficients() [6/6]

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::computeDDE↩
Coefficients [inline]
```

Same as computeDDECoefficients(const TimePointType&, const CurveType&, JetType&), but also computes the partial derivative:

```
Du[k] = \frac{\partial coeffs[k]}{\partial u}
```

where u are all the variables used to evaluate the map. In u[j] is a d-dimensional set used in computation. In Du[k][j] is a matrix of dimension M(d,d) and of course it is

```
Du[k][j] = \frac{\partial coeffs[k]}{\partial u[j]}
```

It is done this way to allow Solvers to use this structure to reduce wrapping effect of interval arithmetics with the help of Lohner algorithm.

7.18.3.10 convert()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
static void capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::convert  [inline],
[static]
```

helper to convert from collection of sets to collection of vectors

7.18.3.11 imageDimension()

```
template<typename SolutionCurveSpec , typename JetSpec = typename SolutionCurveSpec::JetType>
virtual size_type capd::ddes::DDEBasicFunctionalMap< SolutionCurveSpec, JetSpec >::image↩
Dimension
```

output dimension of the map. THIS FUNCTION NEEDS TO BE IMPLEMENTED IN DERIVED CLASSES.

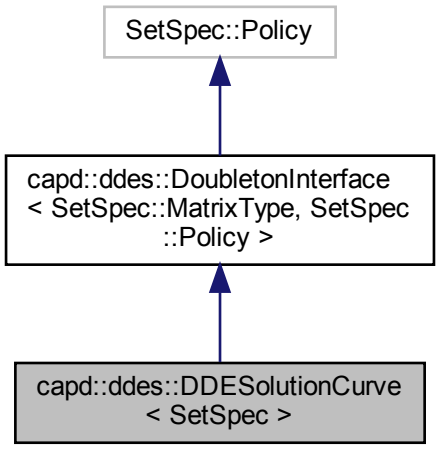
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Functional↩ Map.h

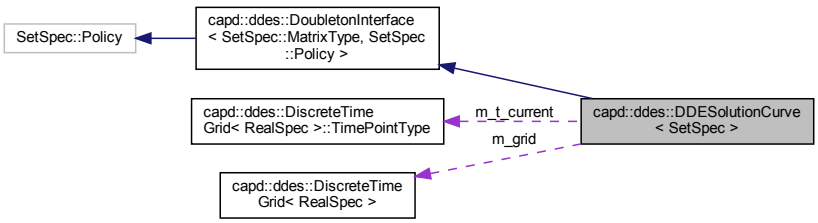
7.19 capd::ddes::DDESolutionCurve< SetSpec > Class Template Reference

```
#include <DDESolutionCurve.h>
```

Inheritance diagram for capd::ddes::DDESolutionCurve< SetSpec >:



Collaboration diagram for capd::ddes::DDESolutionCurve< SetSpec >:



Public Types

- typedef SetSpec **SetType**
- typedef SetType **DataType**
- typedef DoubletonInterface< typename SetSpec::MatrixType, typename SetSpec::Policy > **BaseClass**
- typedef DDESolutionCurve< SetSpec > **Class**
- typedef SetType::MatrixType **MatrixType**
- typedef MatrixType::RowVectorType **VectorType**
- typedef MatrixType::ScalarType **ScalarType**
- typedef ScalarType **RealType**
- typedef DiscreteTimeGrid< RealType > **GridType**
- typedef GridType::TimePointType **TimePointType**
- typedef BaseClass::size_type **size_type**
- typedef DDEForwardTaylorCurvePiece< TimePointType, SetType > **CurvePieceType**
- typedef SetType::Policy **QRPolicy**
- typedef SetType::Policy **Policy**
- typedef CurvePieceType **JetType**
- typedef std::deque< CurvePieceType * > **PiecesStorageType**
- typedef std::deque< bool > **OwnerStorageType**
- typedef PiecesStorageType::iterator **iterator**
- typedef PiecesStorageType::const_iterator **const_iterator**
- typedef PiecesStorageType::reverse_iterator **reverse_iterator**
- typedef PiecesStorageType::const_reverse_iterator **const_reverse_iterator**

Public Member Functions

- DDESolutionCurve (const DDESolutionCurve &orig)
- DDESolutionCurve & operator= (const DDESolutionCurve &orig)
- DDESolutionCurve (const GridType &grid, size_type d=0, size_type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, size_type d=0, size_type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const VectorType &value, size_type N0=0)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size_type order, const SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size_type order, SetType &value)
- DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size_type order, const VectorType &value, size_type N0=0)

- template<typename AnyMatrixSpec >
  DDESolutionCurve (const GridType &grid, const TimePointType &t0, const TimePointType &t1, size_type order, const capd::map::Map< AnyMatrixSpec > &f, size_type N0=0)
- size_type length () const
- DDESolutionCurve subcurve (size_type index1, size_type index2) const
- DDESolutionCurve subcurve (size_type index1) const
- DDESolutionCurve subcurve (TimePointType const &t0, TimePointType const &t1) const
- DDESolutionCurve subcurve (TimePointType const &t0) const
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- iterator **begin** ()
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **at** (TimePointType const &t)
- const_iterator **at** (TimePointType const &t) const
- GridType const & **grid** () const
- size_type **dimension** () const
- TimePointType **t0** () const
- TimePointType **currentTime** () const
- TimePointType **pastTime** () const
- size_type storageDimension () const
- size_type storageN0 () const
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- VectorType get_x () const
- MatrixType get_C () const
- VectorType get_r0 () const
- MatrixType get_B () const
- VectorType get_r () const
- MatrixType get_Binv () const
- BaseClass & set_x (VectorType const &x)
- BaseClass & set_C (MatrixType const &C)
- BaseClass & set_r0 (VectorType const &r0)
- BaseClass & set_Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set_B (MatrixType const &B)
- BaseClass & set_r (VectorType const &r)
- BaseClass & set_Binv (MatrixType const &Binv)
- BaseClass & set_x (VectorType ∗x, bool passOwnership=false)
- BaseClass & set_C (MatrixType ∗C, bool passOwnership=false)
- BaseClass & set_r0 (VectorType ∗r0, bool passOwnership=false)
- BaseClass & set_Cr0 (MatrixType ∗C, VectorType ∗r0, bool passCOwnership=false, bool passr0↩ Ownership=false)
- BaseClass & set_B (MatrixType ∗B, bool passOwnership=false)
- BaseClass & set_r (VectorType ∗r, bool passOwnership=false)
- BaseClass & set_Binv (MatrixType ∗B, bool passOwnership=false)
- VectorType ∗ take_x ()
- MatrixType ∗ take_C ()
- VectorType ∗ take_r0 ()
- MatrixType ∗ take_B ()
- VectorType ∗ take_r ()
- MatrixType ∗ take_Binv ()
- BaseClass & add (VectorType const &v)

< see DoubletonInterface

- BaseClass & add (BaseClass const &set)
- BaseClass & mulThenAdd (ScalarType const &c, BaseClass const &set)
- BaseClass & mul (ScalarType const &c)
- BaseClass & set_Xi (VectorType const &Xi)
- VectorType get_Xi () const
- size_type pointToIndex (TimePointType const &t) const
- CurvePieceType & **getPiece** (TimePointType const &t) const
- CurvePieceType & getPiece (size_type const &index) const
- VectorType **eval** (TimePointType t) const
- VectorType eval (RealType t) const
- void eval (TimePointType t, SetType &out) const
- void **eval** (RealType t, SetType &out) const
- DDESolutionCurve & **addPiece** (CurvePieceType ∗newPiece, bool passOwnership=false)
- DDESolutionCurve & **addPiece** (CurvePieceType const &newPiece)
- DDESolutionCurve & addPastPiece (CurvePieceType ∗newPiece, bool passOwnership=false)
- DDESolutionCurve & addPastPiece (const CurvePieceType &newPiece)
- DDESolutionCurve & setValueAtCurrent (const SetType &value)
- DDESolutionCurve & setValueAtCurrent (const VectorType &value)
- SetType & getValueAtCurrent ()
- const SetType & getValueAtCurrent () const
- JetType **jet** (RealType t0, RealType t1) const
- JetType **jet** (TimePointType t0, TimePointType t1) const
- JetType **jet** (TimePointType t0) const
- JetType ∗ jetPtr (TimePointType t0) const
- SetType **j** (size_type i, size_type k) const
- SetType **j** (TimePointType t0, size_type k) const
- const JetType & **j** (TimePointType t0) const
- JetType & **j** (TimePointType t0)
- size_type jetOrderAt (TimePointType t0)
- const GridType & **getGrid** () const
- const TimePointType **getStep** () const
- Class **midCurve** () const
- Class dt (int k=1) const
- std::string show () const
- template<typename DynSysSpec >
  DDESolutionCurve & move (DynSysSpec &solver)
- template<typename DynSysSpec >
  DDESolutionCurve & extend (DynSysSpec &solver)
- VectorType getLastEnclosure () const
- template<typename DynSysSpec >
  void epsilonShift (DynSysSpec const &solver, RealType const &epsilon, DDESolutionCurve &out_result) const
- template<typename DynSysSpec >
  void epsilonShift (DynSysSpec const &solver, TimePointType const &at_t0, RealType const &epsilon, DDESolutionCurve &out_result) const
- virtual void reinitialize (size_type d, size_type N0)
- ScalarType dot (VectorType const &v) const
- template<typename JetSection >
  ScalarType dot (JetSection const &v) const
- void reduce (DDESolutionCurve &out_result) const
- template<typename SetSpec >
  DDESolutionCurve< SetSpec >::ScalarType **dot** (DDESolutionCurve< SetSpec >::VectorType const &v) const

- template<typename DynSysSpec >
  DDESolutionCurve< SetSpec > & **move** (DynSysSpec &solver)
- virtual VectorType makeStorage_x () const
- virtual MatrixType makeStorage_C () const
  < *see DoubletonInterface*
- virtual VectorType makeStorage_r0 () const
  < *see DoubletonInterface*
- virtual MatrixType makeStorage_B () const
  < *see DoubletonInterface*
- virtual VectorType makeStorage_r () const
  < *see DoubletonInterface*
- virtual VectorType hull () const
  < *see DoubletonInterface*
- virtual VectorType midPoint () const
  < *see DoubletonInterface*

Protected Member Functions

- void **allocate** (size_type N0=0)
- void **deallocatePieces** ()
- void **deallocateR0** ()
- void **deallocate** ()
- void **reallocateR0** (size_type N0)
- void **reallocateR0** ()
- void **reallocate** (size_type N0)
- void **reallocate** ()
- void **updateCommonR0** ()
- template<typename IteratorSpec >
  void copyPieces (IteratorSpec from, IteratorSpec to)
- void **copyPieces** (iterator from, iterator to)
- void **copyPieces** (const_iterator from, const_iterator to)
- void writeTo (std::ostream &out) const

Protected Attributes

- const GridType & **m_grid**
- TimePointType **m_t_current**
- VectorType **m_lastEnclosure**
- VectorType * m_r0
- SetType **m_valueAtCurrent**
- size_type **m_dimension**
- PiecesStorageType **m_pieces**
- OwnerStorageType **m_pieceOwner**

Friends

- std::ostream & operator<< (std::ostream &out, DDESolutionCurve< SetSpec > const &curve)

Additional Inherited Members

7.19.1 Detailed Description

template<typename SetSpec>
class capd::ddes::DDESolutionCurve< SetSpec >

A class to represent the solution $x(t)$ of the DDE. It provides methods To ask for the DDECurvePiece over specific intervals (time intervals) The basic implementation allow to ask about intervals $I_i = [i * h, i* h + h)$, where $h = {\tau}\over{p}$ is the grid size as in FoCM article / PhD thesis. The DDECurvePiece is just a Jet of order n of this solution x at a given time and valid on the specified interval + estimates on the n+1-st derivative. Please consult DDE(ForwardTaylor)CurvePiece for more information how this is organized.

DDESolutionCurve also contains the value of the solution at a maximal time $t_{current}$ - to be used later in the process of solving DDEs. Usually, we will start with some candidate DDESolutionCurve defined over [-\tau, 0], with $t_{current} = 0$. Then we propagate solution by Integrator (see DDESolver) to some t_{future}, so that the solution is estimated over [-\tau, t_{future}]. Please note that the history is preserved for theoretical and technical purposes.

From the theoretical point of view, the history could be used to design solvers for varying delay (as long as it is bounded). From technical point of view, there might be some ways to improve current methods using the history (out of scope as of now).

TODO: (FAR FUTURE) see notes below this line TODOs / DevNOTEs:

Newer note: As of now, class is designed to work with my own implementation of Lohner sets (see 'storage/' subdirectory). They are just Doubletons with some restrictions of classical CAPD sets removed to allow m_C and m_r0 parts to be more freely defined. Ideally I would like to use standard sets from CAPD, but this might be not possible / not optimal (becouse of the part r_0 in definitions, which is ideally shared between sets). Alternatively, I might consider using a big CAPD set and some proxies that can extract subcolumns/subrows of various elements to be passed down to ForwardTaylorCurvePieces or Taylor coefficients of given order.

Older note: The class is templated with a classical CAPD Lohner Set Type, but we will forcefully alter it behaviour (I hope). I will probably use its QR policy etc., but I will alter the matrix m_C and vector m_r - they will be stored from within the set more or less, and might not be related to the space dimension. What is required that the product $m_C * m_r$ has dimension d, i.e. m_C is d rows, $N0$ columns, and m_r0 is of dimension $N0$.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 DDESolutionCurve() [1/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const DDESolutionCurve< SetSpec > & orig )  [inline]
```

copy constructor, standard thing

7.19.2.2 DDESolutionCurve() [2/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            size_type d = 0,
            size_type N0 = 0 )  [inline]
```

makes a Curve that is d dimensional, over given grid, located initially at TimePoint 0, and its r0 vector is N0 dimensional

7.19.2.3 DDESolutionCurve() [3/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            size_type d = 0,
            size_type N0 = 0 )  [inline]
```

makes a Curve that is d dimensional, over given grid starting at a given TimePoint, and its r0 vector is N0 dimensional

7.19.2.4 DDESolutionCurve() [4/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const SetType & value )  [inline]
```

creates a Curve that is a single point of given value at time point t0,m the r0 part of value is used as r0 of the Curve

7.19.2.5 DDESolutionCurve() [5/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const VectorType & value,
            size_type N0 = 0 )  [inline]
```

creates a Curve that is a single point of given value at time point t0 and with given dimension of r0 part.

7.19.2.6 DDESolutionCurve() [6/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const TimePointType & t1,
            size_type order,
            const SetType & value )  [inline]
```

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration). The r0 part will be COPIED from the set and will be common to all elements.

7.19.2.7 DDESolutionCurve() [7/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const TimePointType & t1,
            size_type order,
            SetType & value )  [inline]
```

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration). The r0 part will be transferred (take from, the pointer - of possible / the set implementation allows that) from the set and will be common to all elements.

7.19.2.8 DDESolutionCurve() [8/9]

```
template<typename SetSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const TimePointType & t1,
            size_type order,
            const VectorType & value,
            size_type N0 = 0 )  [inline]
```

creates a constant function with a given value on interval [t1, t0] and with given order (useful for future use as a a set in integration). The r0 part will be N0 dimensional (default = 0, i.e. no C∗r0 set part).

7.19.2.9 DDESolutionCurve() [9/9]

```
template<typename SetSpec >
template<typename AnyMatrixSpec >
capd::ddes::DDESolutionCurve< SetSpec >::DDESolutionCurve (
            const GridType & grid,
            const TimePointType & t0,
            const TimePointType & t1,
            size_type order,
            const capd::map::Map< AnyMatrixSpec > & f,
            size_type N0 = 0 )  [inline]
```

creates a representation of the function f over the interval [t0,t1] The r0 part will be N0 dimensional (default = 0, i.e. no C∗r0 set part).

TODO: rethink: move to a ddeshelper?

7.19.3 Member Function Documentation

7.19.3.1 add() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::add (
            BaseClass const & set )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.2 add() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::add (
            VectorType const & v )  [virtual]
```

< see DoubletonInterface

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.3 addPastPiece() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec > & capd::ddes::DDESolutionCurve< SetSpec >::addPastPiece (
            const CurvePieceType & newPiece )
```

same as addPiece but adds in the past (before existing pieces)

7.19.3.4 addPastPiece() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec > & capd::ddes::DDESolutionCurve< SetSpec >::addPastPiece (
            CurvePieceType * newPiece,
            bool passOwnership = false )
```

same as addPiece but adds in the past (before existing pieces)

7.19.3.5 affineTransform()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::affine↩
Transform (
            MatrixType const & M,
            VectorType const & v )
```

see DoubletonInterface

7.19.3.6 copyPieces()

```
template<typename SetSpec >
template<typename IteratorSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::copyPieces (
            IteratorSpec from,
            IteratorSpec to )  [inline], [protected]
```

low level, does not care about changing time points, does not clear m_pieces, etc. just copies from to

7.19.3.7 dot() [1/2]

```
template<typename SetSpec >
template<typename JetSection >
DDESolutionCurve< SetSpec >::ScalarType capd::ddes::DDESolutionCurve< SetSpec >::dot (
            JetSection const & v ) const
```

computes dot taking appropriate jets into account

7.19.3.8 dot() [2/2]

```
template<typename SetSpec >
ScalarType capd::ddes::DDESolutionCurve< SetSpec >::dot (
            VectorType const & v ) const  [virtual]
```

computes dot taking appropriate number of first components from Curve

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.9 dt()

```
template<typename SetSpec >
Class capd::ddes::DDESolutionCurve< SetSpec >::dt (
            int k = 1 ) const  [inline]
```

compute derivative of the whole curve w.r.t. t of order k

7.19.3.10 epsilonShift() [1/2]

```
template<typename SetSpec >
template<typename DynSysSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::epsilonShift (
            DynSysSpec const & solver,
            RealType const & epsilon,
            DDESolutionCurve< SetSpec > & out_result ) const  [inline]
```

it computes epsilon shift of in_curve and store it in out_result. out_result must be setup so that all desired Time↩ Points are presented and Jets must be initialized to a desired order. Most probable scenario: out_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in_curve starting at t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out_result at -tau, .., 0, respectively! The t0 is taken from in_curve.t0() in this procedure. For a general t0 see other function.

The function does not check if the result is theoretically rigorous, i.e. it does not check if the solution curve is smooth enough to produce rigorous enclosure of a given order. Other classes of the library takes this responsibility (i.e. PoincareMap).

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance (?)

7.19.3.11 epsilonShift() [2/2]

```
template<typename SetSpec >
template<typename DynSysSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::epsilonShift (
            DynSysSpec const & solver,
            TimePointType const & at_t0,
            RealType const & epsilon,
            DDESolutionCurve< SetSpec > & out_result ) const
```

it computes epsilon shift of in_curve and store it in out_result. out_result must be setup so that all desired Time↩ Points are presented and Jets must be initialized to a desired order. Most probable scenario: out_result has Jets of order n at times -tau, -tau+1/h, ..., -h, 0 (current time), where h = 1/p, as in the paper. Then we shift the final part of the in_curve starting at at_t0 - tau - h up to -h, to t0 - tau - h + epsi, ..., -h + epsi. The coefficients will be stored in out_result at -tau, .., 0, respectively!

The function does not check if the result is theoretically rigorous, i.e. it does not check if the solution curve is smooth enough to produce rigorous enclosure of a given order. Other classes of the library takes this responsibility (i.e. PoincareMap).

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

7.19.3.12 eval() [1/2]

```
template<typename SetSpec >
VectorType capd::ddes::DDESolutionCurve< SetSpec >::eval (
            RealType t ) const  [inline]
```

WARNING: THIS FUNCTION IS NOT RIGOROUS FOR INTERVALS! DO NOT USE IN PROOFS WARNING: C↩ OMPUTE RIGOROUSLY ONLY USING eval(TimePointType t) VERSION

7.19.3.13 eval() [2/2]

```
template<typename SetSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::eval (
            TimePointType t,
            SetType & out ) const  [inline]
```

this is rigorous, it returns the value stored in the Jet at a given time point of the grid.

7.19.3.14 extend()

```
template<typename SetSpec >
template<typename DynSysSpec >
DDESolutionCurve& capd::ddes::DDESolutionCurve< SetSpec >::extend (
            DynSysSpec & solver )  [inline]
```

Naming convention. To be a pair of extend <-> epsilonShift See move() for more docs.

7.19.3.15 get_B()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType capd::ddes::DDESolutionCurve< SetSpec >::get_B
```

see DoubletonInterface

7.19.3.16 get_Binv()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType capd::ddes::DDESolutionCurve< SetSpec >::get_Binv
```

see DoubletonInterface

7.19.3.17 get_C()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType capd::ddes::DDESolutionCurve< SetSpec >::get_C
```

see DoubletonInterface

7.19.3.18 get_r()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType capd::ddes::DDESolutionCurve< SetSpec >::get_r
```

see DoubletonInterface

7.19.3.19 get_r0()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType capd::ddes::DDESolutionCurve< SetSpec >::get_r0
```

see DoubletonInterface

7.19.3.20 get_x()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType capd::ddes::DDESolutionCurve< SetSpec >::get_x
```

see DoubletonInterface

7.19.3.21 get_Xi()

```
template<typename SetSpec >
VectorType capd::ddes::DDESolutionCurve< SetSpec >::get_Xi ( ) const  [inline]
```

allows to retrieve Xi for all Jets

7.19.3.22 getLastEnclosure()

```
template<typename SetSpec >
VectorType capd::ddes::DDESolutionCurve< SetSpec >::getLastEnclosure ( ) const  [inline]
```

returns last enclosure of the value of the function over [t0, t0+h] obtained in extend() / move() procedure.

TODO: (NOT URGENT) make a function that returns enclosure of the whole Jet?

7.19.3.23 getPiece()

```
template<typename SetSpec >
CurvePieceType& capd::ddes::DDESolutionCurve< SetSpec >::getPiece (
            size_type const & index ) const  [inline]
```

it assumes that index is already processed (see pointToIndex())

7.19.3.24 getValueAtCurrent() [1/2]

```
template<typename SetSpec >
SetType& capd::ddes::DDESolutionCurve< SetSpec >::getValueAtCurrent ( )  [inline]
```

get the value at current

7.19.3.25 getValueAtCurrent() [2/2]

```
template<typename SetSpec >
const SetType& capd::ddes::DDESolutionCurve< SetSpec >::getValueAtCurrent ( ) const  [inline]
```

get the value at current (const object)

7.19.3.26 jetOrderAt()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::jetOrderAt (
            TimePointType t0 )  [inline]
```

returns order of jet available at point t. This is for optimization purposes (do not copy whole Jet just to know its order)

7.19.3.27 jetPtr()

```
template<typename SetSpec >
JetType* capd::ddes::DDESolutionCurve< SetSpec >::jetPtr (
            TimePointType t0 ) const  [inline]
```

use with caoution! Mainly for internal use for speed purposes

7.19.3.28 length()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::length ( ) const  [inline]
```

number of pieces

7.19.3.29 makeStorage_x()

```
template<typename SetSpec >
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x
[inline]
```

makes a vector that can store x part

7.19.3.30 move()

```
template<typename SetSpec >
template<typename DynSysSpec >
DDESolutionCurve& capd::ddes::DDESolutionCurve< SetSpec >::move (
            DynSysSpec & solver )
```

TODO: (FUTURE) this should be not templated by DynSysSpec but concrete "interfaced" type, known in advance

7.19.3.31 mul()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::mul (
            ScalarType const & c )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.32 mulThenAdd()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::mulThenAdd
(
            ScalarType const & c,
            BaseClass const & set )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.33 operator=()

```
template<typename SetSpec >
DDESolutionCurve& capd::ddes::DDESolutionCurve< SetSpec >::operator= (
            const DDESolutionCurve< SetSpec > & orig )  [inline]
```

assign operator, standard thing

7.19.3.34 pointToIndex()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::pointToIndex (
            TimePointType const & t ) const  [inline]
```

returns index in the array of the Jets for a given TimePoint

7.19.3.35 reduce()

```
template<typename SetSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::reduce (
            DDESolutionCurve< SetSpec > & out_result ) const
```

it reduces the representation so that the maximal order of the jets is the same as in out_result.

It throws exception if the out_result order is higher than this solution at some jet.

7.19.3.36 reinitialize()

```
template<typename SetSpec >
virtual void capd::ddes::DDESolutionCurve< SetSpec >::reinitialize (
            size_type d,
            size_type N0 )  [inline], [virtual]
```

see base class DoubletonInterface

Implements capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.37 set_B() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_B (
            MatrixType * B,
            bool passOwnership = false )
```

see DoubletonInterface

7.19.3.38 set_B() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_B (
            MatrixType const & B )
```

see DoubletonInterface

7.19.3.39 set_Binv() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_Binv (
            MatrixType * B,
            bool passOwnership = false )
```

see DoubletonInterface

7.19.3.40 set_Binv() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_Binv (
            MatrixType const & Binv )
```

see DoubletonInterface

7.19.3.41 set_C() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_C (
            MatrixType * C,
            bool passOwnership = false )
```

see DoubletonInterface

7.19.3.42 set_C() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_C (
            MatrixType const & C )
```

see DoubletonInterface

7.19.3.43 set_Cr0() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_Cr0 (
            MatrixType * C,
            VectorType * r0,
            bool passCOwnership = false,
            bool passr0Ownership = false )
```

see DoubletonInterface

7.19.3.44 set_Cr0() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_Cr0 (
            MatrixType const & C,
            VectorType const & r0 )
```

Set C ∗ r0 part of this set (as a doubleton). Because this set is defined by many sub-sets then we need to assume some order. The order is that, the value at current time is in first d (d = dimension() of the curve) rows of C, the the first jet past the current time is in the following d rows,..., finally last d rows is the Jet at pastTime(). see also DoubletonInterface.

7.19.3.45 set_r() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_r (
            VectorType * r,
            bool passOwnership = false )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.46 set_r() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_r (
            VectorType const & r )  [virtual]
```

see DoubletonInterface

Implements capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.47 set_r0() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_r0 (
            VectorType * r0,
            bool passOwnership = false )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.48 set_r0() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_r0 (
            VectorType const & r0 )  [virtual]
```

see DoubletonInterface

Implements capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.49 set_x() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_x (
            VectorType * x,
            bool passOwnership = false )  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.50 set_x() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::set_x (
            VectorType const & x )  [virtual]
```

see DoubletonInterface

Implements capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.51 set_Xi()

```
template<typename SetSpec >
BaseClass& capd::ddes::DDESolutionCurve< SetSpec >::set_Xi (
            VectorType const & Xi )  [inline]
```

allows to set Xi for all Jets

7.19.3.52 setValueAtCurrent() [1/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec > & capd::ddes::DDESolutionCurve< SetSpec >::setValueAtCurrent (
            const SetType & value )
```

set the value at a current time

7.19.3.53 setValueAtCurrent() [2/2]

```
template<typename SetSpec >
DDESolutionCurve< SetSpec > & capd::ddes::DDESolutionCurve< SetSpec >::setValueAtCurrent (
            const VectorType & value )
```

set the value at a current time

7.19.3.54 show()

```
template<typename SetSpec >
std::string capd::ddes::DDESolutionCurve< SetSpec >::show
```

show detailed info

7.19.3.55 storageDimension()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::storageDimension ( ) const  [inline]
```

DoubletonStorageInterface: returns a dimension of the Jet as a Vector (sequence) to store all the coefficients (without Xi part)

7.19.3.56 storageN0()

```
template<typename SetSpec >
size_type capd::ddes::DDESolutionCurve< SetSpec >::storageN0 ( ) const  [inline]
```

DoubletonStorageInterface: returns a dimension of the r0 vector

7.19.3.57 subcurve() [1/4]

```
template<typename SetSpec >
DDESolutionCurve capd::ddes::DDESolutionCurve< SetSpec >::subcurve (
            size_type index1 ) const  [inline]
```

Returns subcurve from idex1 to current time. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

7.19.3.58 subcurve() [2/4]

```
template<typename SetSpec >
DDESolutionCurve capd::ddes::DDESolutionCurve< SetSpec >::subcurve (
            size_type index1,
            size_type index2 ) const  [inline]
```

returns subcurve from index1 to index2 (in indexaction of internal data. Mainly for internal use. Better use subcurve(TimePoint, TimePoint) version in your programs.

7.19.3.59 subcurve() [3/4]

```
template<typename SetSpec >
DDESolutionCurve capd::ddes::DDESolutionCurve< SetSpec >::subcurve (
            TimePointType const & t0 ) const  [inline]
```

returns subcurve of the segment between t0 and currentTime()

7.19.3.60 subcurve() [4/4]

```
template<typename SetSpec >
DDESolutionCurve capd::ddes::DDESolutionCurve< SetSpec >::subcurve (
            TimePointType const & t0,
            TimePointType const & t1 ) const  [inline]
```

returns subcurve of the segment between t0 and t1

7.19.3.61 take_B()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_↩
B  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.62 take_Binv()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_Binv
[virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.63 take_C()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::MatrixType * capd::ddes::DDESolutionCurve< SetSpec >::take_↩
C  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.64 take_r()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take_↩
r  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.65 take_r0()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take_r0
[virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.66 take_x()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::VectorType * capd::ddes::DDESolutionCurve< SetSpec >::take_↩
x  [virtual]
```

see DoubletonInterface

Reimplemented from capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.67 translate()

```
template<typename SetSpec >
DDESolutionCurve< SetSpec >::BaseClass & capd::ddes::DDESolutionCurve< SetSpec >::translate (
            VectorType const & v )  [virtual]
```

see DoubletonInterface

Implements capd::ddes::DoubletonInterface< SetSpec::MatrixType, SetSpec::Policy >.

7.19.3.68 writeTo()

```
template<typename SetSpec >
void capd::ddes::DDESolutionCurve< SetSpec >::writeTo (
            std::ostream & out ) const  [protected]
```

internal function to make short impl. of << operator and the long impl. in .hpp file (see comment in << operator)

7.19.4 Friends And Related Function Documentation

7.19.4.1 operator<<

```
template<typename SetSpec >
std::ostream& operator<< (
            std::ostream & out,
            DDESolutionCurve< SetSpec > const & curve )  [friend]
```

friend operator must be inline, or the linker gets confused

7.19.5 Member Data Documentation

7.19.5.1 m_r0

```
template<typename SetSpec >
VectorType* capd::ddes::DDESolutionCurve< SetSpec >::m_r0  [protected]
```

this r0 will always be common for all elements. THIS IS IMPORTANT (because how the Lohner algorithm is implemented). Solution is always the owner of r0, all copies must copy it by creating new instance. Each instance must assure that m_r0 is always copied as a pointer (not owned) to the pieces and the current value. This variable must be before other dependent variables, because I use this fact to allocate it before others, so I can sometimes pass it to later dependable objects in the constructor list!

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve_dotImpl.hpp
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve_DoubletonImpl.hpp
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve_IO.hpp
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve_Move.hpp
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ SolutionCurve_PiecesManip.hpp

7.20 capd::ddes::DDETaylorSolver< FunctionalMapSpec > Class Template Reference

Public Types

- typedef FunctionalMapSpec **FunctionalMapType**
- typedef FunctionalMapType::CurveType **SolutionCurveType**
- typedef FunctionalMapType::CurveType **CurveType**
- typedef FunctionalMapType::JetType **JetType**
- typedef SolutionCurveType::SetType **SetType**
- typedef FunctionalMapType::DataType **DataType**
- typedef FunctionalMapType::VectorType **VectorType**
- typedef FunctionalMapType::MatrixType **MatrixType**
- typedef FunctionalMapType::TimePointType **TimePointType**
- typedef FunctionalMapType::RealType **RealType**
- typedef FunctionalMapType::size_type **size_type**
- typedef FunctionalMapType::VariableStorageType **VariableStorageType**
- typedef FunctionalMapType::JacobianStorageType **JacobianStorageType**
- typedef FunctionalMapType::ValueStorageType **ValueStorageType**

Public Member Functions

- **DDETaylorSolver** (DDETaylorSolver const &solver)
- **DDETaylorSolver** (FunctionalMapType const &map, size_type maxOrder)
- void encloseSolution (TimePointType const &in_t0, SolutionCurveType const &in_curve, TimePointType &out_th, RealType &out_HH, VariableStorageType &out_u, ValueStorageType &out_u_encl, ValueStorage↩ Type &out_Phi_coeffs_t0, JacobianStorageType &out_JacPhi_coeffs_t0, ValueStorageType &out_Rem_↩ coeffs_t0, ValueStorageType &out_Y, ValueStorageType &out_Phi_z, JacobianStorageType &out_JacPhi_z, ValueStorageType &out_Rem_z)
- void encloseSolution (SolutionCurveType const &in_curve, TimePointType &out_th, RealType &out_↩ HH, VariableStorageType &out_u, ValueStorageType &out_u_encl, ValueStorageType &out_Phi_coeffs_t0, JacobianStorageType &out_JacPhi_coeffs_t0, ValueStorageType &out_Rem_coeffs_t0, ValueStorage↩ Type &out_CoeffsEnclosure, ValueStorageType &out_Phi_z, JacobianStorageType &out_JacPhi_z, Value↩ StorageType &out_Rem_z)
- const FunctionalMapType & **getMap** () const
- FunctionalMapType & **getMap** ()

7.20.1 Member Function Documentation

7.20.1.1 encloseSolution() [1/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDETaylorSolver< FunctionalMapSpec >::encloseSolution (
             SolutionCurveType const & in_curve,
             TimePointType & out_th,
             RealType & out_HH,
             VariableStorageType & out_u,
             ValueStorageType & out_u_encl,
             ValueStorageType & out_Phi_coeffs_t0,
             JacobianStorageType & out_JacPhi_coeffs_t0,
             ValueStorageType & out_Rem_coeffs_t0,
             ValueStorageType & out_CoeffsEnclosure,
             ValueStorageType & out_Phi_z,
             JacobianStorageType & out_JacPhi_z,
             ValueStorageType & out_Rem_z )  [inline]
```

see docs for encloseSolution(TimePointType const& t0, ...). It does the same but for t0 = in_curve.t0()

7.20.1.2 encloseSolution() [2/2]

```
template<typename FunctionalMapSpec >
void capd::ddes::DDETaylorSolver< FunctionalMapSpec >::encloseSolution (
             TimePointType const & in_t0,
             SolutionCurveType const & in_curve,
             TimePointType & out_th,
             RealType & out_HH,
             VariableStorageType & out_u,
             ValueStorageType & out_u_encl,
             ValueStorageType & out_Phi_coeffs_t0,
             JacobianStorageType & out_JacPhi_coeffs_t0,
             ValueStorageType & out_Rem_coeffs_t0,
             ValueStorageType & out_Y,
             ValueStorageType & out_Phi_z,
             JacobianStorageType & out_JacPhi_z,
             ValueStorageType & out_Rem_z )  [inline]
```

this function produces estimates that could be used to produce solution after time h = distance between grid points in the input curve. The solution will be valid on [t0, t0+h], its enclosure will be as interval set in out_CoeffsEnclosure up to order one higher than the out_Phi_coeffs_t0 The var out_Phi_coeffs_t0 will hold the Jet coefficients at point t0 computed for the middle point of the curve, out_JacPhi_coeffs_t0 will be the Jacobian of coeffs w.r.t. variables in out_u, over the whole set of the curve. The value out_Rem_coeffs_t0 is the remainder, but it will be 0 always in this implementation, as Jets are computed exactly for grid points. Then out_Phi_x[0] is the value at t = t0+h computed by the Taylor method, i.e. evaluation of out_Phi_coeffs_t0(t0+h). out_JacPhi_x[0] and out_Rem_x[0] are like in Taylor method for ODEs.

Curve.move() procedure then should be able to extract information needed to construct a succesive CurvePiece of the solution over [t0, t0+h) and to set current value by use of out_{Phi|JacPhi|Rem}_x as it is customary in ODEs code.

TODO: (FAR FUTURE) rethink that all data in and out encapsulated in some other structure?

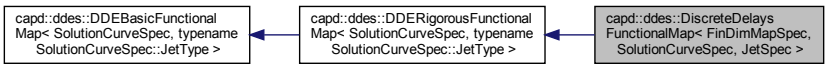
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ TaylorSolver.h

7.21 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > Class Template Reference

`#include <DiscreteDelaysFunctionalMap.h>`

Inheritance diagram for capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩ Spec >:



Collaboration diagram for capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩ Spec >:

Classes

- struct rebind

Public Types

- typedef DDERigorousFunctionalMap< SolutionCurveSpec, JetSpec > **BaseClass**
- typedef DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec > **Class**
- typedef SolutionCurveSpec **CurveType**
- typedef JetSpec **JetType**
- typedef FinDimMapSpec **MapType**
- typedef BaseClass::RealType **RealType**
- typedef BaseClass::TimePointType **TimePointType**
- typedef BaseClass::MatrixType **MatrixType**
- typedef BaseClass::VectorType **VectorType**
- typedef BaseClass::ScalarType **ScalarType**
- typedef BaseClass::size_type **size_type**
- typedef BaseClass::DataType **DataType**
- typedef BaseClass::ValueStorageType **ValueStorageType**
- typedef BaseClass::VariableStorageType **VariableStorageType**
- typedef BaseClass::JacobianStorageType **JacobianStorageType**
- typedef std::vector< TimePointType > DelayStorageType
- typedef DelayStorageType::const_iterator const_iterator
- typedef DelayStorageType::iterator iterator

Public Member Functions

- template<typename OtherDiscreteDelaysFunctionalMap >
  DiscreteDelaysFunctionalMap (const OtherDiscreteDelaysFunctionalMap &orig)
- **DiscreteDelaysFunctionalMap** (const DiscreteDelaysFunctionalMap &orig)
- **DiscreteDelaysFunctionalMap** (const TimePointType &tau)
- **DiscreteDelaysFunctionalMap** (const MapType &f)
- **DiscreteDelaysFunctionalMap** (const MapType &f, const TimePointType &tau)
- **DiscreteDelaysFunctionalMap** (const MapType &f, std::vector< TimePointType > delays)
- **DiscreteDelaysFunctionalMap** (const MapType &f, int n, TimePointType delays[])
- virtual ~DiscreteDelaysFunctionalMap ()
- iterator begin ()
- iterator end ()
- const_iterator begin () const
- const_iterator end () const
- size_type imageDimension () const
- size_type dimension () const
- size_type delaysCount () const
- VectorType operator() (const TimePointType &t, const CurveType &x) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, ValueStorageType &out_encl, size_type &out_admissible_order) const
- virtual void collectComputationData (const TimePointType &t0, const TimePointType &th, const RealType &dt, const CurveType &x, VariableStorageType &out_u, size_type &out_admissible_order) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs) const
- virtual void computeDDECoefficients (const RealType &t0, const ValueStorageType &u, ValueStorageType &coeffs, JacobianStorageType &Du) const

  computeDDECoefficients

Protected Member Functions

- void **checkDimension** (size_type d) const
- template<typename AnyVector >
  void **checkDimension** (AnyVector const &v) const
- void **checkMapSignature** () const
- void findRoughEnclosure (RealType const &t0, RealType const &dt, ValueStorageType const &pastData, VectorType const &z, VectorType &out_Z, VectorType &out_Y) const

Protected Attributes

- MapType **m_map**
- DelayStorageType **m_delays**

Static Protected Attributes

- static const double **ROUGH_MULFACT** = 1.05
- static const double **ROUGH_REFINE_FACTOR** = 0.8
- static const double **ROUGH_TRIALSTEP** = 0.1
- static const size_type **ROUGH_LIMIT** = 32

Additional Inherited Members

7.21.1 Detailed Description

template<typename FinDimMapSpec, typename SolutionCurveSpec, typename JetSpec = typename SolutionCurveSpec::JetType>
class capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >

A class to represent right hand side of the DDE of the form:

```
x' = f(t, x(t), x(t-tau_1), ..., x(t-tau_m))
```

NOTE: This is a demo version of what I want to achieve with the DDEs code interface I assume it should be similar to ODEs in original CAPD. I assume that any RHS map f of the equation x'(t) = f(x_\tau) should be representable with similar interface (i.e. the function accepts some SolutionCurve and can produce Jet at current time in the solution defined with the equation (function computeDDECoefficients) I do not assume anything about SolutionCurve except, it can produce Forward Taylor Jets of itself at a given point (that is used in the equation) and that it can return its value at a current time.

In the implementation below, I assume that I can ask about SOlutionCUrve and its Jet at $t-\tau$, t is current time. Then I construct an AD jet of x(t-\tau) and pass it to standard CAPD map.

NOTE: I do not know if integral equations (i.e. x'(t) = F(\int_{t-\tau}^t g(x(s)) ds) could be described in this manner... But I hope it could be done, at least if g is linear in x (e.g. g = Id). NOTE: in fact, functional map could have operators that are templates, but we assume that the argument curve might have some requirements when computing, for example it could determine only the delays at the specific grid points (because of the loss of regularity, we are not able to allways give jets over some general intervals, see new notes). Therefore, we supply the Map with appropriate CurveType that it can manipulate.

7.21.2 Member Typedef Documentation

7.21.2.1 const_iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef DelayStorageType::const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMap
Spec, SolutionCurveSpec, JetSpec >::const_iterator
```

iterator over this map will iterate over delays!

7.21.2.2 DelayStorageType

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef std::vector< TimePointType > capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec,
SolutionCurveSpec, JetSpec >::DelayStorageType
```

storage type for discrete number of delays

7.21.2.3 iterator

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
typedef DelayStorageType::iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec,
SolutionCurveSpec, JetSpec >::iterator
```

iterator over this map will iterate over delays!

7.21.3 Constructor & Destructor Documentation

7.21.3.1 ~DiscreteDelaysFunctionalMap()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >↩
::~DiscreteDelaysFunctionalMap ( )  [inline], [virtual]
```

standard

7.21.4 Member Function Documentation

7.21.4.1 begin() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::begin ( )  [inline]
```

iterator over delays

7.21.4.2 begin() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::begin ( ) const  [inline]
```

iterator over delays

7.21.4.3 collectComputationData() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::collectComputationData (
            const TimePointType & t0,
            const TimePointType & th,
            const RealType & dt,
            const CurveType & x,
            VariableStorageType & out_u,
            size_type & out_admissible_order ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h] < this is used in computation of rough enclosure, it only need to be the enclosures of value of solution in the past.

7.21.4.4 collectComputationData() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::collectComputationData (
            const TimePointType & t0,
            const TimePointType & th,
            const RealType & dt,
            const CurveType & x,
            VariableStorageType & out_u,
            ValueStorageType & out_encl,
            size_type & out_admissible_order ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs.

TODO: (NOT URGENT) explain what is stored in output for DiscreteDelaysMap...

Warning: dt should be enclosure for the step [0, h] < this is used in computation of rough enclosure, it only need to be the enclosures of value of solution in the past.

7.21.4.5 computeDDECoefficients() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::computeDDECoefficients (
            const RealType & t0,
            const ValueStorageType & u,
            ValueStorageType & coeffs ) const  [inline], [virtual]
```

Implementation of virual func. See Interface docs. for k loop

7.21.4.6 computeDDECoefficients() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
virtual void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, Jet↩
Spec >::computeDDECoefficients (
            const RealType & t0,
            const ValueStorageType & u,
            ValueStorageType & coeffs,
            JacobianStorageType & Du ) const  [inline], [virtual]
```

computeDDECoefficients

see Interface docs.

7.21.4.7 delaysCount()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::delaysCount ( ) const  [inline]
```

number of delays

7.21.4.8 dimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.21.4.9 end() [1/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::end ( )  [inline]
```

iterator over delays

7.21.4.10 end() [2/2]

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
const_iterator capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec,
JetSpec >::end ( ) const  [inline]
```

iterator over delays

7.21.4.11 findRoughEnclosure()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
void capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >↩
::findRoughEnclosure (
            RealType const & t0,
            RealType const & dt,
            ValueStorageType const & pastData,
            VectorType const & z,
            VectorType & out_Z,
            VectorType & out_Y ) const  [inline], [protected]
```

Finds a roughEnclosure Z and Y such that: Y = z + h ∗ f(t0+h, Z, enclosure_of_past_data(x)) \subset \interior Z h = [t0, t0+dt] and z \in Z, where set Z is chosen heuristically. The theorem guarantees that solution to DDE x(t) exists for initial data z (at t0) for any t \in h and x(t) \in Y. The enclosure_of_past_data(x) is the enclosure of the value of the solution in the past times over intervals of length h. See current paper for better explanation.

Note: the code is direct reimplementation of CAPD code for ODEs

7.21.4.12 imageDimension()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
size_type capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::imageDimension ( ) const  [inline]
```

output dimension of the internal map

7.21.4.13 operator()()

```
template<typename FinDimMapSpec , typename SolutionCurveSpec , typename JetSpec = typename
SolutionCurveSpec::JetType>
VectorType capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec
>::operator() (
            const TimePointType & t,
            const CurveType & x ) const  [inline]
```

Implementation of virual func. See Interface docs.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Discrete↩ DelaysFunctionalMap.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Discrete↩ DelaysFunctionalMap.hpp

7.22 capd::ddes::DiscreteTimeGrid< RealSpec > Class Template Reference

```
#include <DDECommon.h>
```

Classes

- class TimePointType

Public Types

- typedef DiscreteTimeGrid **ClassType**

Public Member Functions

- **DiscreteTimeGrid** (DiscreteTimeGrid const &orig)
- DiscreteTimeGrid & **operator=** (DiscreteTimeGrid const &orig)
- **DiscreteTimeGrid** (RealSpec h)
- TimePointType **point** (int i) const
- TimePointType **operator()** (int i) const
- void **split** (RealSpec t, TimePointType &ti, RealSpec &epsi) const

Static Public Member Functions

- static std::string badge ()

Static Public Attributes

- static const RealSpec **zero** = RealSpec(0)

Protected Attributes

- RealSpec **m_h**

7.22.1 Detailed Description

template<typename RealSpec>
class capd::ddes::DiscreteTimeGrid< RealSpec >

represents time points on a grid i ∗ h, i - integer, h - real

TODO: (NOT URGENT, FAR FUTURE, RETHINK) rewrite so that TimePoint holds ref to Grid, then no problems will be presented! TODO: (NOT URGENT, FAR FUTURE, RETHINK) (one problem: when we want to have TimePoint(), i.e. to have 0.0 time point)

7.22.2 Member Function Documentation

7.22.2.1 badge()

```
template<typename RealSpec >
static std::string capd::ddes::DiscreteTimeGrid< RealSpec >::badge ( )  [inline], [static]
```

Badge must be a single word!

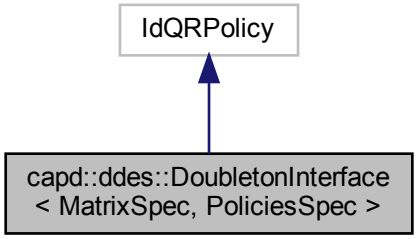
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ Common.h

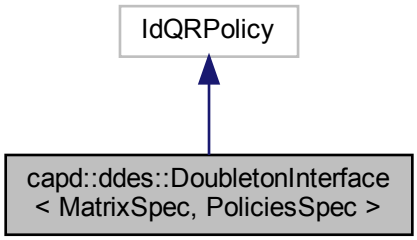
7.23 capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec > Class Template Reference

```
#include <DoubletonInterface.h>
```

Inheritance diagram for capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >:



Collaboration diagram for capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >:



Public Types

- typedef MatrixSpec **MatrixType**
- typedef MatrixType::RowVectorType **VectorType**
- typedef MatrixType::ScalarType **ScalarType**
- typedef MatrixType::size_type **size_type**
- typedef DoubletonInterface **Class**
- typedef DoubletonInterface **BaseClass**
- typedef PoliciesSpec **Policy**
- typedef PoliciesSpec **QRPolicy**

Public Member Functions

- virtual void reinitialize (size_type d, size_type N0)=0
- virtual size_type storageDimension () const =0
- virtual size_type dimension () const
- virtual size_type storageN0 () const =0
- virtual VectorType makeStorage_x () const
- virtual MatrixType makeStorage_C () const
- virtual VectorType makeStorage_r0 () const
- virtual MatrixType makeStorage_B () const
- virtual VectorType makeStorage_r () const
- virtual VectorType midPoint () const
- virtual VectorType hull () const
- virtual operator VectorType () const
- virtual VectorType get_x () const =0
- virtual MatrixType get_C () const =0
- virtual MatrixType get_B () const =0
- virtual MatrixType get_Binv () const
- virtual VectorType get_r () const =0
- virtual VectorType get_r0 () const =0
- virtual Class & set_x (VectorType const &x)=0
- virtual Class & set_C (MatrixType const &C)=0
- virtual Class & set_r0 (VectorType const &r0)=0
- virtual Class & set_B (MatrixType const &B)=0
- virtual Class & set_r (VectorType const &r)=0
- virtual Class & set_Cr0 (MatrixType const &C, VectorType const &r0)=0
- virtual Class & set_x (VectorType ∗x, bool passOwnership=false)
- virtual Class & set_C (MatrixType ∗C, bool passOwnership=false)
- virtual Class & set_r0 (VectorType ∗r0, bool passOwnership=false)
- virtual Class & set_B (MatrixType ∗B, bool passOwnership=false)
- virtual Class & set_r (VectorType ∗r, bool passOwnership=false)
- virtual Class & set_Cr0 (MatrixType ∗C, VectorType ∗r0, bool passOwnership1=false, bool pass↩ Ownership2=false)
- virtual Class & set_Binv (MatrixType ∗Binv, bool passOwnership=false)
- virtual Class & set_Binv (MatrixType const &Binv)
- virtual bool common_x (VectorType const ∗x) const
- virtual bool common_C (MatrixType const ∗C) const
- virtual bool common_r0 (VectorType const ∗r0) const
- virtual bool common_B (MatrixType const ∗B) const
- virtual bool common_r (VectorType const ∗r) const
- virtual ScalarType dot (VectorType const &v) const
- virtual Class & add (VectorType const &v)
- virtual Class & add (Class const &set)

- virtual Class & mul (ScalarType const &c)
- virtual Class & mulThenAdd (ScalarType const &c, Class const &set)
- virtual Class & affineTransform (MatrixType const &M, VectorType const &v)=0
- virtual Class & translate (VectorType const &v)=0
- virtual VectorType * take_x ()
- virtual MatrixType * take_C ()
- virtual VectorType * take_r0 ()
- virtual MatrixType * take_B ()
- virtual VectorType * take_r ()
- virtual MatrixType * take_Binv ()
- virtual ~DoubletonInterface ()
- DoubletonInterface & operator*= (ScalarType const &c)

Static Public Member Functions

- static std::string **badge** ()

Friends

- std::ostream & operator<< (std::ostream &out, Class const &c)
- std::istream & operator>> (std::istream &in, Class &c)

7.23.1 Detailed Description

template<typename MatrixSpec, typename PoliciesSpec = capd::dynset::IdQRPolicy>
class capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >

interface I use in my code to store doubleton data of the form:

X = x + C * r0 + B * r

I use storage* to name methods, as for example dimension() is usually related to the algebraic dimension of the space in CAPD. In my case I will have doubletons that represent data by a huge number of coefficients, but in small dimensional space, i.e. piecewise Taylor curves of order n in d -dimensional space. so the storageDimension will be (n+1) * d;

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

NOTE: THIS IS JUST AN INTERFACE for a doubleton set. We provide two specific versions: Basic* and Shared* - first for testing, second for applications.

TODO: (NOT URGENT) move big implementation into .hpp part. TODO: (FAR FUTURE, RETHINK) make set_*(ptr*) methods return bool if either they truly accepted to hold the ownership and the ptr and it is really shared. Then user would be able to take actions accordingly. Now, it is not allways so obvious.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 ~DoubletonInterface()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::~DoubletonInterface ( )
[inline], [virtual]
```

virtual destructor for strict warning compliance

7.23.3 Member Function Documentation

7.23.3.1 add() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add (
            Class const & set )  [inline], [virtual]
```

add a Set to this representation. It takes the representation into consideration.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >.

7.23.3.2 add() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::add (
            VectorType const & v )  [inline], [virtual]
```

add vector to this representation. It is distributed between x and B∗r part by default

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.3 affineTransform()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::affineTransform (
            MatrixType const & M,
            VectorType const & v )  [pure virtual]
```

applies in a smart way to this set X the affine transform f(y) = M ∗ (X - v). Needs to be implemented.

Implemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesS

7.23.3.4 common_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::common_B (
            MatrixType const * B ) const  [inline], [virtual]
```

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, Policie

7.23.3.5 common_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::common_C (
            MatrixType const * C ) const  [inline], [virtual]
```

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, Policie

7.23.3.6 common_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::common_r (
            VectorType const * r ) const  [inline], [virtual]
```

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, Policie

7.23.3.7 common_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::common_r0 (
            VectorType const * r0 ) const  [inline], [virtual]
```

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, Policie

7.23.3.8 common_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual bool capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::common_x (
            VectorType const * x ) const  [inline], [virtual]
```

tests if the corresponding part of the set is the same entity as the parameter

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, Policie

7.23.3.9 dimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension ( )
const  [inline], [virtual]
```

by default dimension is equal to storage dimension

7.23.3.10 dot()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual ScalarType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dot (
            VectorType const & v ) const  [inline], [virtual]
```

computes v . thisSet, it is virtual to allow optimization. Now uses get_* in computation (might be slow)

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >.

7.23.3.11 get_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_B ( ) const
[pure virtual]
```

returns B value of the doubleton. Needs to be implemented.

7.23.3.12 get_Binv()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_Binv ( )
const  [inline], [virtual]
```

returns inverse of B value of the doubleton. Default implementation by capd inverse.

7.23.3.13 get_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_C ( ) const
[pure virtual]
```

returns B value of the doubleton. Needs to be implemented.

7.23.3.14 get_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_r ( ) const
[pure virtual]
```

returns r value of the doubleton. Needs to be implemented.

7.23.3.15 get_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_r0 ( )
const  [pure virtual]
```

returns r0 value of the doubleton. Needs to be implemented.

7.23.3.16 get_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::get_x ( ) const
[pure virtual]
```

returns x value of the doubleton. Needs to be implemented.

7.23.3.17 hull()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::hull ( ) const
[inline], [virtual]
```

returns mid point of the set, default: returns value obtained using get_∗() methods. You might want to reimplement for better performance(?)

7.23.3.18 makeStorage_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_B (
) const  [inline], [virtual]
```

makes a vector that can store B part

7.23.3.19 makeStorage_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_C (
) const  [inline], [virtual]
```

makes a vector that can store C part

7.23.3.20 makeStorage_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_r (
) const  [inline], [virtual]
```

makes a vector that can store r part

7.23.3.21 makeStorage_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_r0
( ) const  [inline], [virtual]
```

makes a vector that can store r0 part

7.23.3.22 makeStorage_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::makeStorage_x (
) const  [inline], [virtual]
```

makes a vector that can store x part

7.23.3.23 midPoint()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::midPoint ( )
const  [inline], [virtual]
```

returns mid point of the set, default: returns get_X();

7.23.3.24 mul()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::mul (
             ScalarType const & c )  [inline], [virtual]
```

multiply set by a scalar. Should take set structure into consideration.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.25 mulThenAdd()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::mulThenAdd (
            ScalarType const & c,
            Class const & set )  [inline], [virtual]
```

multiply set by a scalar, then add another set. Very simple basic implementation by first mul then add.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >.

7.23.3.26 operator VectorType()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::operator VectorType ( )
const  [inline], [virtual]
```

returns Vector representation of the set, default: returns hull()

7.23.3.27 operator*=()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
DoubletonInterface& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::operator*= (
            ScalarType const & c )  [inline]
```

alternative to mul(); uses mul to gain polymorphism

7.23.3.28 reinitialize()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual void capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::reinitialize (
            size_type d,
            size_type N0 )  [pure virtual]
```

reinitializes the Doubleton to a gibven structure (all should be 0 or Id for B, Binv)

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

7.23.3.29 set_B() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_B (
            MatrixType * B,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.30 set_B() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_B (
            MatrixType const & B )  [pure virtual]
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesS

7.23.3.31 set_Binv() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_Binv (
            MatrixType * Binv,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.32 set_Binv() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_Binv (
            MatrixType const & Binv )  [inline], [virtual]
```

for future implementations. By default it does nothing - Binv is computed always by inverse matrix. NOTE: user of this function is responsible for providing TRUE inverse. Code does not check anything in this regard.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.33 set_C() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_C (
            MatrixType * C,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.34 set_C() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_C (
            MatrixType const & C )  [pure virtual]
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesS

7.23.3.35 set_Cr0() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_Cr0 (
            MatrixType * C,
            VectorType * r0,
            bool passOwnership1 = false,
            bool passOwnership2 = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.36 set_Cr0() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_Cr0 (
            MatrixType const & C,
            VectorType const & r0 )  [pure virtual]
```

This is to allow to safely change the structure of the set. If changing r0 dimension, then C matrix must be updated to match the dimension, and this must be an atomic operation. Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesS

7.23.3.37 set_r() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_r (
            VectorType * r,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.38 set_r() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_r (
            VectorType const & r )  [pure virtual]
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

7.23.3.39 set_r0() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_r0 (
            VectorType * r0,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.40 set_r0() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_r0 (
            VectorType const & r0 )  [pure virtual]
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

7.23.3.41 set_x() [1/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x (
            VectorType * x,
            bool passOwnership = false )  [inline], [virtual]
```

A version of the interface that allows for sharing of the pointers from external source. passOwnership is a flag to indicate if the class should take resposibility of freeing the memory allocation of the precceding pointer. A default implementation provided for convenience uses the corresponding set_∗ method and then frees the memory of the argument if passOwnership = true. Must assure that args are compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.42 set_x() [2/2]

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::set_x (
            VectorType const & x )  [pure virtual]
```

Must assure that arg is compatible with other parts (dimensions!) or throw exception! Should return ∗this for a cascade.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

7.23.3.43 storageDimension()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::storageDimension
( ) const  [pure virtual]
```

returns a dimension of the stored data. Needs to be implemented.

7.23.3.44 storageN0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::storageN0 ( )
const  [pure virtual]
```

returns a dimension of the r0 vector. Needs to be implemented.

7.23.3.45 take_B()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_B ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_∗

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.46 take_Binv()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_Binv ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_∗

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.47 take_C()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual MatrixType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_C ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.48 take_r()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_r ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.49 take_r0()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_r0 ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::DDEForwardTaylorCurvePiece< TimePointSpec, SetSp and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.50 take_x()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual VectorType* capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::take_x ( )
[inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. It provides basic implementation as returning copy of the data by get_*

Reimplemented in capd::ddes::DDESolutionCurve< SetSpec >, and capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >.

7.23.3.51 translate()

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
virtual Class& capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::translate (
            VectorType const & v )  [pure virtual]
```

applies in a smart way to this set X the transform f(y) = X - v. Needs to be implemented.

Implemented in capd::ddes::DDESolutionCurve< SetSpec >, capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >, and capd::ddes::BasicDoubleton< MatrixSpec, PoliciesSpec >.

7.23.4 Friends And Related Function Documentation

7.23.4.1 operator<<

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
std::ostream& operator<< (
            std::ostream & out,
            Class const & c )  [friend]
```

Should be good for any Doubleton that inherits

7.23.4.2 operator>>

```
template<typename MatrixSpec , typename PoliciesSpec = capd::dynset::IdQRPolicy>
std::istream& operator>> (
            std::istream & in,
            Class & c )  [friend]
```

Should be good for any Doubleton that inherits

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Doubleton↩ Interface.h

7.24 ElNino< ScalarSpec, ParamSpec > Class Template Reference

```
#include <setup-common.h>
```

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **ElNino** (ParamType kappa)
- **ElNino** (ParamType alpha, ParamType beta, ParamType gamma, ParamType kappa)
- **ElNino** (ElNino const &orig)
- **ElNino** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- ElNino & **operator=** (ElNino const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **alpha**
- ParamType **beta**
- ParamType **gamma**
- ParamType **kappa**

7.24.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class ElNino< ScalarSpec, ParamSpec >

A model map of what I expect from a Map R^m -> R^n to be good for use with DDE codes.

7.24.2 Member Function Documentation

7.24.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::dimension ( )  [inline], [static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.24.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline], [static]
```

number of parameters to fully configure equation

7.24.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type ElNino< ScalarSpec, ParamSpec >::imageDimension ( )  [inline], [static]
```

output dimension of the internal map

7.24.2.4 operator()()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void ElNino< ScalarSpec, ParamSpec >::operator() (
            const RealSpec & t,
            const InVectorSpec x,
            OutVectorSpec & fx ) const  [inline]
```

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d $*$ (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - \tau)

Note that tau is not explicitely present in the equation. The concrete value of the delay is defined when constructing DiscreteDelaysFunctionalMap from this template function. See docs there.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/programs/examples/demo-elninio/setup-common.h

7.25 capd::ddeshelper::GeneratorRange< IType > Class Template Reference

Public Member Functions

- **GeneratorRange** (IType lo=0, IType up=0, IType dt=1)
- int **i** ()
- IType **v** ()
- IType **lo** ()
- IType **up** ()
- void **next** ()
- bool **isFinished** ()
- void **reset** ()
- bool **isTrivial** ()
- template<typename iterator >
  void **fill** (iterator from, iterator to)

Static Public Member Functions

- template<typename iterator >
  static bool increment (iterator from, iterator to)

Public Attributes

- IType **start**
- IType **end**
- IType **inc**
- int **index**

Friends

- std::ostream & **operator**<< (std::ostream &out, GeneratorRange const &range)

7.25.1 Member Function Documentation

7.25.1.1 increment()

```
template<typename IType >
template<typename iterator >
static bool capd::ddeshelper::GeneratorRange< IType >::increment (
             iterator from,
             iterator to )  [inline], [static]
```

returns true if the incrementation was successful and the next element is available returns false if the incrementation reached last element and reseted all elements (we are back to all indices 0)

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DECubeSet.h

7.26 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > Class Template Reference

```
#include <GenericJet.h>
```

Classes

- struct rebind

Public Types

- typedef TimePointSpec **TimePointType**
- typedef DataSpec **DataType**
- typedef VectorSpec **VectorType**
- typedef MatrixSpec **MatrixType**
- typedef GenericJet< TimePointType, DataType, VectorType, MatrixType > **Class**
- typedef Class **BaseClass**
- typedef MatrixType::ScalarType **ScalarType**
- typedef MatrixType::size_type **size_type**
- typedef ScalarType **RealType**
- typedef Class **JetType**
- typedef DataType const * **const_iterator**
- typedef DataType * **iterator**

Public Member Functions

- virtual GenericJet & **setupCoeffs** (const_iterator it, const_iterator itEnd, size_type overrideOrder=0)
- virtual GenericJet & **setupCoeffs** (std::vector< DataType > const &coeffs, size_type overrideOrder=0)
- GenericJet & operator= (GenericJet const &orig)
- GenericJet (GenericJet const &orig)
- GenericJet (TimePointType t0=TimePointType(), size_type dimension=0, size_type order=0)
- GenericJet (TimePointType t0, size_type order, const DataType &v)
- GenericJet (TimePointType t0, const_iterator it, const_iterator itEnd)
- GenericJet (TimePointType t0, size_type order, const_iterator it, const_iterator itEnd)
- GenericJet (TimePointType t0, std::vector< DataType > coeffs)
- size_type dimension () const
- size_type order () const
- size_type storageDimension () const
- virtual VectorType evalAtDelta (const RealType &delta_t) const
- virtual void evalAtDelta (const RealType &delta_t, DataType &out) const
- virtual VectorType eval (const RealType &t) const
- virtual void eval (const RealType &t, DataType &out) const
- virtual VectorType evalCoeffAtDelta (size_type n, const RealType &delta_t) const
- virtual void evalCoeffAtDelta (size_type n, const RealType &delta_t, DataType &out) const
- virtual VectorType evalCoeff (size_type n, const RealType &t) const
- virtual void evalCoeff (size_type n, const RealType &t, DataType &out) const
- GenericJet dt (size_type n=1)
- GenericJet coeff (size_type n=1)
- virtual std::string show () const
- operator VectorType () const
- virtual VectorType hull () const
- Class & set_x (VectorType const &x)
- VectorType get_x () const
- iterator begin ()
- iterator end ()
- iterator back ()
- iterator at (size_type k)
- const_iterator begin () const
- const_iterator end () const
- const_iterator back () const
- const_iterator at (size_type k) const
- DataType & operator[ ] (size_type k)
- DataType const & operator[ ] (size_type k) const

- Class & setT0 (TimePointType const &t0)
- TimePointType getT0 () const
- TimePointType t0 () const
- DataType * getCoeffs ()
- Class setCoeffs (DataType *coeffs, size_type order)
- Class setOrder (size_type n)
- Class increasedOrder (size_type n=1)
- Class decreasedOrder (size_type n=1)
- virtual ~GenericJet ()

Static Public Member Functions

- static std::string badge ()

Protected Member Functions

- void allocateCoeffs ()
- void deallocateCoeffs ()
- void reallocateCoeffs ()

Protected Attributes

- TimePointType **m_t0**
- size_type **m_dimension**
- size_type **m_order**
- DataType * **m_coeffs**

Friends

- std::ostream & operator<< (std::ostream &out, GenericJet const &jet)
- std::istream & operator>> (std::istream &in, GenericJet &jet)

7.26.1 Detailed Description

template<typename TimePointSpec, typename DataSpec, typename VectorSpec = typename DataSpec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >

A class to represent a generic jet for any data type

We assume that DataType has default constructor and is convertible form/to VectorType, i.e. DataType has:

- a constructor DataType(VectorType) or DataType(const VectorType&)
- a cast operator VectorType()
- a copy operator and copy constructor.
- default constructor

Examples: capd::Vector and Sets types are compatible. ddes/storage/doubleton types are compatible.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 GenericJet() [1/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > const &
orig )
```

copy constructor

7.26.2.2 GenericJet() [2/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            TimePointType t0 = TimePointType(),
            size_type dimension = 0,
            size_type order = 0 )
```

makes a constant function of the vector value 0

7.26.2.3 GenericJet() [3/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            TimePointType t0,
            size_type order,
            const DataType & v )
```

makes a constant function of the vector value v

7.26.2.4 GenericJet() [4/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            TimePointType t0,
            const_iterator it,
            const_iterator itEnd )
```

fills vector with data. Determine order from the size of data.

7.26.2.5 GenericJet() [5/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            TimePointType t0,
            size_type order,
            const_iterator it,
            const_iterator itEnd )
```

fills vector with data, but sets the order to n (fills higher orders with 0 vectors)

7.26.2.6 GenericJet() [6/6]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::GenericJet
(
            TimePointType t0,
            std::vector< DataType > coeffs )
```

fills vector with data. Determine order from the size of data.

7.26.2.7 ~GenericJet()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::~GenericJet ( )  [inline], [virtual]
```

standard thing

7.26.3 Member Function Documentation

7.26.3.1 allocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::allocateCoeffs ( )  [inline], [protected]
```

TODO: docs

7.26.3.2 at() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::at (
            size_type k )  [inline]
```

iterator to the k-th element in coefficients. Standard thing in C++

7.26.3.3 at() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::at (
            size_type k ) const  [inline]
```

iterator to the k-th element in coefficients. Standard thing in C++

7.26.3.4 back() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::back ( )  [inline]
```

iterator to the last element in coefficients. Standard thing in C++

7.26.3.5 back() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::back ( ) const  [inline]
```

iterator to the last element in coefficients. Standard thing in C++

7.26.3.6 badge()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
static std::string capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::badge ( )  [inline], [static]
```

Badge must be a single word!

7.26.3.7 begin() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::begin ( )  [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

7.26.3.8 begin() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::begin ( ) const  [inline]
```

iterator to the 0-th order Taylor coefficient (might by many-dimensions)

7.26.3.9 coeff()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
GenericJet capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::coeff (
            size_type n = 1 )  [inline]
```

makes a Jet that represents n-th coefficient (i.e. jet^{(n)}/n! of the jet as a function of t, by default n=1

7.26.3.10 deallocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::deallocateCoeffs ( )  [inline], [protected]
```

TODO: docs

7.26.3.11 decreasedOrder()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::decreasedOrder (
            size_type n = 1 )  [inline]
```

TODO: docs

7.26.3.12 dimension()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::dimension ( ) const  [inline]
```

dimension of the value space of the Jet (i.e. $J : \R \to \R^{dimension}$)

7.26.3.13 dt()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
GenericJet capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::dt (
            size_type n = 1 )  [inline]
```

makes a Jet that represents n-th derivative of the jet, by default n=1

7.26.3.14 end() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::end ( )  [inline]
```

iterator to the past the order place. Standard thing in C++

7.26.3.15 end() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
const_iterator capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::end ( ) const  [inline]
```

iterator to the past the order place. Standard thing in C++

7.26.3.16 eval() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::eval (
            const RealType & t ) const  [inline], [virtual]
```

evaluates the the jet at t

7.26.3.17 eval() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::eval (
            const RealType & t,
            DataType & out ) const  [inline], [virtual]
```

evaluates the jet at t

7.26.3.18 evalAtDelta() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
VectorSpec capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::evalAtDelta (
            const RealType & delta_t ) const  [virtual]
```

evaluates the the jet at t0 + delta_t

7.26.3.19 evalAtDelta() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::evalAtDelta (
            const RealType & delta_t,
            DataType & out ) const  [virtual]
```

evaluates the jet at t0 + delta_t

7.26.3.20 evalCoeff() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::evalCoeff (
            size_type n,
            const RealType & t ) const  [inline], [virtual]
```

evaluates the n-th time derivative of the jet at t (n-th coeff is 1/n! ∗ jet^{(n)}(t) = jet^{[n]}(t))

7.26.3.21 evalCoeff() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::evalCoeff (
            size_type n,
            const RealType & t,
            DataType & out ) const  [inline], [virtual]
```

evaluates the n-th time derivative of jet at t (n-th coeff is 1/n! ∗ jet^{(n)}(t) = jet^{[n]}(t))

7.26.3.22 evalCoeffAtDelta() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
VectorSpec capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::evalCoeffAtDelta (
            size_type n,
            const RealType & delta_t ) const  [virtual]
```

evaluates the n-th Coeff of jet at t0 + delta_t (n-th coeff is 1/n! ∗ jet^{(n)}(t) = jet^{[n]}(t))

7.26.3.23 evalCoeffAtDelta() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::evalCoeffAtDelta (
            size_type n,
            const RealType & delta_t,
            DataType & out ) const  [virtual]
```

evaluates the n-th time derivative of jet at t0 + delta_t (n-th coeff is 1/n! ∗ jet^{(n)}(t) = jet^{[n]}(t))

7.26.3.24 get_x()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::get_x ( ) const  [inline]
```

gets the coefficients as a single vector of dimension storageDimension()

7.26.3.25 getCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
DataType* capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::getCoeffs ( )  [inline]
```

TODO: docs

7.26.3.26 getT0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
TimePointType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::getT0 ( ) const  [inline]
```

returns time at which this jet is located

7.26.3.27 hull()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
virtual VectorType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec,
isInterval >::hull ( ) const  [inline], [virtual]
```

return vector representation of the jet. Other name (more explicit for VectorType), can be overwritten (virtual).

7.26.3.28 increasedOrder()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::increasedOrder (
            size_type n = 1 )  [inline]
```

TODO: docs

7.26.3.29 operator VectorType()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::operator
VectorType ( ) const  [inline]
```

return VectorSpec representation of the jet. It is vector of dimension storageDimension and d-dim blocks of the natural order (0-th coefficient first).

7.26.3.30 operator=()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > & capd::ddes::GenericJet<
TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::operator= (
            GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > const &
orig )
```

copy operator

7.26.3.31 operator[]() [1/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
DataType& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::operator[] (
            size_type k )  [inline]
```

returns n-th order Taylor coefficient.

7.26.3.32 operator[]() [2/2]

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
DataType const& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::operator[] (
            size_type k ) const  [inline]
```

returns n-th order Taylor coefficient.

7.26.3.33 order()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::order ( ) const  [inline]
```

order of the Taylor part of the Jet

7.26.3.34 reallocateCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
void capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::reallocateCoeffs ( )  [inline], [protected]
```

TODO: docs

7.26.3.35 set_x()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::set_x (
            VectorType const & x )  [inline]
```

set the coeficients from a vector. get/set_x() are compatible.

7.26.3.36 setCoeffs()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::setCoeffs (
            DataType * coeffs,
            size_type order )  [inline]
```

TODO: docs

7.26.3.37 setOrder()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::setOrder (
            size_type n )  [inline]
```

TODO: docs

7.26.3.38 setT0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
Class& capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >↩
::setT0 (
            TimePointType const & t0 )  [inline]
```

set the base time of the jet

7.26.3.39 show()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec , typename Matrix↩
Spec , bool isInterval>
std::string capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::show  [virtual]
```

more verbose output, more human-friendly. Not rigorous-friendly.

7.26.3.40 storageDimension()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
size_type capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval
>::storageDimension ( ) const  [inline]
```

how many data need to store everything

7.26.3.41 t0()

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
TimePointType capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, is↩
Interval >::t0 ( ) const  [inline]
```

naming convention. See getT0()

7.26.4 Friends And Related Function Documentation

7.26.4.1 operator<<

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
std::ostream& operator<< (
            std::ostream & out,
            GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > const &
jet )  [friend]
```

TODO: docs

7.26.4.2 operator>>

```
template<typename TimePointSpec , typename DataSpec , typename VectorSpec = typename Data↩
Spec::VectorType, typename MatrixSpec = typename DataSpec::MatrixType, bool isInterval = capd↩
::TypeTraits<typename MatrixSpec::ScalarType>::isInterval>
std::istream& operator>> (
            std::istream & in,
            GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval > & jet )
[friend]
```

TODO: docs

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Generic↩ Jet.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Generic↩ Jet.hpp

7.27 capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **LasotaWazewska** (ParamType n)
- **LasotaWazewska** (ParamType beta, ParamType gamma, ParamType n)
- **LasotaWazewska** (LasotaWazewska const &orig)
- **LasotaWazewska** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- LasotaWazewska & **operator=** (LasotaWazewska const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **beta**
- ParamType **gamma**
- ParamType **n**

7.27.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >

Lasota-Wazewska equation for chaos as mentioned in scholarpedia article on Mackey-Glass equation.

x'(t) = -gamma * x(t) + beta * x(t-tau)^n * e^(-x(t-tau))

7.27.2 Member Function Documentation

7.27.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::dimension ( )  [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.27.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::getParamsCount ( )
[inline], [static]
```

number of parameters to fully configure equation

7.27.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::imageDimension ( )
[inline], [static]
```

output dimension of the internal map

7.27.2.4 operator()()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void capd::ddes::LasotaWazewska< ScalarSpec, ParamSpec >::operator() (
             const RealSpec & t,
             const InVectorSpec x,
             OutVectorSpec & fx ) const  [inline]
```

Evaluate r.h.s of the equation.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

7.28 capd::ddes::LinearMap< MatrixSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef MatrixSpec **MatrixType**
- typedef MatrixSpec::RowVectorType **VectorType**
- typedef VectorType::ScalarType **ScalarType**
- typedef VectorType::size_type **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**

Public Member Functions

- void **reinitialize** ()
- **LinearMap** (size_type d)
- **LinearMap** (MatrixType const &A, MatrixType const &B, VectorType const &b)
- **LinearMap** (LinearMap const &orig)
- LinearMap & **operator=** (LinearMap const &orig)
- size_type imageDimension () const
- size_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Protected Member Functions

- static std::string **show** ()

Protected Attributes

- MatrixType **A**
- MatrixType **B**
- VectorType **b**
- MatrixType **AB**

7.28.1 Detailed Description

template<typename MatrixSpec, typename ParamSpec = typename MatrixSpec::ScalarType>
class capd::ddes::LinearMap< MatrixSpec, ParamSpec >

Vector linear equation of the form:

```
x'(t) = A * x(t) + B * x(t-tau) + b
```

Use with 'DiscreteDelaysFunctionalMap' to define tau

7.28.2 Member Function Documentation

7.28.2.1 dimension()

```
template<typename MatrixSpec , typename ParamSpec = typename MatrixSpec::ScalarType>
size_type capd::ddes::LinearMap< MatrixSpec, ParamSpec >::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.28.2.2 imageDimension()

```
template<typename MatrixSpec , typename ParamSpec = typename MatrixSpec::ScalarType>
size_type capd::ddes::LinearMap< MatrixSpec, ParamSpec >::imageDimension ( ) const  [inline]
```

output dimension of the internal map

7.28.2.3 operator()()

```
template<typename MatrixSpec , typename ParamSpec = typename MatrixSpec::ScalarType>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void capd::ddes::LinearMap< MatrixSpec, ParamSpec >::operator() (
            const RealSpec & t,
            const InVectorSpec x,
            OutVectorSpec & fx ) const  [inline]
```

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d ∗ (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - \tau)

Note that tau is not explicitely present in the equation. The concrete value of the delay is defined when constructing DiscreteDelaysFunctionalMap from this template function. See docs there.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

7.29 capd::ddes::MackeyGlass< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **MackeyGlass** (ParamType n)
- **MackeyGlass** (ParamType beta, ParamType gamma, ParamType n)
- **MackeyGlass** (MackeyGlass const &orig)
- **MackeyGlass** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- MackeyGlass & **operator=** (MackeyGlass const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **beta**
- ParamType gamma
- ParamType n

7.29.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >

For now, you need to use class similar to those below to define f(x, y, ..) in your equation. Then you define delays in 'DiscreteDelaysFunctionalMap' or some other capd::ddes::FunctionalMap derived classes. Please remember that usually 'DiscreteDelaysFunctionalMap' must specify the Solution template parameter to be able to determine how to handle solution curves.

The setup is much complicated than in the case of ODEs, as the equations must take into account the problem with the grid over which the solution is defined (as of now). See the problem with the continuity of solution at grid points and long enough integration time described in the papers. Because of those problems, the grid, the solution and the equation (DiscreteDelaysFunctionalMap) must be aware to some extent of each other. TODO: (FAR FUTURE) try to reduce dependency only to Grid (i.e. DiscreteDelaysFunctionalMap and Solution depend on the Grid, TODO: (FAR FUTURE) but not on each other). It could be done, if DiscreteDelaysFunctionalMap accepts all solutions that can TODO: (FAR FUTURE) produce concrete Interface of a Jet at Grid::TimePoints)

Example: typedef BasicDoubleton<IMatrix> SetType; typedef DDESolutionCurve<SetType> SolutionType; typedef MackeyGlass<typename SolutionType::ScalarType> F; typedef DiscreteDelaysFunctionalMap<F> RHS; typedef typename SolutionType::GridType GridType; int p = 128; GridType grid(2.0 / p); F f(2.0, 1.0, 8.0); RHS rhs(f, grid(2 * p));

TODO: (FAR FUTURE) make a class that can use integral differential equation to computeDDECoefficients()

TODO: (FAR FUTURE) Unfortunatelly, as of today, capd::map::Map is not good for my use. TODO: (FAR FUTURE) Need to consult Daniel Wilczak on the matter. TODO: (FAR FUTURE) Need to understand the Node and the Graph of execution used by Daniel Wilczak TODO: (FAR FUTURE) to implement my version of computeDDECoefficients with capd::map::Map (seems doable) A model map of what I expect from a Map R^m -> R^n to be good for use with DDE codes.

7.29.2 Member Function Documentation

7.29.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::dimension ( )  [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.29.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline],
[static]
```

number of parameters to fully configure equation

7.29.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::imageDimension ( )  [inline],
[static]
```

output dimension of the internal map

7.29.2.4 operator()()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::operator() (
            const RealSpec & t,
            const InVectorSpec x,
            OutVectorSpec & fx ) const  [inline]
```

We require that solution class has a templated operator of this signature the program will pass in x the m values where m = dimension() in fx there will be reference to already initialized vector of dimension d = imageDimension(). The dimensions are as follows: if d = imageDimension() then usually m = d $*$ (number of delayed terms + 1). +1 is for the current term, which is always present, and always assumed to be stored in x[0]. (so if your equation is not dependent on this, then you should not use it in your formulas). In this example (Mackey-Glass Eq.), we have: d = 1 (scalar), m = 2 (two terms: current value at 0 and delayed term at t = t - \tau)

Note that tau is not explicitely present in the equation. The concrete value of the delay is defined when constructing DiscreteDelaysFunctionalMap from this template function. See docs there.

7.29.3 Member Data Documentation

7.29.3.1 gamma

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::gamma  [protected]
```

classical parameter, default: 2

7.29.3.2 n

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::MackeyGlass< ScalarSpec, ParamSpec >::n  [protected]
```

classical parameter, default: 1

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

7.30 capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec > Class Template Reference

```
#include <DDEHelperNonrigorous.h>
```

Public Types

- typedef EqSpec **Eq**
- typedef Eq::ScalarType **Scalar**
- typedef Eq::RealType **Real**
- typedef Eq::ParamType **ParamType**
- typedef capd::vectalg::Vector< Scalar, 0 > **Vector**
- typedef capd::vectalg::Matrix< Scalar, 0, 0 > **Matrix**
- typedef capd::vectalg::Vector< ParamType, 0 > **ParamsVector**
- typedef capd::ddes::DiscreteTimeGrid< Real > **Grid**
- typedef Grid::TimePointType **TimePoint**
- typedef capd::ddes::GenericJet< TimePoint, Vector, Vector, Matrix > **Jet**
- typedef capd::ddes::GenericJet< TimePoint, capd::ddes::VectorWithJacData< Vector, Matrix >, Vector, Matrix > **JetWithJacobian**
- typedef capd::ddes::DDEPiecewisePolynomialCurve< Grid, Jet > **Solution**
- typedef capd::ddes::DDEPiecewisePolynomialCurve< Grid, JetWithJacobian > **JacSolution**
- typedef Jet **CurvePiece**
- typedef capd::ddes::BasicDiscreteDelaysFunctionalMap< Eq, JacSolution > **JacDDEq**
- typedef capd::ddes::BasicDiscreteDelaysFunctionalMap< Eq, Solution > **DDEq**
- typedef capd::ddes::DDENonrigorousTaylorSolver< JacDDEq > **JacSolver**
- typedef capd::ddes::DDENonrigorousTaylorSolver< DDEq > **Solver**
- typedef JacSolver::VariableStorageType **Variables**
- typedef JacSolver::JacobianStorageType **Jacobians**
- typedef JacSolver::ValueStorageType **Values**
- typedef JacSolver::size_type **size_type**
- typedef int **step_type**
- typedef capd::ddes::DDEJetSection< JacSolution > **JacJetSection**
- typedef JacJetSection::JetType **JacSecJet**
- typedef capd::ddes::DDEBasicPoincareMap< JacSolver, JacJetSection > **JacPoincareMap**
- typedef capd::ddes::DDEJetSection< Solution > **JetSection**
- typedef capd::ddes::DDEBasicPoincareMap< Solver, JetSection > **PoincareMap**
- typedef capd::ddeshelper::CoordinateFrame< Matrix, Vector, Scalar > **CoordinateFrame**

Public Member Functions

- NonrigorousHelper (ParamsVector params, size_type p, size_type n, step_type reqSteps=0, step_type max↩ Steps=0, size_type maxOrder=10)
- NonrigorousHelper (std::string filepath, step_type reqSteps=0, step_type maxSteps=0, size_type max↩ Order=10)
- NonrigorousHelper (std::istream &input, step_type reqSteps=0, step_type maxSteps=0, size_type max↩ Order=10)
- Solver makeSolver ()
- Solution integrate (int iters, const Vector &initial, Vector &result, bool use_extension=false)
- Solution integrate (int iters, const Vector &initial)
- Solution integrate (int iters, const Vector &initial, Vector &result, PathConfig const &outconfig)
- Real **iteratePoincare** (std::ostream &info, int iters, JacJetSection &section, Vector &x, Matrix &V, Matrix &DP)
- template<typename SecionType >
  void makeSection (Vector const &s, Scalar const &c, SecionType &out_section)
- Real refinePeriodic (std::ostream &info, JacJetSection &section, Vector &x, Matrix &V, Matrix &DP)
- capd::poincare::CrossingDirection **detectCorssingDirection** (JacJetSection section, Vector const &x)
- JacSolution poincare (JacJetSection section, Vector const &x, Matrix &initV, double &reachTime, int &steps, Vector &Px, Vector &fPx, Matrix &V, Matrix &DP)
- JacSolution poincare (JacJetSection section, Vector const &x0, double &reachTime, Vector &Px, Vector &fPx, Matrix &V, Matrix &DP)
- Solution poincare (JetSection section, Vector const &x, double &reachTime, Vector &Px)
- Solution **vectorToSolution** (Vector const &x) const
- JacSolution **vectorToJacSolution** (Vector const &x) const
- template<typename SolutionT >
  std::string **drawSolution** (std::string dirpath, std::string filename, SolutionT const &X, double tshift=0.) const
- std::string **drawSolution** (std::string dirpath, std::string filename, Vector const &x, double tshift=0.) const
- template<typename SolutionT >
  void **drawSolution** (PathConfig const &paths, SolutionT const &X, double tshift=0.) const
- template<typename SolutionT >
  void **drawDelayMap** (std::string dirpath, std::string filename, SolutionT const &X) const
- template<typename SolutionT >
  void **drawDelayMap** (PathConfig const &paths, SolutionT const &X) const
- template<typename VectorIteratorSpec >
  void **saveData** (std::string filepath, VectorIteratorSpec start, VectorIteratorSpec end, std::string extra↩ Comment="")
- void **loadSetup** (std::string filepath)
- std::vector< Vector > loadData (std::string filepath)
- void **setRequiredSteps** (step_type reqSteps, bool control_steps=true)
- void **setMaximumSteps** (step_type maxSteps, bool control_steps=true)
- step_type **getRequiredSteps** () const
- step_type **getMaximumSteps** () const
- step_type **getMaximumOrder** () const
- ParamsVector **params** () const
- size_type **M** () const
- size_type **p** () const
- size_type **n** () const
- size_type **d** () const
- Real **h** () const
- Real **getBasicIntervalLength** () const
- void **setCrossingDirection** (capd::poincare::CrossingDirection const &d)
- const Grid & **grid** () const
- NonrigorousHelper & **setExperimentalRenormalizeVariational** (bool v)
- ParamsVector setParams (ParamsVector const &new_params)
- ParamType setParam (size_type index, ParamType const &new_param)
- TimePoint **t** (int i) const

Static Public Attributes

- static const size_type **PARAMS_COUNT** = EqSpec::getParamsCount() + delaysSpec
- static const size_type **DIMENSION** = EqSpec::imageDimension()

7.30.1 Detailed Description

template<typename EqSpec, int delaysSpec = 1>
class capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >

A helper class that made all the fuzzy boilerplate for you, so you do not need to know various things by yourself.

It provides functions like integrate and poincare to compute images of the initial functions under your own equations. You only need to supply the equation and the number of delays in the system

It only server discrete delay differential equations in the form of

x'(t) = f(x(t), x(t-\tau_1), ... , x(t-tau_m))

TODO: (IMPORTANT) Currently it only support one delay!!!!!! TODO: make sure all const qualifiers are put in the right places!

7.30.2 Constructor & Destructor Documentation

7.30.2.1 NonrigorousHelper() [1/3]

```
template<typename EqSpec , int delaysSpec = 1>
capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::NonrigorousHelper (
             ParamsVector params,
             size_type p,
             size_type n,
             step_type reqSteps = 0,
             step_type maxSteps = 0,
             size_type maxOrder = 10 )  [inline]
```

you need to call at least one of the constructors and then use the setup variable

This constructor takes parameters vector:

(q_1, ..., q_k, tau_1, ... tau_m)

where q_i's are the parameters of the equation (r.h.s f : \R^{d∗(m+1)} \to \R^d of the DDE) and tau_i's are the delays, so that the equation is

x'(t) = f(x(t), x(t-tau_1), ...)

p, n - are parameters of the (p,n)-representation of the functions. p is the number of grid points in the basic interval [-tau, 0], tau = max(tau_i) n is the order of the jet at each point. See papers for more details.

7.30.2.2 NonrigorousHelper() [2/3]

```
template<typename EqSpec , int delaysSpec = 1>
capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::NonrigorousHelper (
            std::string filepath,
            step_type reqSteps = 0,
            step_type maxSteps = 0,
            size_type maxOrder = 10 )  [inline]
```

similar to the first one, but reads the parameters and p and n from a given file.

7.30.2.3 NonrigorousHelper() [3/3]

```
template<typename EqSpec , int delaysSpec = 1>
capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::NonrigorousHelper (
            std::istream & input,
            step_type reqSteps = 0,
            step_type maxSteps = 0,
            size_type maxOrder = 10 )  [inline]
```

similar to the second one, but user gives istream instead of filepath

7.30.3 Member Function Documentation

7.30.3.1 integrate() [1/3]

```
template<typename EqSpec , int delaysSpec = 1>
Solution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::integrate (
            int iters,
            const Vector & initial )  [inline]
```

just integrate the solution and return the solution

initial can be either of (1) d-dimension (\R^d) or (2) the M(d,p,n)-dimension (see papers) In the (1) case it integrates initial function x_0(s) == initial for all s \in [-tau, 0] In the (2) case it constructs initial function from data stored in x0 (the order of coefficients in this vector is described elsewhere (see papers for example). But suppling it by hand is extremely cumbersome. It is best to use output of other functions o get initials in this form.

7.30.3.2 integrate() [2/3]

```
template<typename EqSpec , int delaysSpec = 1>
Solution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::integrate (
            int iters,
            const Vector & initial,
            Vector & result,
            bool use_extension = false )  [inline]
```

just integrate the solution and return the solution

initial can be either of (1) d-dimension (\R^d) or (2) the M(d,p,n)-dimension (see papers) In the (1) case it integrates initial function x_0(s) == initial for all s \in [-tau, 0] In the (2) case it constructs initial function from data stored in x0 (the order of coefficients in this vector is described elsewhere (see papers for example). But suppling it by hand is extremally cumbersome. It is best to use output of other functions o get initials in this form.

Integrates for time T = iters * tau_max / p = iters * h. The value h is the step size of the method / grid

In the result the last segment x_T from the solution in the vector format. Can be used as an input to the next integrate() procedure, to create an initial solution curve, or in other functions from the helper (e.g. poincare).

TODO: use_extension was used for backward compatibility of some programs it should work with use_↩ extension=true in the default version and user should control it with setting m_maxOrder with getter/setter

7.30.3.3 integrate() [3/3]

```
template<typename EqSpec , int delaysSpec = 1>
Solution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::integrate (
            int iters,
            const Vector & initial,
            Vector & result,
            PathConfig const & outconfig )  [inline]
```

integrate, save output to files and draw the solution. See other versions for more information on input and output.

The variable outconfig holds a pair of strings that define paths to where store the results of the computations. See documentation there.

7.30.3.4 loadData()

```
template<typename EqSpec , int delaysSpec = 1>
std::vector<Vector> capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::loadData (
            std::string filepath )  [inline]
```

load data only if compatible with setup

7.30.3.5 makeSection()

```
template<typename EqSpec , int delaysSpec = 1>
template<typename SecionType >
void capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::makeSection (
            Vector const & s,
            Scalar const & c,
            SecionType & out_section )  [inline]
```

makes a section and stores it in out_section

if dim(s) == d, then we setup section in the coordinate x(0) . s = c otherwise we set full-space section s . x_0 = c

7.30.3.6 makeSolver()

```
template<typename EqSpec , int delaysSpec = 1>
Solver capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::makeSolver ( )  [inline]
```

This creates a raw solver, if you want to do nonstandard tasks If you want just to integrate initial values, consider using iterate() or poincare() instead. TODO: add functions to create more elements

7.30.3.7 poincare() [1/3]

```
template<typename EqSpec , int delaysSpec = 1>
JacSolution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::poincare (
            JacJetSection section,
            Vector const & x,
            Matrix & initV,
            double & reachTime,
            int & steps,
            Vector & Px,
            Vector & fPx,
            Matrix & V,
            Matrix & DP )  [inline]
```

if you do not know what initV is, then you should probably stick to using the other JacSolution poincare() method (without initV).

7.30.3.8 poincare() [2/3]

```
template<typename EqSpec , int delaysSpec = 1>
JacSolution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::poincare (
            JacJetSection section,
            Vector const & x0,
            double & reachTime,
            Vector & Px,
            Vector & fPx,
            Matrix & V,
            Matrix & DP )  [inline]
```

compuytes poincare map and the (approximate) solution to the variational equation on the coefficients in V it returns the monodromy matrix, that is D_x \varph(t_P(x0), x) in DP you get D \varphi(t_p(x), x) (sic! the difference in t_p argument) in fPx you get the value of the "vector field" at P(x0) this is not straightforward as in ODE that is for ODE you have fPx = f(P(x0)), where f is r.h.s. of the ODE. But for DDE you do not have r.h.s for all of the points in [-tau, 0] But it can be shown, that if $t_p \ge tau$ then $fPx = (Px)'$ (note Px : [-tau,0] \to \R^d is a function of time, so it can be differentiated)

7.30.3.9 poincare() [3/3]

```
template<typename EqSpec , int delaysSpec = 1>
Solution capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::poincare (
            JetSection section,
            Vector const & x,
            double & reachTime,
            Vector & Px )  [inline]
```

the simples computation of poincare map, without extra data

7.30.3.10 refinePeriodic()

```
template<typename EqSpec , int delaysSpec = 1>
Real capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::refinePeriodic (
             std::ostream & info,
             JacJetSection & section,
             Vector & x,
             Matrix & V,
             Matrix & DP )  [inline]
```

helper function to refine a candidate periodic orbit with a Newton method.

First parameter is to get some text info back during the process, you can pass std::cout there.

Section you need to set-up, it will be s ∗ x_0 = x, s \in \R^{M(d,p,n), c \in \R}

7.30.3.11 setParam()

```
template<typename EqSpec , int delaysSpec = 1>
ParamType capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::setParam (
             size_type index,
             ParamType const & new_param )  [inline]
```

returns old param value; TODO: add index checking

7.30.3.12 setParams()

```
template<typename EqSpec , int delaysSpec = 1>
ParamsVector capd::ddeshelper::NonrigorousHelper< EqSpec, delaysSpec >::setParams (
             ParamsVector const & new_params )  [inline]
```

returns old params

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperNonrigorous.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperNonrigorous.hpp

7.31 capd::ddes::ODEPendulum Class Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef long **size_type**

Public Member Functions

- **ODEPendulum** (ODEPendulum const &orig)
- ODEPendulum & **operator=** (ODEPendulum const &orig)
- size_type imageDimension () const
- size_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static std::string **show** ()

7.31.1 Detailed Description

Mainly for testing how DDE code compares to ODE in CAPD

7.31.2 Member Function Documentation

7.31.2.1 dimension()

```
size_type capd::ddes::ODEPendulum::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.31.2.2 imageDimension()

```
size_type capd::ddes::ODEPendulum::imageDimension ( ) const  [inline]
```

output dimension of the internal map

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩
  Eqns.h

7.32 capd::ddeshelper::PathConfig Class Reference

```
#include <DDEHelperCommon.h>
```

Public Member Functions

- PathConfig (std::string dirPath=".", std::string prefix="")
- PathConfig suffix (std::string suffix) const
- std::string fullpath () const
- std::string filepath (std::string name) const
- std::string filename (std::string name) const
- void mkdir_p () const
- std::string **dirpath** () const

Public Attributes

- std::string **dirPath**
- std::string **prefix**

7.32.1 Detailed Description

used in some helpers to define where the output goes. It does not check the correctness of those paths! You need to assure they are correct!

dirPath - a base directory to where the output save prefix - as name says - prefix of all generated files.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 PathConfig()

```
capd::ddeshelper::PathConfig::PathConfig (
            std::string dirPath = ".",
            std::string prefix = "" )  [inline]
```

make a config path, default is ./ with no prefix

7.32.3 Member Function Documentation

7.32.3.1 filename()

```
std::string capd::ddeshelper::PathConfig::filename (
            std::string name ) const  [inline]
```

makes just a prefixed filename (no path attached)

7.32.3.2 filepath()

```
std::string capd::ddeshelper::PathConfig::filepath (
            std::string name ) const  [inline]
```

makes a new prefixed filepath for a given filename

7.32.3.3 fullpath()

```
std::string capd::ddeshelper::PathConfig::fullpath ( ) const  [inline]
```

dirpath with prefix

7.32.3.4 mkdir_p()

```
void capd::ddeshelper::PathConfig::mkdir_p ( ) const  [inline]
```

assure there is really this folder - not very safe!!!

7.32.3.5 suffix()

```
PathConfig capd::ddeshelper::PathConfig::suffix (
            std::string suffix ) const  [inline]
```

make a new PathConfix with a longer prefix : "prefix-suffix"

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/DDEHelperCommon.h

7.33 capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **PerezMaltaCoutinho** (ParamType b1)
- **PerezMaltaCoutinho** (ParamType gamma, ParamType b0, ParamType b1)
- **PerezMaltaCoutinho** (PerezMaltaCoutinho const &orig)
- **PerezMaltaCoutinho** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- PerezMaltaCoutinho & **operator=** (PerezMaltaCoutinho const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **gamma**
- ParamType **b0**
- ParamType **b1**

7.33.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >

Perez-Malta-Coutinho equation for chaos as mentioned in scholarpedia article on Mackey-Glass equation.

x'(t) = -gamma ∗ x(t) + (b0 - b1 ∗ x(t-tau)) ∗ x(t-tau)

7.33.2 Member Function Documentation

7.33.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::dimension ( )  [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.33.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::getParamsCount ( )
[inline], [static]
```

number of parameters to fully configure equation

7.33.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::imageDimension ( )
[inline], [static]
```

output dimension of the internal map

7.33.2.4 operator()()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void capd::ddes::PerezMaltaCoutinho< ScalarSpec, ParamSpec >::operator() (
            const RealSpec & t,
            const InVectorSpec x,
            OutVectorSpec & fx ) const  [inline]
```

Evaluate r.h.s of the equation.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/SampleEqns.h

7.34 capd::ddes::DiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind< OtherSolutionSpec, OtherJetSpec > Struct Template Reference

Public Types

- typedef DiscreteDelaysFunctionalMap< FinDimMapSpec, OtherSolutionSpec, OtherJetSpec > **other**

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DiscreteDelaysFunctionalMap.h

7.35 capd::ddes::BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, SolutionCurveSpec, JetSpec >::rebind< OtherSolutionSpec, OtherJetSpec > Struct Template Reference

Public Types

- typedef BasicDiscreteDelaysFunctionalMap< FinDimMapSpec, OtherSolutionSpec, OtherJetSpec > **other**

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/BasicDiscreteDelaysFunctionalMap.h

7.36 capd::ddes::GenericJet< TimePointSpec, DataSpec, VectorSpec, MatrixSpec, isInterval >::rebind< OtherTimePointSpec, OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIsInterval > Struct Template Reference

Public Types

- typedef GenericJet< OtherTimePointSpec, OtherDataSpec, OtherVectorSpec, OtherMatrixSpec, OtherIsInterval > **other**

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/GenericJet.h

7.37 capd::ddes::DDEPiecewisePolynomialCurve< GridSpec, JetSpec >::rebind< OtherJetTypeSpec > Struct Template Reference

Public Types

- typedef DDEPiecewisePolynomialCurve< GridSpec, OtherJetTypeSpec > **other**

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDEPiecewisePolynomialCurve.h

7.38 capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec > Class Template Reference

`#include <DDEHelperRigorous.h>`

Classes

- class CoordinateSystem

Public Types

- typedef EqSpec **Eq**
- typedef Eq::ScalarType **Scalar**
- typedef Eq::RealType **Real**
- typedef Eq::ParamType **ParamType**
- typedef capd::vectalg::Vector< Scalar, 0 > **Vector**
- typedef capd::vectalg::Matrix< Scalar, 0, 0 > **Matrix**
- typedef capd::vectalg::Vector< ParamType, 0 > **ParamsVector**
- typedef PoliciesSpec **Policies**
- typedef capd::ddes::SharedDoubleton< Matrix, Policies > **SetType**
- typedef capd::ddes::DDESolutionCurve< SetType > **Solution**
- typedef Solution::GridType **Grid**
- typedef Solution::TimePointType **TimePoint**
- typedef Solution::CurvePieceType **CurvePiece**
- typedef capd::ddes::DiscreteDelaysFunctionalMap< Eq, Solution > **DDEq**
- typedef capd::ddes::DDETaylorSolver< DDEq > **Solver**
- typedef Solver::VariableStorageType **Variables**
- typedef Solver::JacobianStorageType **Jacobians**
- typedef Solver::ValueStorageType **Values**
- typedef Solver::size_type **size_type**
- typedef capd::ddes::DDEJetSection< Solution > **Section**
- typedef Section::JetType **SectionJet**
- typedef capd::ddes::DDEPoincareMap< Solver, Section > **PoincareMap**
- typedef DDECompareHelper< Vector > **Comparator**

Public Member Functions

- **RigorousHelper** (std::string filepath, int reqSteps=0, int maxSteps=0, int maxOrder=10)
- **RigorousHelper** (int p, int n, ParamsVector const &params, int reqSteps=0, int maxSteps=0, int max↩ Order=10)
- Solution dataToSolution (CoordinateSystem const &coords, Vector dx, Vector r0, Vector Xi) const
- Solution dataToSolution (Vector dx, Vector r0, Vector Xi) const
- Solution constantInitialSolution (Vector const &vx) const
- Solution constantInitialSolution (Vector const &vx, size_type order) const
- template<typename AnyMatrixSpec >
  Solution functionToSolution (capd::map::Map< AnyMatrixSpec > f) const
- template<typename AnyMatrixSpec >
  Solution functionToSolution (capd::map::Map< AnyMatrixSpec > f, int starti) const
- Solution r0ToSolution (CoordinateSystem const &coords, Vector r0, Vector Xi) const
- Solution r0ToSolution (Vector r0, Vector Xi) const
- capd::poincare::CrossingDirection **detectCrossingDirection** ()
- Solver **makeSolver** ()
- Solution timemap (Solution &X, const size_type &steps, const Real &epsilon=0.)
- Solution **timemap** (CoordinateSystem const &in_coords, Vector const &dx, Vector const &r0, Vector const &Xi, const size_type &steps, const Real &epsilon, CoordinateSystem const &out_coords, Vector &Tdx, Vector &Tr0, Vector &TXi)
- Solution **timemap** (Vector const &dx, Vector const &r0, Vector const &Xi, const size_type &steps, const Real &epsilon, Vector &Tdx, Vector &Tr0, Vector &TXi)

- Solution poincare (Section section, capd::poincare::CrossingDirection crossing_direction, Solution const &X, Real &reachTime, size_type &steps, Real &epsilon, Solution &PX)
- Solution **poincare** (Section section, capd::poincare::CrossingDirection crossing_direction, Solution const &X, Real &reachTime, Solution &PX)
- Solution **poincare** (CoordinateSystem const &in_coords, Vector const &dx, Vector const &r0, Vector const &Xi, CoordinateSystem const &out_coords, Real &reachTime, size_type &steps, Real &epsilon, Vector &Pdx, Vector &Pr0, Vector &PXi)
- void **toCoords** (Solution const &PX, CoordinateSystem const &coords, Vector &Pdx, Vector &Pr0, Vector &PXi)
- Solution poincare (CoordinateSystem const &in_coords, Vector const &dx, Vector const &r0, Vector const &Xi, Real &reachTime, Vector &Pdx, Vector &Pr0, Vector &PXi)
- Solution poincare (Vector const &dx, Vector const &r0, Vector const &Xi, Real &reachTime, Vector &Pdx, Vector &Pr0, Vector &PXi)
- void **loadSetup** (std::string filepath)
- void **setRequiredSteps** (int reqSteps, bool ensure_long_enough=true)
- int **getRequiredSteps** () const
- void **setMaximumSteps** (int maxSteps, bool ensure_long_enough=true)
- int **getMaximumSteps** () const
- void **setMaximumOrder** (int maxOrder)
- int **getMaximumOrder** () const
- ParamsVector **params** () const
- int **M** () const
- int **p** () const
- int **n** () const
- int **d** () const
- TimePoint **h** () const
- Real **H** () const
- Real **getBasicIntervalLength** () const
- TimePoint **tau** () const
- Grid & **grid** ()
- const Grid & **grid** () const
- capd::poincare::CrossingDirection **getCrossingDirection** ()
- void **setCrossingDirection** (capd::poincare::CrossingDirection d)
- CoordinateSystem **coords** () const
- void **setCoords** (CoordinateSystem const &c)
- Vector **reference** ()
- Vector **x0** ()
- Matrix **C** ()
- Matrix **invC** ()
- Vector **sectionVector** ()
- Vector **sectionValue** ()
- Section **section** ()
- void dumpData (std::ostream &out, CoordinateSystem const &coords, Vector const &dx, Vector const &r0, Vector const &Xi)
- void **saveData** (PathConfig const &paths, CoordinateSystem const &coords, Vector const &dx, Vector const &r0, Vector const &Xi, std::string extraMsg="")
- CoordinateSystem **loadData** (std::istream &input, Vector &dx, Vector &r0, Vector &Xi)
- CoordinateSystem **loadData** (std::string filepath, Vector &dx, Vector &r0, Vector &Xi)
- template<typename SolutionT >
  void **drawSolution** (std::string dirpath, std::string filename, SolutionT const &X)
- ParamsVector setParams (ParamsVector const &new_params)
- ParamType setParam (size_type index, ParamType const &new_param)

Static Public Attributes

- static const int **PARAMS_COUNT** = EqSpec::getParamsCount() + delaysSpec
- static const int **DIMENSION** = EqSpec::imageDimension()

7.38.1 Detailed Description

template<typename EqSpec, int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11Rect2Policies>
class capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >

allows to supply good coordinates and compute in those coordinates (also, to give data in those coordinates) So for example if you have coordinates C such that the are given by eigenvectors of Jacobian of poincare map then you can slice your data by 1 along any vector j by specifying dx = {0,...,0,1,0,...}, where 1 is on j-th place.

The coordinate change P to good coordinates is: B(dx) = reference + C $*$ dx (so that B(0) == reference) So the backward change is B^{-1} (y) = C^{-1} $*$ (y - reference) Backward change is used in poincare to produce result in 'good' coordinates That is we compute B^{-1}(P(B(.))) applied on a set (dx, r0, Xi) And we get (Pdx, Pr0, PXi) for an easy comparison (coord by coord)

7.38.2 Member Function Documentation

7.38.2.1 constantInitialSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::constant↩
InitialSolution (
            Vector const & vx ) const  [inline]
```

makes a constant solution over basic delay interval [-tau, 0]

7.38.2.2 constantInitialSolution() [2/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::constant↩
InitialSolution (
            Vector const & vx,
            size_type order ) const  [inline]
```

makes a constant solution over basic delay interval [-tau, 0]

7.38.2.3 dataToSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::dataToSolution
(
            CoordinateSystem const & coords,
            Vector dx,
            Vector r0,
            Vector Xi ) const  [inline]
```

Makes a set of the form (reference + C ∗ dx) + C ∗ r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

7.38.2.4 dataToSolution() [2/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::dataToSolution
(
            Vector dx,
            Vector r0,
            Vector Xi ) const  [inline]
```

Makes a set of the form (reference + C ∗ dx) + C ∗ r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

7.38.2.5 dumpData()

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
void capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::dumpData (
            std::ostream & out,
            CoordinateSystem const & coords,
            Vector const & dx,
            Vector const & r0,
            Vector const & Xi )  [inline]
```

this is to dump data as text format (vectors as text representation) Not the best to use in the proofs, but can be of use in preparation of data.

7.38.2.6 functionToSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
template<typename AnyMatrixSpec >
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::functionTo↩
Solution (
            capd::map::Map< AnyMatrixSpec > f ) const  [inline]
```

makes a solution based on the function values over basic delay interval [-tau, 0] The function must be capd::map↩ ::Map type, IMAP works great. Function must be R -> R^dimension, otherwise - exception!

7.38.2.7 functionToSolution() [2/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
template<typename AnyMatrixSpec >
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::functionTo↩
Solution (
            capd::map::Map< AnyMatrixSpec > f,
            int starti ) const  [inline]
```

makes a solution based on the function values over basic delay interval [-tau, 0] The function must be capd::map↩ ::Map type, IMAP works great. Function must be R -> R^dimension, otherwise - exception!

7.38.2.8 poincare() [1/3]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::poincare (
            CoordinateSystem const & in_coords,
            Vector const & dx,
            Vector const & r0,
            Vector const & Xi,
            Real & reachTime,
            Vector & Pdx,
            Vector & Pr0,
            Vector & PXi )  [inline]
```

out_coords are given by the coords of this Helper

7.38.2.9 poincare() [2/3]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::poincare (
            Section section,
            capd::poincare::CrossingDirection crossing_direction,
            Solution const & X,
            Real & reachTime,
            size_type & steps,
            Real & epsilon,
            Solution & PX )  [inline]
```

TODO: (IMPORTANT!!! IN THIS CASE!) docs,

7.38.2.10 poincare() [3/3]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::poincare (
            Vector const & dx,
            Vector const & r0,
            Vector const & Xi,
            Real & reachTime,
            Vector & Pdx,
            Vector & Pr0,
            Vector & PXi )  [inline]
```

in_ and out_ coords are given by the coords of this Helper

7.38.2.11 r0ToSolution() [1/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::r0ToSolution (
            CoordinateSystem const & coords,
            Vector r0,
            Vector Xi ) const  [inline]
```

Makes a set of the form reference + C ∗ r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

7.38.2.12 r0ToSolution() [2/2]

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::r0ToSolution (
            Vector r0,
            Vector Xi ) const  [inline]
```

Makes a set of the form reference + C ∗ r0 It is best to supply r0 as zero centered vector, but the program makes an adjustment to assure that.

7.38.2.13 setParam()

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
ParamType capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::setParam (
            size_type index,
            ParamType const & new_param )  [inline]
```

returns old param value; TODO: add index checking

7.38.2.14 setParams()

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
ParamsVector capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::setParams (
            ParamsVector const & new_params )  [inline]
```

returns old params

7.38.2.15 timemap()

```
template<typename EqSpec , int delaysSpec = 1, typename PoliciesSpec = capd::dynset::C11↩
Rect2Policies>
Solution capd::ddeshelper::RigorousHelper< EqSpec, delaysSpec, PoliciesSpec >::timemap (
            Solution & X,
            const size_type & steps,
            const Real & epsilon = 0.  )  [inline]
```

computes timemap using the solution as initial data WARNING: it will extend the solution given in X by a given number of steps! returns: it returns the last segment of the solution in a shape of the same representation as the initial segment of X. if the segment cannot be produced (due to continuity class issues) the exception will be thrown. if epsilon is given as != 0. then the procedure will try to do the epsilonStep procedure.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperRigorous.h
- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddeshelper/D↩ DEHelperRigorous.hpp

7.39 capd::ddes::RosslerDelay< ScalarSpec, ParamSpec > Class Template Reference

`#include <SampleEqns.h>`

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**

Public Member Functions

- **RosslerDelay** (ParamType a, ParamType b, ParamType c)
- **RosslerDelay** (RosslerDelay const &orig)
- template<typename VecSpec >
  RosslerDelay (VecSpec const &params)
- RosslerDelay & **operator=** (RosslerDelay const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type **dimension** ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType a
- ParamType b
- ParamType c
- ParamType epsi

7.39.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >

A Toy Example for proof of chaotic behaviour in DDE (chaotic ODE perturbed by a small DDE term - delay might be big, but perturbation should be small) The equation is:

```
v'(t) = f(v(t)) + \epsilon f(v(t-\tau))
```

where v = (x, y, z) \in \R^3 and f is the r.h.s. of original R"ossler ODE:

f(x, y, z) = (-(y+z), (x + ay), (b + z * (x - c)))

The parameters are: a, b, c (as in R"ossler) and \epsilon.

7.39.2 Member Function Documentation

7.39.2.1 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline],
[static]
```

number of parameters to fully configure equation

7.39.2.2 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::imageDimension ( )  [inline],
[static]
```

output dimension of the internal map

7.39.3 Member Data Documentation

7.39.3.1 a

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::a  [protected]
```

classic rossler parameters, see scholarpedia.

7.39.3.2 b

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::b  [protected]
```

classic rossler parameters, see scholarpedia.

7.39.3.3 c

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::c  [protected]
```

classic rossler parameters, see scholarpedia.

7.39.3.4 epsi

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
ParamType capd::ddes::RosslerDelay< ScalarSpec, ParamSpec >::epsi  [protected]
```

extra parameter for a size of the delayed term

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/SampleEqns.h

7.40 capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef ScalarSpec **ScalarType**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef int **size_type**

Public Member Functions

- **ScalarLinearMap** (ScalarType const &a, ScalarType const &b, ScalarType const &c)
- **ScalarLinearMap** (ScalarLinearMap const &orig)
- ScalarLinearMap & **operator=** (ScalarLinearMap const &orig)
- size_type imageDimension () const
- size_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static std::string **show** ()

Protected Attributes

- ParamType **a**
- ParamType **b**
- ParamType **c**

7.40.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >

Scalar linear equation of the form:

```
x'(t) = a * x(t) + b * x(t-tau) + c
```

Use with 'DiscreteDelaysFunctionalMap' to define tau

7.40.2 Member Function Documentation

7.40.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
size_type capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.40.2.2 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
size_type capd::ddes::ScalarLinearMap< ScalarSpec, ParamSpec >::imageDimension ( ) const
[inline]
```

output dimension of the internal map

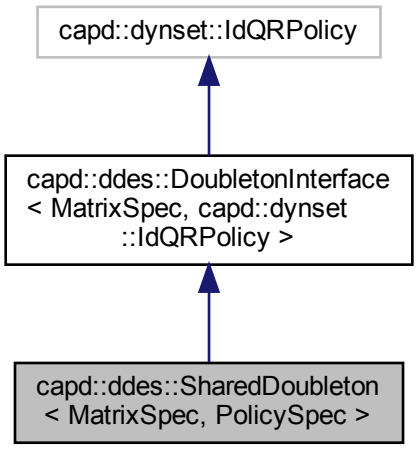
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/SampleEqns.h

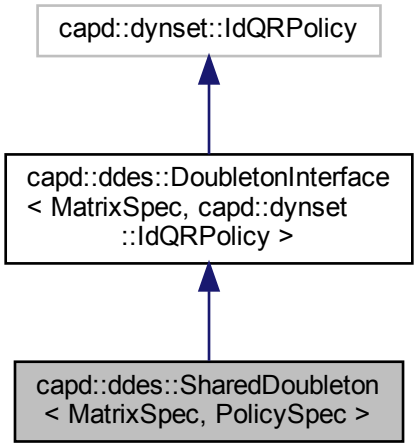
7.41 capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec > Class Template Reference

```
#include <SharedDoubleton.h>
```

Inheritance diagram for capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >:



Collaboration diagram for capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >:



Public Types

- typedef MatrixSpec **MatrixType**
- typedef MatrixType::RowVectorType **VectorType**
- typedef MatrixType::ScalarType **ScalarType**
- typedef MatrixType::size_type **size_type**
- typedef SharedDoubleton **Class**
- typedef DoubletonInterface< MatrixSpec, PolicySpec > **BaseClass**
- typedef VectorType * **VectorTypePtr**
- typedef MatrixType * **MatrixTypePtr**
- typedef std::bitset< 6 > **OwnershipType**
- typedef PolicySpec **Policy**
- typedef PolicySpec **QRPolicy**

Public Member Functions

- Class & operator= (Class const &orig)
- SharedDoubleton ()
- SharedDoubleton (VectorType const &x, VectorType *set_external_r0=0, bool passOwnership=false)
- SharedDoubleton (Class const &orig)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0, MatrixType const &B, MatrixType const &Binv, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0, MatrixType const &B, Vector↩ Type const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0, MatrixType const &B, Matrix↩ Type const &Binv, VectorType const &r)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType const &r0)
- SharedDoubleton (VectorType const &x, MatrixType const &C, VectorType *r0)

- SharedDoubleton (VectorType ∗x, MatrixType ∗C, VectorType ∗r0, MatrixType ∗B, VectorType ∗r)
- SharedDoubleton (size_type d, size_type N0=-1)
- virtual ∼SharedDoubleton ()
- size_type storageN0 () const

  < *import dimension from interface*
- size_type storageDimension () const
- VectorType ∗ take_x ()
- MatrixType ∗ take_C ()
- VectorType ∗ take_r0 ()
- MatrixType ∗ take_B ()
- VectorType ∗ take_r ()
- MatrixType ∗ take_Binv ()
- VectorType get_x () const
- MatrixType get_C () const
- VectorType get_r0 () const
- MatrixType get_B () const
- VectorType get_r () const
- MatrixType get_Binv () const
- BaseClass & set_x (VectorType const &x)
- BaseClass & set_C (MatrixType const &C)
- BaseClass & set_r0 (VectorType const &r0)
- BaseClass & set_B (MatrixType const &B)
- BaseClass & set_r (VectorType const &r)
- BaseClass & set_Cr0 (MatrixType const &C, VectorType const &r0)
- BaseClass & set_Binv (MatrixType const &Binv)
- VectorType **midPoint** () const
- VectorType **hull** () const
- BaseClass & set_x (VectorType ∗x, bool passOwnership=false)
- BaseClass & set_C (MatrixType ∗C, bool passOwnership=false)
- BaseClass & set_r0 (VectorType ∗r0, bool passOwnership=false)
- BaseClass & set_Cr0 (MatrixType ∗C, VectorType ∗r0, bool passCOwnership=false, bool passR0↩ Ownership=false)
- BaseClass & set_B (MatrixType ∗B, bool passOwnership=false)
- BaseClass & set_r (VectorType ∗r, bool passOwnership=false)
- BaseClass & set_Binv (MatrixType ∗Binv, bool passOwnership=false)
- bool common_x (VectorType const ∗x) const
- bool common_C (MatrixType const ∗C) const
- bool common_r0 (VectorType const ∗r0) const
- bool common_B (MatrixType const ∗B) const
- bool common_r (VectorType const ∗r) const
- bool common_Binv (MatrixType const ∗Binv) const
- BaseClass & add (VectorType const &v)
- BaseClass & add (BaseClass const &set)
- BaseClass & mul (ScalarType const &c)
- BaseClass & mulThenAdd (ScalarType const &c, BaseClass const &set)
- BaseClass & affineTransform (MatrixType const &M, VectorType const &v)
- BaseClass & translate (VectorType const &v)
- std::string **show** () const
- virtual void reinitialize (size_type d, size_type N0)
- void **reinit** (const VectorType &x, const MatrixType &C, const VectorType &r0, const MatrixType &B, const VectorType &r)
- void **reinit** (const VectorType ∗x, const MatrixType ∗C, const VectorType ∗r0, const MatrixType ∗B, const VectorType ∗r)
- virtual size_type dimension () const

Protected Types

- enum {
  OWNERBIT_x = 5, **OWNERBIT_C** = 4, **OWNERBIT_r0** = 3, **OWNERBIT_B** = 2,
  OWNERBIT_r = 1, **OWNERBIT_Binv** = 0 }

Protected Member Functions

- void updateBinv ()

  bitset "I own nothing" - used for binary operations
- void rawSetup (size_type dim, size_type N0, const VectorType ∗x, const MatrixType ∗C, const VectorType ∗r0, const MatrixType ∗B, const VectorType ∗r, const MatrixType ∗Binv, OwnershipType ownership)
- void assureOwner (OwnershipType const &what=OWN_ALL)
- void sanityCheck (std::string const &what="") const
- void deallocate (OwnershipType const &what=OWN_ALL)
- void allocate (size_type d, size_type N0=0, OwnershipType const &what=OWN_ALL)
- void reallocate (size_type d, size_type N0=0, OwnershipType const &what=OWN_ALL)

Protected Attributes

- VectorTypePtr **m_x**
- MatrixTypePtr **m_C**
- VectorTypePtr **m_r0**
- MatrixTypePtr **m_B**
- VectorTypePtr **m_r**
- MatrixTypePtr **m_Binv**
- OwnershipType **m_owner**

Static Protected Attributes

- static OwnershipType **OWN_x**
- static OwnershipType OWN_C

  bitset "I own only x" - used for binary operations
- static OwnershipType OWN_r0

  similarly to above - used for binary operations
- static OwnershipType OWN_B

  similarly to above - used for binary operations
- static OwnershipType OWN_r

  similarly to above - used for binary operations
- static OwnershipType OWN_Binv

  similarly to above - used for binary operations
- static OwnershipType OWN_ALL

  similarly to above - used for binary operations
- static OwnershipType OWN_NONE

  bitset "I own everything" - used for binary operations

Additional Inherited Members

7.41.1 Detailed Description

template<typename MatrixSpec, typename PolicySpec = capd::dynset::IdQRPolicy>
class capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >

This class stores a set X \in \R^d of the form:

X = x + C ∗ r0 + B ∗ r

with $x \in \R^d$, $r0, r$ - closed intervals centered at 0, $r0 \subset \R^{N_0}$, $C \in Lin(\R^{N_0}, \R^d)$, $B \in Lin(\R^d, \R^d)$, $r \in \R^d$, B being a matrix easy to invert, e.g. B = ID (Interval form of the remainder) or B orthogonal (Doubleton set with QR decomposition).

NOTE: compare to the classic Lohner Doubleton Set in CAPD, where authors assume $N_0 = d$) NOTE: the set allows for its components to be set outside of the object and controlled there. We use pointers for this purpose. However, we always work with the set to assure that no pointer is NULL. If user does not supply us with a pointer, then we are creating a default one instead. We use the fact that CAPD can have Vectors of dimension 0 and Matrices of dimension (d, 0) and (0, d) and algebra is well defined for them.

NOTE: This is quite long and tedious code in C++ because of C++ This should be heavily tested for correctedness and memory leaks Reader (e.g. reviewer of the manuscript for publication) should not worry to check that code

TODO: (NOT URGENT) move big implementation into .hpp part.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 SharedDoubleton() [1/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton ( )  [inline]
```

default constructor makes a 1D point-set at 0

7.41.2.2 SharedDoubleton() [2/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            VectorType * set_external_r0 = 0,
            bool passOwnership = false )  [inline]
```

setup set with a given vector, if it is an interval vector, then it be split into midPoint and the error part, which goes to B∗r part (error part). If you want some structure (i.e. Lohner part C ∗ r0) then use appropriate constructor.

7.41.2.3 SharedDoubleton() [3/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            Class const & orig )  [inline]
```

copies the original set (ownership is preserved)

7.41.2.4 SharedDoubleton() [4/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0,
            MatrixType const & B,
            VectorType const & r )  [inline]
```

setup this set with a given data, set owns everything

7.41.2.5 SharedDoubleton() [5/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0,
            MatrixType const & B,
            MatrixType const & Binv,
            VectorType const & r )  [inline]
```

setup this set with a given data, set owns everything

7.41.2.6 SharedDoubleton() [6/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0,
            MatrixType const & B,
            VectorType const & r )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.41.2.7 SharedDoubleton() [7/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0,
            MatrixType const & B,
            MatrixType const & Binv,
            VectorType const & r )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.41.2.8 SharedDoubleton() [8/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType const & r0 )  [inline]
```

setup this set with a given data, set is owner of everything

7.41.2.9 SharedDoubleton() [9/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType const & x,
            MatrixType const & C,
            VectorType * r0 )  [inline]
```

setup this set with a given data but the set does not own the r0 (user is responsible for deleting it)

7.41.2.10 SharedDoubleton() [10/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            VectorType * x,
            MatrixType * C,
            VectorType * r0,
            MatrixType * B,
            VectorType * r )  [inline]
```

setup this set with a given data but the set does not own the data (user is responsible for deleting)

7.41.2.11 SharedDoubleton() [11/11]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::SharedDoubleton (
            size_type d,
            size_type N0 = -1 )  [inline]
```

setup this set as zero vector, but the structure of given dimensions. If second arg is < 0 then d is used instead.

7.41.2.12 ~SharedDoubleton()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
virtual capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::~SharedDoubleton ( )  [inline],
[virtual]
```

standard thing

7.41.3 Member Function Documentation

7.41.3.1 add() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::add (
            BaseClass const & set )  [inline]
```

reimplemented for better perfoprmance

7.41.3.2 add() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::add (
            VectorType const & v )  [inline], [virtual]
```

reimplemented for better perfoprmance

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.3 affineTransform()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::affineTransform (
            MatrixType const & M,
            VectorType const & v )  [inline], [virtual]
```

applies in a smart way to this set X the affine transform f(y) = M ∗ (X - v)

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.4 allocate()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::allocate (
            size_type d,
            size_type N0 = 0,
            OwnershipType const & what = OWN_ALL )  [inline], [protected]
```

allocate specified elements of the object (default: all) so it is d- dimensional, and has N0 dimensional r0 part.

7.41.3.5 assureOwner()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::assureOwner (
            OwnershipType const & what = OWN_ALL )  [inline], [protected]
```

does nothing if is owner of others, otherwise it reasigns the memory and copies values form the external data

7.41.3.6 common_B()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_B (
            MatrixType const * B ) const  [inline], [virtual]
```

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.7 common_Binv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_Binv (
            MatrixType const * Binv ) const  [inline]
```

checked by pointer equality

7.41.3.8 common_C()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_C (
            MatrixType const * C ) const  [inline], [virtual]
```

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.9 common_r()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_r (
            VectorType const * r ) const  [inline], [virtual]
```

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.10 common_r0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_r0 (
            VectorType const * r0 ) const  [inline], [virtual]
```

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.11 common_x()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
bool capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::common_x (
            VectorType const * x ) const  [inline], [virtual]
```

checked by pointer equality

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.12 deallocate()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::deallocate (
            OwnershipType const & what = OWN_ALL )  [inline], [protected]
```

safely removes all specified data (default: all) from the object, checking the ownership if neccesary.

7.41.3.13 dimension()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
virtual size_type capd::ddes::DoubletonInterface< MatrixSpec, PoliciesSpec >::dimension  [inline]
```

by default dimension is equal to storage dimension

7.41.3.14 get_B()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_B ( ) const  [inline]
```

see interface docs

7.41.3.15 get_Binv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_Binv ( ) const  [inline]
```

see interface docs

7.41.3.16 get_C()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_C ( ) const  [inline]
```

see interface docs

7.41.3.17 get_r()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_r ( ) const  [inline]
```

see interface docs

7.41.3.18 get_r0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_r0 ( ) const  [inline]
```

see interface docs

7.41.3.19 get_x()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::get_x ( ) const  [inline]
```

see interface docs

7.41.3.20 mul()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::mul (
             ScalarType const & c )  [inline], [virtual]
```

reimplemented for better performance

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.21 mulThenAdd()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::mulThenAdd (
            ScalarType const & c,
            BaseClass const & set )  [inline]
```

reimplemented for better perfoprmance

7.41.3.22 operator=()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
Class& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::operator= (
            Class const & orig )  [inline]
```

assign operator - it needs to deallocate memory if necessary, then setup set anew

7.41.3.23 rawSetup()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::rawSetup (
            size_type dim,
            size_type N0,
            const VectorType * x,
            const MatrixType * C,
            const VectorType * r0,
            const MatrixType * B,
            const VectorType * r,
            const MatrixType * Binv,
            OwnershipType ownership )  [inline], [protected]
```

Low level setup function to be called in constructors for DRY. This does not deallocate memory! For safer version see public setup().

7.41.3.24 reallocate()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::reallocate (
            size_type d,
            size_type N0 = 0,
            OwnershipType const & what = OWN_ALL )  [inline], [protected]
```

dealocates and allocates new memory for specified elements (default: all)

7.41.3.25 reinitialize()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
virtual void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::reinitialize (
            size_type d,
            size_type N0 )  [inline], [virtual]
```

see base class

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.26 sanityCheck()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::sanityCheck (
            std::string const & what = "" ) const  [inline], [protected]
```

helper function to check if the object represents sane data (all dimensions compatible and all pointers set if neccesary) should be called after constructing or manipulating object

7.41.3.27 set_B() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_B (
            MatrixType * B,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.28 set_B() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_B (
            MatrixType const & B )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.29 set_Binv() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_Binv (
            MatrixType * Binv,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.30 set_Binv() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_Binv (
            MatrixType const & Binv )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.31 set_C() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_C (
            MatrixType * C,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.32 set_C() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_C (
            MatrixType const & C )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.33 set_Cr0() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_Cr0 (
            MatrixType * C,
            VectorType * r0,
            bool passCOwnership = false,
            bool passR0Ownership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.34 set_Cr0() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_Cr0 (
            MatrixType const & C,
            VectorType const & r0 )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.35 set_r() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_r (
            VectorType * r,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.36 set_r() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_r (
            VectorType const & r )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.37 set_r0() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_r0 (
            VectorType * r0,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.38 set_r0() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_r0 (
            VectorType const & r0 )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.39 set_x() [1/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_x (
            VectorType * x,
            bool passOwnership = false )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!)

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.40 set_x() [2/2]

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::set_x (
            VectorType const & x )  [inline], [virtual]
```

Note: the element must be compatible with other elements (dimensions!) or we throw exception!

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.41 storageDimension()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::storageDimension ( ) const
[inline]
```

see interface description

7.41.3.42 storageN0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
size_type capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::storageN0 ( ) const  [inline]
```

< import dimension from interface

see interface description

7.41.3.43 take_B()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_B ( )  [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.44 take_Binv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_Binv ( )  [inline],
[virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.45 take_C()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
MatrixType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_C ( )  [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.46 take_r()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_r ( )  [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.47 take_r0()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_r0 ( )  [inline],
[virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.48 take_x()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
VectorType* capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::take_x ( )  [inline], [virtual]
```

unmark itself as owner and returns the pointer. From now on user is responsible for deleting. Use with caution.

Reimplemented from capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.49 translate()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
BaseClass& capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::translate (
            VectorType const & v )  [inline], [virtual]
```

applies in a smart way to this set X the transform f(y) = X - v

Implements capd::ddes::DoubletonInterface< MatrixSpec, capd::dynset::IdQRPolicy >.

7.41.3.50 updateBinv()

```
template<typename MatrixSpec , typename PolicySpec = capd::dynset::IdQRPolicy>
void capd::ddes::SharedDoubleton< MatrixSpec, PolicySpec >::updateBinv ( )  [inline], [protected]
```

bitset "I own nothing" - used for binary operations

assures m_Binv holds real inverse of B

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/storage/Shared↩ Doubleton.h

7.42 capd::ddes::DiscreteTimeGrid< RealSpec >::TimePointType Class Reference

Public Types

- typedef RealSpec **RealType**

Public Member Functions

- **operator int** () const
- int **toInt** () const
- **operator RealType** () const
- bool **isZero** () const
- std::string **show** ()
- TimePointType **operator+=** (TimePointType const &b)
- TimePointType **operator-=** (TimePointType const &b)
- **TimePointType** (TimePointType const &orig)
- TimePointType & **operator=** (TimePointType const &orig)

Static Public Member Functions

- static std::string badge ()

Protected Member Functions

- void **checkGridCompatible** (TimePointType const &other, std::string extraInfo="") const
- **TimePointType** (RealType const &h, int i)

Protected Attributes

- const RealSpec & **m_h**
- int **m_i**

Friends

- class **DiscreteTimeGrid**< **RealType** >
- TimePointType **operator+** (TimePointType const &a, TimePointType const &b)
- TimePointType **operator-** (TimePointType const &a, TimePointType const &b)
- TimePointType **operator-** (const TimePointType &a)
- TimePointType **operator*** (RealType const &a, TimePointType const &b)
- TimePointType **operator*** (TimePointType const &a, RealType const &b)
- bool **operator==** (TimePointType const &a, TimePointType const &b)
- bool **operator<** (TimePointType const &a, TimePointType const &b)
- bool **operator>** (TimePointType const &a, TimePointType const &b)
- bool **operator<=** (TimePointType const &a, TimePointType const &b)
- bool **operator>=** (TimePointType const &a, TimePointType const &b)
- std::ostream & **operator<<** (std::ostream &out, TimePointType const &t)
- std::istream & **operator>>** (std::istream &in, TimePointType &t)

7.42.1 Member Function Documentation

7.42.1.1 badge()

```
template<typename RealSpec >
static std::string capd::ddes::DiscreteTimeGrid< RealSpec >::TimePointType::badge ( )  [inline],
[static]
```

Badge must be a single word!

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ Common.h

7.43 capd::ddes::ToyModel Class Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef long **size_type**

Public Member Functions

- **ToyModel** (ToyModel const &orig)
- ToyModel & **operator=** (ToyModel const &orig)
- size_type imageDimension () const
- size_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static std::string **show** ()

7.43.1 Detailed Description

Scalar eq. x(t) ∗ x(t-tau).

Use with 'DiscreteDelaysFunctionalMap' to define tau

7.43.2 Member Function Documentation

7.43.2.1 dimension()

```
size_type capd::ddes::ToyModel::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.43.2.2 imageDimension()

```
size_type capd::ddes::ToyModel::imageDimension ( ) const  [inline]
```

output dimension of the internal map

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

7.44 capd::ddes::ToyModelSq Class Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef long **size_type**

Public Member Functions

- **ToyModelSq** (ToyModelSq const &orig)
- ToyModelSq & **operator=** (ToyModelSq const &orig)
- size_type imageDimension () const
- size_type dimension () const
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec > void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static std::string **show** ()

7.44.1 Detailed Description

Scalar eq. Depends only on the past, f(x(t), x(t-1)) = 2∗x(t-1) + (x(t-1))^2. Used for tests.

Use with 'DiscreteDelaysFunctionalMap' to define tau

7.44.2 Member Function Documentation

7.44.2.1 dimension()

```
size_type capd::ddes::ToyModelSq::dimension ( ) const  [inline]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.44.2.2 imageDimension()

```
size_type capd::ddes::ToyModelSq::imageDimension ( ) const  [inline]
```

output dimension of the internal map

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/SampleEqns.h

7.45 capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec > Class Template Reference

```
#include <SampleEqns.h>
```

Public Types

- typedef unsigned int **size_type**
- typedef ParamSpec **ParamType**
- typedef ScalarSpec **ScalarType**
- typedef ScalarType **RealType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarSpec, 0 > **VectorType**

Public Member Functions

- **ToyModelSqA** (ParamType a)
- **ToyModelSqA** (ToyModelSqA const &orig)
- **ToyModelSqA** (ParamsVectorType const &params)
- ToyModelSqA & **operator=** (ToyModelSqA const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void **operator()** (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Public Attributes

- ParamSpec **a**
- ParamSpec **c**

7.45.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec>
class capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >

Scalar eq. Depends only on the past, f(x(t), x(t-1)) = -a∗x(t-1) + c ∗ x(t-1) + (x(t-1))^2. For c = 0 it has Hopf biffurcation at a^∗ = (5∗pi) / (3∗sqrt(3)) It has 2 fixed points: 0 and a, a is repelling with one unstable to infty and to 0, 0 is stable Param c is used to move the fixed point a to 0, set c = 2a Use with 'DiscreteDelaysFunctionalMap' to define tau. Tau should be 1

7.45.2 Member Function Documentation

7.45.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::dimension ( )  [inline],
[static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.45.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline],
[static]
```

number of parameters to fully configure equation

7.45.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec >
static size_type capd::ddes::ToyModelSqA< ScalarSpec, ParamSpec >::imageDimension ( )  [inline],
[static]
```

output dimension of the internal map

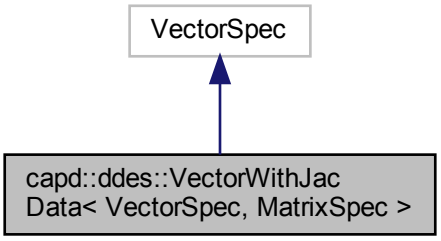
The documentation for this class was generated from the following file:

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

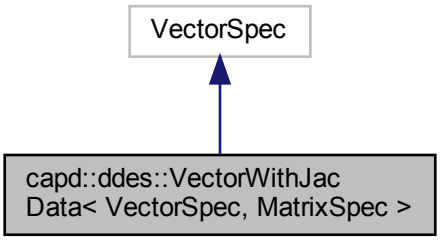
7.46 capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec > Class Template Reference

```
#include <DDENonrigorousTaylorSolver.h>
```

Inheritance diagram for capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >:



Collaboration diagram for capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >:



Public Types

- typedef MatrixSpec **MatrixType**
- typedef MatrixSpec::ScalarType **ScalarType**
- typedef VectorWithJacData< VectorSpec, MatrixSpec > **Class**
- typedef VectorSpec **BaseClass**
- typedef BaseClass::size_type **size_type**
- typedef std::vector< MatrixSpec > **MatrixStorageType**

Public Member Functions

- **VectorWithJacData** (size_type d=0)
- **VectorWithJacData** (const VectorWithJacData &orig)
- **VectorWithJacData** (const VectorSpec &orig)
- **VectorWithJacData** (const VectorSpec &v, const std::vector< MatrixSpec > &D)
- **VectorWithJacData** (const VectorSpec &v, const MatrixSpec &D)
- VectorWithJacData & **setMatrix** (MatrixType const &D)
- VectorWithJacData & **operator=** (const VectorWithJacData &orig)
- VectorWithJacData & **operator=** (const BaseClass &orig)
- VectorWithJacData & **operator*=** (ScalarType const &c)
- VectorWithJacData & **operator+=** (VectorWithJacData const &other)
- **operator VectorSpec** ()
- **operator MatrixSpec** ()
- const MatrixStorageType & **getMatrixData** () const
- MatrixStorageType & **getMatrixData** ()

Protected Attributes

- std::vector< MatrixSpec > **m_Jac**

7.46.1 Detailed Description

template<typename VectorSpec, typename MatrixSpec>
class capd::ddes::VectorWithJacData< VectorSpec, MatrixSpec >

a class to hold additional data to compute Jacobian of the (nonrigorous / approximate) flow w.r.t. initial data.

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/DDE↩ NonrigorousTaylorSolver.h

7.47 capd::ddes::Wischert< ScalarSpec, ParamSpec > Class Template Reference

`#include <SampleEqns.h>`

Public Types

- typedef ScalarSpec **ScalarType**
- typedef unsigned int **size_type**
- typedef ScalarType **RealType**
- typedef ParamSpec **ParamType**
- typedef capd::vectalg::Vector< ParamSpec, 0 > **ParamsVectorType**
- typedef capd::vectalg::Vector< ScalarType, 0 > **VectorType**

Public Member Functions

- **Wischert** (ParamType a=2.)
- **Wischert** (Wischert const &orig)
- **Wischert** (capd::vectalg::Vector< ParamSpec, 0 > const &params)
- Wischert & **operator=** (Wischert const &orig)
- template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
  void operator() (const RealSpec &t, const InVectorSpec x, OutVectorSpec &fx) const

Static Public Member Functions

- static size_type imageDimension ()
- static size_type dimension ()
- static size_type getParamsCount ()
- static std::string **show** ()

Protected Attributes

- ParamType **a**

7.47.1 Detailed Description

template<typename ScalarSpec, typename ParamSpec = ScalarSpec>
class capd::ddes::Wischert< ScalarSpec, ParamSpec >

x'(t) = a∗sin(x(t-1))

7.47.2 Member Function Documentation

7.47.2.1 dimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert< ScalarSpec, ParamSpec >::dimension ( )  [inline], [static]
```

input dimension of the internal map, x(t) is one, x(t-tau_1) is another, etc.. (thus the formula)

7.47.2.2 getParamsCount()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert< ScalarSpec, ParamSpec >::getParamsCount ( )  [inline],
[static]
```

number of parameters to fully configure equation

7.47.2.3 imageDimension()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
static size_type capd::ddes::Wischert< ScalarSpec, ParamSpec >::imageDimension ( )  [inline],
[static]
```

output dimension of the internal map

7.47.2.4 operator()()

```
template<typename ScalarSpec , typename ParamSpec = ScalarSpec>
template<typename RealSpec , typename InVectorSpec , typename OutVectorSpec >
void capd::ddes::Wischert< ScalarSpec, ParamSpec >::operator() (
            const RealSpec & t,
            const InVectorSpec x,
            OutVectorSpec & fx ) const  [inline]
```

evaluation of rhs

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

- /home/robson/ROBERT-PRACA-CHMURA/eclipse-workspace/capdDDEs5.1.2/include/capd/ddes/Sample↩ Eqns.h

Index