# RodSystemEstimator

Generated by Doxygen 1.9.3

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

# Chapter 3

# File Index

## 3.1 File List

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

# Chapter 4

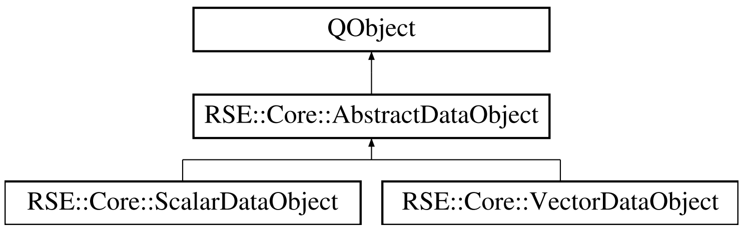
# Class Documentation

## 4.1 RSE::Core::AbstractDataObject Class Reference

Data object which is designied in the way to be represented in a table easily.

```
#include <abstractdataobject.h>
```

Inheritance diagram for RSE::Core::AbstractDataObject:


QObject
RSE::Core::AbstractDataObject
RSE::Core::ScalarDataObject
RSE::Core::VectorDataObject

### Public Types

- enum **ObjectType** { **kScalar** , **kVector** , **kMatrix** , **kSurface** }

### Public Member Functions

- **AbstractDataObject** (ObjectType type, QString const &name)

  *Base constructor.*
- virtual AbstractDataObject ∗ clone () const =0
- virtual DataItemType & addItem (DataKeyType key)=0
- void **removeItem** (DataValueType key)

  *Remove the entity paired to the specified key.*
- bool **changeItemKey** (DataKeyType oldKey, DataKeyType newKey, DataHolder ∗items=nullptr)

  *Modify an existing key.*
- bool **setArrayValue** (DataKeyType key, DataValueType newValue, IndexType iRow=0, IndexType iColumn=0)

  *Set an array value with the specified indices.*
- DataValueType **arrayValue** (DataKeyType key, IndexType iRow=0, IndexType iColumn=0)

  *Retrieve a value from an array.*

- std::vector< DataKeyType > **keys** () const

  *Retrieve all the keys.*
- quint32 **numberItems** () const
- DataHolder const & **getItems** ()
- DataIDType **id** () const
- ObjectType **type** () const
- QString const & **name** () const
- void **setName** (QString const &name)
- virtual void **serialize** (QDataStream &stream) const

  *Serialize an abstract data object.*
- virtual void deserialize (QDataStream &stream)

  *Partly deserialize an abstract data object.*
- virtual void import (QTextStream &stream)=0
- void **write** (QTextStream &stream) const

  *Write an abstract data object to a file.*

## Static Public Member Functions

- static DataIDType **maxObjectID** ()
- static void **setMaxObjectID** (DataIDType iMaxObjectID)

## Protected Attributes

- const ObjectType **mkType**
- QString **mName**
- DataIDType **mID**
- DataHolder **mItems**

## Static Private Attributes

- static DataIDType **smMaxObjectID** = 0

## Friends

- QDataStream & **operator**<< (QDataStream &stream, AbstractDataObject const &obj)

  *Print a data object to a binary stream.*

### 4.1.1 Detailed Description

Data object which is designied in the way to be represented in a table easily.

### 4.1.2 Member Function Documentation

#### 4.1.2.1 addItem()

```
virtual DataItemType & RSE::Core::AbstractDataObject::addItem (
             DataKeyType key )  [pure virtual]
```

Implemented in RSE::Core::ScalarDataObject, and RSE::Core::VectorDataObject.

#### 4.1.2.2 clone()

```
virtual AbstractDataObject * RSE::Core::AbstractDataObject::clone ( ) const  [pure virtual]
```

Implemented in RSE::Core::ScalarDataObject, and RSE::Core::VectorDataObject.

#### 4.1.2.3 deserialize()

```
void AbstractDataObject::deserialize (
             QDataStream & stream )  [virtual]
```

Partly deserialize an abstract data object.

It is assumed that a type and name have already been assigned. So, only an identifier and items need to be set.

#### 4.1.2.4 import()

```
virtual void RSE::Core::AbstractDataObject::import (
             QTextStream & stream )  [pure virtual]
```

Implemented in RSE::Core::ScalarDataObject, and RSE::Core::VectorDataObject.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/abstractdataobject.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/abstractdataobject.cpp

## 4.2 RSE::Core::Array< T > Class Template Reference

Numerical array class.

```
#include <array.h>
```

### Classes

- class Row

  *Proxy class to acquire a row by index.*

## Public Member Functions

- **Array** (IndexType numRows=0, IndexType numCols=0)
- **Array** (Array< T > const &another)

  *Copy constructor.*
- **Array** (Array< T > &&another)

  *Move constructor.*
- T ∗ **data** ()
- void **resize** (IndexType numRows, IndexType numCols)

  *Resize and copy previous values if possible.*
- void **removeColumn** (IndexType iRemoveColumn)

  *Remove a column by index.*
- void **swapColumns** (IndexType iFirstColumn, IndexType iSecondColumn)

  *Swap two columns.*
- void **clear** ()

  *Remove all the values.*
- IndexType **rows** () const
- IndexType **cols** () const
- IndexType **size** () const
- Row< T > **operator[ ]** (IndexType iRow)
- Row< T > **operator[ ]** (IndexType iRow) const
- Array & **operator=** (Array< T > const &another)

  *Assignment operator.*

## Private Attributes

- IndexType **mNumRows**

  *Number of rows.*
- IndexType **mNumCols**

  *Number of columns.*
- T ∗ **mpData** = nullptr

  *Pointer to the data stored.*

## Friends

- template<typename K >
  QDebug **operator**<< (QDebug stream, Array< K > &array)

  *Print all array values using the matrix format.*
- template<typename K >
  QDataStream & **operator**<< (QDataStream &stream, Array< K > const &array)

  *Write an array to a binary stream.*
- template<typename K >
  QDataStream & **operator**>> (QDataStream &stream, Array< K > &array)

  *Read an array from a stream.*
- template<typename K >
  QTextStream & **operator**<< (QTextStream &stream, Array< K > const &array)

  *Write an array to a text stream.*

### 4.2.1 Detailed Description

**template<typename T>**
**class RSE::Core::Array< T >**

Numerical array class.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/array.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/array.cpp

## 4.3 RSE::Core::Cable Struct Reference

Mechanical properties of a cable.

```
#include <databasecables.h>
```

### Public Attributes

- std::string **name**

  *Name of a cable.*
- double **bendingStiffness**

  *Bending stiffness, N.*
- double **torsionalStiffness**

  *Torsional stiffness, N.*
- double **massPerLength**

  *Mass per length, kg/m.*
- double **youngsModulus**

  *Youngs modulus, Pa.*
- double **area**

  *Area of a cross-section, m^2.*

### 4.3.1 Detailed Description

Mechanical properties of a cable.

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

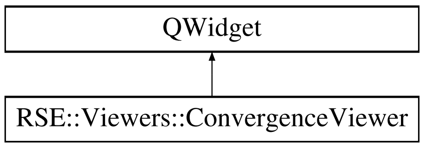
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/databasecables.h

## 4.4 RSE::Viewers::ConvergenceViewer Class Reference

Class to represent convergence of viscosities.

```
#include <convergenceviewer.h>
```

Inheritance diagram for RSE::Viewers::ConvergenceViewer:



### Public Member Functions

- **ConvergenceViewer** (QString const &pathFile, QWidget ∗pParent=nullptr)
- void **plot** ()

  *Represent the convergence.*

### Private Member Functions

- void **initialize** ()

  *Initialize the widget.*
- bool **read** ()

  *Read the file contained viscosities of dampers.*

### Private Attributes

- QString const **mkPathFile**
- QCustomPlot ∗ **mpFigure**
- QStringList **mAvailableColors**
- QVector< QCPScatterStyle::ScatterShape > **mAvailableShapes**
- QVector< int > **mCalcModes**
- Core::Array< double > **mDampingValues**

### 4.4.1 Detailed Description

Class to represent convergence of viscosities.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/viewers/convergenceviewer.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/viewers/convergenceviewer.cpp

## 4.5 RSE::Core::Damper Class Reference

Class to compute and collect properties of a damper.

```
#include <damper.h>
```

### Public Member Functions

- **Damper** (double massCable, double massLoadedCable, double workingLength, double bouncerLength, double springLength=0, double springStiffness=0)
- double **massCable** () const
- double **massLoadedCable** () const
- double **workingLength** () const
- double **bouncerLength** () const
- double **springLength** () const
- double **springStiffness** () const
- void **setMassCable** (double massCable)
- void **setMassLoadedCable** (double massLoadedCable)
- void **setWorkingLength** (double workingLength)
- void **setBouncerLength** (double bouncerLength)
- void **setSpringLength** (double springLength)
- void **setSpringStiffness** (double springStiffness)
- void **computeSpring** ()

  *Compute parameters of a spring belonged to a damper.*

### Private Attributes

- double **mMassCable**

  *Mass of a cable, kg.*
- double **mMassLoadedCable**

  *Mass of a cable with ice on it, kg.*
- double **mWorkingLength**

  *Working length, m.*
- double **mBouncerLength**

  *Length of a bouncer, m.*
- double **mSpringLength** = 0.0

  *Length of a spring, m.*
- double **mSpringStiffness** = 0.0

  *Spring stiffness, N/m.*

### 4.5.1 Detailed Description

Class to compute and collect properties of a damper.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/damper.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/damper.cpp

## 4.6 RSE::Core::DataBaseCables Class Reference

Aggregate data of cables.

```
#include <databasecables.h>
```

### Public Member Functions

- **DataBaseCables** (QString const &directory, QString const &fileName)
- std::vector< std::string > **names** () const

  *Names of available cables.*
- Cable const & **getItem** (std::string const &name) const

### Private Member Functions

- bool **readDataBase** (QString const &pathFile)

  *Read a database from a file.*

### Private Attributes

- std::unordered_map< std::string, Cable > **mData**

### 4.6.1 Detailed Description

Aggregate data of cables.

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

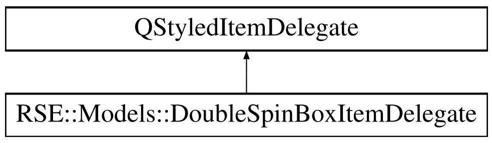
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/databasecables.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/databasecables.cpp

## 4.7 RSE::Models::DoubleSpinBoxItemDelegate Class Reference

Class to specify how table values can be edited.

```
#include <doublespinboxitemdelegate.h>
```

Inheritance diagram for RSE::Models::DoubleSpinBoxItemDelegate:



## Public Member Functions

- **DoubleSpinBoxItemDelegate** (QObject ∗parent=nullptr)
- QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const override

  *Create a double value editor.*
- void **setEditorData** (QWidget ∗pEditor, const QModelIndex &index) const override

  *Specify data to show.*
- void **setModelData** (QWidget ∗pEditor, QAbstractItemModel ∗pModel, const QModelIndex &index) const override

  *Set data to a model.*
- void **updateEditorGeometry** (QWidget ∗pEditor, const QStyleOptionViewItem &option, const QModelIndex &index) const override

  *Set a geometry to render.*

### 4.7.1 Detailed Description

Class to specify how table values can be edited.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/doublespinboxitemdelegate.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/doublespinboxitemdelegate.cpp

## 4.8 KLP::EnergyFrame Struct Reference

Energy quantities associated with a frame.

```
#include <framecollection.h>
```

## Public Attributes

- FloatFrameObject **kinetic**
- FloatFrameObject **potential**
- FloatFrameObject **full**

### 4.8.1 Detailed Description

Energy quantities associated with a frame.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/framecollection.h

## 4.9 KLP::FrameCollection Struct Reference

Set of all quantities belonged to a frame.

```
#include <framecollection.h>
```

### Public Attributes

- int **numRods**

  *Number of rods.*
- FloatFrameObject **parameter**

  *Parameter.*
- FloatFrameObject **naturalLength**

  *Natural length.*
- FloatFrameObject **accumulatedNaturalLength**
- FloatFrameObject **coordinates** [kNumDirections]

  *Coordinates.*
- StateFrame **state**

  *Regular state.*
- StateFrame **projectedState**

  *Projected regular state.*
- StateFrame **firstDerivativeState**

  *First-order derivate of the state with respect to time.*
- StateFrame **secondDerivativeState**

  *Second-order derivate of the state with respect to time.*
- StateFrame **errorState**

  *State error.*
- FloatFrameObject **strain**

  *Strain.*
- std::vector< StateFrame > **modalStates**

  *Set of modal states.*
- FloatFrameObject **frequencies**

  *Frequencies.*
- EnergyFrame **energy**

  *Energy.*

### 4.9.1 Detailed Description

Set of all quantities belonged to a frame.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/framecollection.h

## 4.10 KLP::FrameObject< T > Class Template Reference

### Public Types

- using **iterator** = FrameObjectIterator< T >

### Public Member Functions

- **FrameObject** (T const ∗pData=nullptr, T normFactor=1.0, qint64 size=0, qint64 step=1)
- bool **isEmpty** () const
- qint64 **size** () const
- iterator **begin** ()
- iterator **end** ()
- iterator **operator[ ]** (int index)

### Private Attributes

- T const ∗ **mpData**
- T **mNormFactor**
- qint64 **mSize**
- qint64 **mStep**

### Friends

- template<typename K >
  QDebug **operator<<** (QDebug stream, FrameObject< K > &frameObject)

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/frameobject.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/frameobject.cpp

## 4.11 KLP::FrameObjectIterator< T > Class Template Reference

Class to iterate through data of a record.

```
#include <frameobjectiterator.h>
```

### Public Types

- using **self_type** = FrameObjectIterator< T >
- using **iterator_category** = std::random_access_iterator_tag
- using **difference_type** = std::ptrdiff_t
- using **value_type** = T
- using **pointer** = T const ∗
- using **reference** = T const &

### Public Member Functions

- **FrameObjectIterator** (pointer pData, T normFactor, qint64 step)
- value_type **operator∗** ()
- self_type & **operator++** ()
- self_type **operator++** (int)
- self_type **operator+** (const difference_type &movement)
- difference_type **operator-** (const FrameObjectIterator &another) const

### Private Attributes

- pointer **mpData**
- T **mNormFactor**
- qint64 const **mStep**

### Friends

- bool **operator==** (self_type const &first, self_type const &second)
- bool **operator!=** (self_type const &first, self_type const &second)

### 4.11.1 Detailed Description

**template<typename T>**
**class KLP::FrameObjectIterator< T >**

Class to iterate through data of a record.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/frameobjectiterator.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/frameobjectiterator.cpp

## 4.12 KLP::Index Struct Reference

Structure to navigate through records.

```
#include <index.h>
```

### Public Member Functions

- **Index** ()
  *Base constructor.*

### Public Attributes

- std::vector< IndexData > **data**
  *Data.*
- quint64 **recordShift** = 0
  *Shift of the main record.*
- quint64 **relativeDataShift** = 0
  *Relative shift of data.*

### 4.12.1 Detailed Description

Structure to navigate through records.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/index.h

## 4.13 KLP::IndexData Struct Reference

Data of each record.

```
#include <index.h>
```

### Public Attributes

- qint64 **position** = 0
  *Position of a record in the buffer.*
- qint64 **size** = 0
  *Size of a record.*
- qint64 **step** = 1
  *Step for iterating inside a record.*
- qint64 **partSize** = 0
  *Partial length of a quantity inside a record.*

### 4.13.1 Detailed Description

Data of each record.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/index.h

## 4.14 RSE::Core::IO Class Reference

Class to save the project and solution data.

```
#include <io.h>
```

### Public Member Functions

- **IO** (QString const &lastPath)
- QString const & **lastPath** () const
- QString const & **extension** () const
- void **saveAs** (QString const &pathFile, Project &project, Solution::SolutionOptions &options)
  *Save the project and solution data to a file.*
- IOPair **open** (QString const &pathFile, DataBaseCables const &dataBaseCables)
  *Read the computational data from a file.*

## Private Attributes

- const QString **mkProjectExtension** = ".rse"
- QString **mLastPath**

### 4.14.1 Detailed Description

Class to save the project and solution data.

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

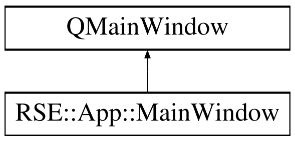
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/io.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/io.cpp

## 4.15 RSE::App::MainWindow Class Reference

Central window of the program.

```
#include <mainwindow.h>
```

Inheritance diagram for RSE::App::MainWindow:



## Public Member Functions

- **MainWindow** (QWidget ∗parent=nullptr)

## Private Slots

- void **saveSettings** ()
  *Save current graphical settings of floating widgets.*
- void **restoreSettings** ()
  *Read graphical settings from a file.*
- void **setMassCable** (double)
  *Set mass of a cable.*
- void **setMassLoadedCable** (double)
  *Set mass of a cable with ice on it.*
- void **setWorkingLength** (double)
  *Specify working length.*
- void **setBouncerLength** (double)
  *Set length of a bouncer.*
- void **setSpringLength** (double)
  *Assingn length of a spring.*

- void **setSpringStiffness** (double)

  *Set spring stiffness of a damper.*
- void **computeSpring** ()

  *Compute parameters of a spring.*
- void **setCable** (QString const &name)

  *Assign cables to a rod system.*
- void **setForce** (double)

  *Specify stretching force.*
- void **computeSpans** ()

  *Compute length of all cables.*
- void **setLongitudinalStiffness** (double)

  *Specify longitudinal stiffness of all supports.*
- void **setVerticalStiffness** (double)

  *Specify vertical stiffness of all supports.*
- void **runRodSystemSolution** ()

  *Solve the rod system.*
- void **runOptimizationSolution** ()

  *Optimize viscosities of dampers.*
- void **appendOutputData** (QByteArray)

  *Process the message from the solution process.*
- void **showConvergence** ()

  *Represent the convergence of the optimization process.*
- void **createProject** ()

  *Open a new project.*
- void **saveAsProject** ()

  *Save the project using a dialog window.*
- void **saveProject** ()

  *Save the current project.*
- void **openProjectDialog** ()

  *Open a project by means of a dialog window.*
- void **openProject** (QString const &)

  *Open a project using a path specified.*
- void **setProjectTitle** ()

  *Set the name of a project.*
- void **setProjectData** ()

  *Set project data.*
- void **setSolutionOptions** ()

  *Set the data to be used as the solution parameters.*
- void **setCurrentCable** ()

  *Select a current cable.*
- void **setBlockedSignals** (bool)

  *(Un)Block all the signals from widget*
- void **aboutProgram** ()

  *Show the information about the program.*

## Private Member Functions

- void **initializeWindow** ()

  *Set the state and geometry of the central window.*
- void **createContent** ()

  *Create all the widgets and links between them.*
- void **createDefaultProject** ()

  *Create a default project.*
- void **createDefaultSolutionOptions** ()

  *Create default solution options.*
- void **closeEvent** (QCloseEvent ∗pEvent) override

  *Save settings and parameters of project while closing the central window.*
- ads::CDockWidget ∗ **createDamperWidget** ()

  *Create a widget to specify data of a damper.*
- ads::CDockWidget ∗ **createRodSystemWidget** ()

  *Create a widget to set and control data of a rod system.*
- ads::CDockWidget ∗ **createSupportWidget** ()

  *Create a widget to specify data of supports.*
- ads::CDockWidget ∗ **createCalculationWidget** ()

  *Create a widget to control the solution process.*
- ads::CDockWidget ∗ **createConsole** ()

  *Construct a widget to view solution information.*
- void **specifyMenuConnections** ()

  *Specify menu interactions.*

## Private Attributes

- Ui::MainWindow ∗ **mpUi**
- ads::CDockManager ∗ **mpDockManager**
- Models::RodSystemTableModel ∗ **mpRodSystemTableModel**
- QDoubleSpinBox ∗ **mpMassCable**
- QDoubleSpinBox ∗ **mpMassLoadedCable**
- QDoubleSpinBox ∗ **mpWorkingLength**
- QDoubleSpinBox ∗ **mpBouncerLength**
- QDoubleSpinBox ∗ **mpSpringLength**
- QDoubleSpinBox ∗ **mpSpringStiffness**
- QComboBox ∗ **mpNameCable**
- QDoubleSpinBox ∗ **mpForce**
- QDoubleSpinBox ∗ **mpLongitudinalStiffness**
- QDoubleSpinBox ∗ **mpVerticalStiffness**
- QSpinBox ∗ **mpNumCalcModes**
- QSpinBox ∗ **mpNumDampModes**
- QSpinBox ∗ **mpStepModes**
- QDoubleSpinBox ∗ **mpTolTrunc**
- QTextEdit ∗ **mpConsole**
- RSE::Core::Project ∗ **mpProject**
- RSE::Solution::SolutionManager ∗ **mpSolutionManager**
- RSE::Solution::SolutionOptions ∗ **mpSolutionOptions**
- RSE::Core::IO ∗ **mpIO**
- QSharedPointer< QSettings > **mpSettings**

### 4.15.1 Detailed Description

Central window of the program.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/mainwindow.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/mainwindow.cpp

## 4.16 RSE::Core::Project Class Reference

### Public Member Functions

- **Project** (QString const &name, DataBaseCables dataBaseCables, Damper damper, RodSystem rodSystem, Support support)
- QString const & **name** () const
- void **setName** (QString const &name)
- Damper & **damper** ()
- RodSystem & **rodSystem** ()
- Support & **support** ()
- DataBaseCables const & **dataBaseCables** () const
- void **readTemplateData** (QString const &path)

  *Read template data.*
- void **writeCalcData** (QString const &path, Solution::SolutionOptions const &options)

  *Write the computational data.*

### Private Member Functions

- AbstractDataObject ∗ **addDataObject** (AbstractDataObject::ObjectType type)

  *Create a data object with the specified type.*
- void **importDataObjects** (QString const &path, QString const &fileName)

  *Import several data objects from a file.*
- void **readProjectID** (QString const &path)

  *Read the identifier of a project.*
- void **modifyScalarDataObjects** ()

  *Modify scalar data objects.*
- void **modifyVectorDataObjects** (Spans const &spans)

  *Modify vector data objects.*
- void **writeDataObjects** (DataObjects const &dataObjects, QString const &path, QString const &fileName)

  *Write data objects to a file.*
- void **writeRods** (QString const &path, QString const &fileName)

  *Write data of rods.*
- void **writeProgram** (QString const &path, QString const &fileName, int numRods, int numCalcModes)

  *Write data of a program.*

### Private Attributes

- QString **mName**

    *Name of a project.*
- Damper **mDamper**

    *Parameters of a damper.*
- RodSystem **mRodSystem**

    *Parameters of a rod system.*
- Support **mSupport**

    *Parameters of supports.*
- DataBaseCables **mDataBaseCables**

    *Database of cables.*
- DataObjects **mScalarDataObjects**

    *Data objects.*
- DataObjects **mVectorDataObjects**
- int **mProjectID**

    *Project identifier.*
- QStringList **mRods**

    *Content of the file named RODS.*
- QStringList **mProgram**

    *Content of the file name PROG.*

### Static Private Attributes

- static const QString **skProjectExtension**

    *Project extension.*

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/project.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/project.cpp

## 4.17 KLP::Result Class Reference

Class to aggregate all the records.

```
#include <result.h>
```

### Public Member Functions

- **Result** (QString const &pathFile)
- bool **isEmpty** () const
- int **numRods** (qint64 iFrame) const

    *Get the number of rods associated with the requested frame.*
- FloatFrameObject **getFrameObject** (qint64 iFrame, RecordType type, float normFactor=1.0f, qint64 shift=0) const

    *Get the object associated with the requested frame.*
- FrameCollection **getFrameCollection** (qint64 iFrame) const

    *Retrieve the collection of the frame objects.*
- void **update** ()

    *Retrieve the updated content from the file.*

### Private Member Functions

- bool **read** ()

    *Read all the content of the file.*
- void **buildIndex** ()

    *Construct an object to navigate through records.*
- void **setStateFrameData** (StateFrame &state, RecordType type, qint64 iFrame, qint64 iStartData, std↩ ::vector< float > const &normFactors) const

    *Specify state data for each direction.*

### Private Attributes

- QString const **mkPathFile**

    *Path to the KLP file.*
- QByteArray **mContent**

    *Content of the file.*
- std::vector< Index > **mIndex**

    *Index of the data buffer.*
- qint64 **mNumRecords**

    *Number of records.*
- std::vector< float > **mTime**

    *Time array.*
- char **mNumBytesRod**

    *Number of bytes per rod.*

### 4.17.1 Detailed Description

Class to aggregate all the records.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/result.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/result.cpp

## 4.18 RSE::Core::RodSystem Class Reference

### Public Member Functions

- **RodSystem** (std::vector< double > distances, Cable const &cable, double force)
- std::vector< double > const & **distances** () const
- std::string const & **nameCable** () const
- double **force** () const
- int **numRods** () const
- double **massPerLength** () const
- void **setDistances** (std::vector< double > const &distances)

    *Specify distances between supports.*
- void **setCable** (Cable const &cable)

    *Modify the cable used in the rod system.*
- void **setForce** (double force)
- Spans **computeSpans** ()

    *Compute characteristics of spans.*

## Private Attributes

- RodSystemParameters **mParameters**
- std::string **mNameCable**

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/rodsystem.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/rodsystem.cpp

## 4.19 RSE::Core::RodSystemParameters Struct Reference

Parameters of a rod system.

```
#include <rodsystem.h>
```

### Public Attributes

- std::vector< double > **distances**

  *Distance between supports, m.*
- double **massPerLength**

  *Mass per length, kg.*
- double **youngsModulus**

  *Youngs modulus, Pa.*
- double **area**

  *Area of a cross-section, m^2.*
- double **force**

  *Stretching force, N.*
- int **numRods** = 0

  *Number of rods.*

### 4.19.1 Detailed Description

Parameters of a rod system.

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

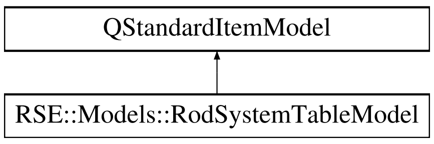
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/rodsystem.h

## 4.20 RSE::Models::RodSystemTableModel Class Reference

Table model to set and represent data of a rod system.

```
#include <rodsystemtablemodel.h>
```

Inheritance diagram for RSE::Models::RodSystemTableModel:



## Signals

- void **modified** ()

## Public Member Functions

- **RodSystemTableModel** (QObject ∗parent=nullptr)
- void **setRodSystem** (Core::RodSystem ∗pRodSystem)

  *Acquire the pointer to a rod system.*
- void **updateContent** ()

  *Represent all data of a rod system.*
- void **insertAfterSelected** ()

  *Insert fresh rows after selected ones.*
- void **removeSelected** ()

  *Remove the selected rows.*

## Private Member Functions

- void **clearContent** ()

  *Remove all the objects created.*
- void **setChangedData** (QStandardItem ∗pItem)

  *Set the changed distances between supports.*

## Private Attributes

- Core::RodSystem ∗ **mpRodSystem** = nullptr

### 4.20.1 Detailed Description

Table model to set and represent data of a rod system.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/rodsystemtablemodel.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/central/rodsystemtablemodel.cpp

## 4.21 RSE::Core::Array< T >::Row< U > Class Template Reference

Proxy class to acquire a row by index.

## Public Member Functions

- **Row** (T ∗pData)
- T & **operator[ ]** (IndexType iCol)
- T const & **operator[ ]** (IndexType iCol) const
- T ∗ **data** ()

### Private Attributes

- T ∗ **mpRow**

### 4.21.1 Detailed Description

**template<typename T>**
**template<typename U>**
**class RSE::Core::Array< T >::Row< U >**

Proxy class to acquire a row by index.

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

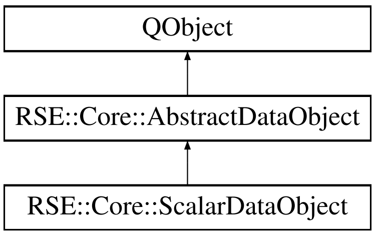
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/array.h

## 4.22 RSE::Core::ScalarDataObject Class Reference

Scalar data object.

```
#include <scalardataobject.h>
```

Inheritance diagram for RSE::Core::ScalarDataObject:



### Public Member Functions

- **ScalarDataObject** (QString const &name)
    *Construct a scalar data object.*
- ∼**ScalarDataObject** ()
    *Decrease a number of instances while being destroyed.*
- AbstractDataObject ∗ clone () const override
    *Clone a scalar data object.*
- DataItemType & addItem (DataValueType key) override
    *Insert a new item into ScalarDataObject.*
- virtual void import (QTextStream &stream) override
    *Import a scalar data object from a file.*

### Static Public Member Functions

- static quint32 **numberInstances** ()

## Static Private Attributes

- static quint32 **smNumInstances** = 0

## Additional Inherited Members

## 4.22.1 Detailed Description

Scalar data object.

## 4.22.2 Member Function Documentation

### 4.22.2.1 addItem()

```
DataItemType & ScalarDataObject::addItem (
            DataValueType key )  [override], [virtual]
```

Insert a new item into ScalarDataObject.

Implements RSE::Core::AbstractDataObject.

### 4.22.2.2 clone()

```
AbstractDataObject * ScalarDataObject::clone ( ) const  [override], [virtual]
```

Clone a scalar data object.

Implements RSE::Core::AbstractDataObject.

### 4.22.2.3 import()

```
void ScalarDataObject::import (
            QTextStream & stream )  [override], [virtual]
```

Import a scalar data object from a file.

Implements RSE::Core::AbstractDataObject.

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

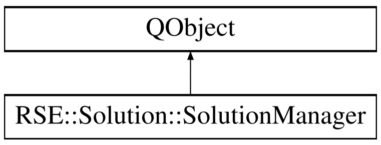
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/scalardataobject.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/scalardataobject.cpp

## 4.23 RSE::Solution::SolutionManager Class Reference

Class to control the solution process.

```
#include <solutionmanager.h>
```

Inheritance diagram for RSE::Solution::SolutionManager:



### Public Slots

- void **stopSolution** ()

  *Stop the solution process.*

### Signals

- void **outputSent** (QByteArray)
- void **rodSystemSolved** ()
- void **optimizationSolved** ()
- void **optimizationStepPerformed** ()

### Public Member Functions

- **SolutionManager** (QString const &rootPath, QString const &relativeInputPath, QString const &relative↩ OutputPath)
- void **solveRodSystem** (Core::Project &project, SolutionOptions const &options)

  *Solve a rod system.*
- void **solveOptimization** (Core::Project &project, SolutionOptions const &options)

  *Optimize viscosities of dampers as to damp selected set of modes.*
- void **runVisualizer** ()

  *Run the visualizer of a rod system.*

### Private Member Functions

- void **processRodSystemStream** ()

  *Process the output of the rod system solver.*
- void **processOptimizationStream** ()

  *Process the optimization output.*
- void **runParserProcess** ()

  *Prepare data for the optimization process.*
- void **writeOptimizationInput** (QString const &pathFile, int numDampers, SolutionOptions const &options)

  *Write the input data for optimization of viscosities.*
- int **getRodSystemStatus** ()

  *Check if the solution process if finished.*

### Private Attributes

- QString **mRootPath**
- QString **mInputPath**
- QString **mOutputPath**
- QProcess $*$ **mpRodSystemSolver** = nullptr
- QProcess $*$ **mpOptimizationSolver** = nullptr

### 4.23.1 Detailed Description

Class to control the solution process.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/solutionmanager.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/solutionmanager.cpp

## 4.24 RSE::Solution::SolutionOptions Class Reference

### Public Member Functions

- **SolutionOptions** (int numCalcModes, int numDampModes, int stepModes, double tolTrunc)
- int **numCalcModes** () const
- int **numDampModes** () const
- int **stepModes** () const
- double **tolTrunc** () const
- void **setNumCalcModes** (int numCalcModes)
- void **setNumDampModes** (int numDampModes)
- void **setStepModes** (int stepModes)
- void **setTolTrunc** (double tolTrunc)

### Private Attributes

- int **mNumCalcModes**

  *Number of computational modes.*
- int **mNumDampModes**

  *Number of modes to be damped.*
- int **mStepModes**

  *Step through computational modes.*
- double **mTolTrunc**

  *Limit to truncate computational modes.*

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/solutionoptions.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/solutionoptions.cpp

## 4.25 RSE::Core::Spans Struct Reference

Computed parameters of spans.

```
#include <rodsystem.h>
```

### Public Member Functions

- **Spans** (int numRods)

### Public Attributes

- std::vector< double > **u0**

  *Constant at the left end.*
- std::vector< double > **uL**

  *Constant at the right end.*
- std::vector< double > **L**

  *Length of a rod, m.*
- double **projectedForce**

  *Projected stretching force, N.*

### 4.25.1 Detailed Description

Computed parameters of spans.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/rodsystem.h

## 4.26 KLP::StateFrame Struct Reference

Kinematic and dynamic quantities associated with a frame.

```
#include <framecollection.h>
```

### Public Attributes

- FloatFrameObject **displacements** [kNumDirections]
- FloatFrameObject **rotations** [kNumDirections]
- FloatFrameObject **forces** [kNumDirections]
- FloatFrameObject **moments** [kNumDirections]

### 4.26.1 Detailed Description

Kinematic and dynamic quantities associated with a frame.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/klp/framecollection.h

## 4.27 RSE::Core::Support Class Reference

Class to aggregate data of supports.

`#include <support.h>`

### Public Member Functions

- **Support** (double longitudinalStiffness, double verticalStiffness)
- double **longitudinalStiffness** () const
- double **verticalStiffness** () const
- void **setLongitudinalStiffness** (double longitudinalStiffness)
- void **setVerticalStiffness** (double verticalStiffness)

### Private Attributes

- double **mLongitudinalStiffness**

  *Longitudinal stiffness (1), N/m.*
- double **mVerticalStiffness**

  *Vertical stiffness (2), N/m.*

### 4.27.1 Detailed Description

Class to aggregate data of supports.

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

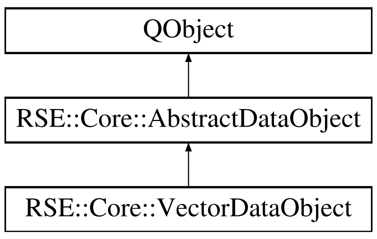
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/support.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/support.cpp

## 4.28 RSE::Core::VectorDataObject Class Reference

Vector data object.

`#include <vectordataobject.h>`

Inheritance diagram for RSE::Core::VectorDataObject:



## Public Member Functions

- **VectorDataObject** (QString const &name)

  *Construct a vector data object.*
- ∼**VectorDataObject** ()

  *Decrease a number of instances while being destroyed.*
- AbstractDataObject ∗ clone () const override

  *Clone a vector data object.*
- DataItemType & addItem (DataValueType key) override

  *Insert a new item into VectorDataObject.*
- virtual void import (QTextStream &stream) override

  *Import a vector data object from a file.*

## Static Public Member Functions

- static quint32 **numberInstances** ()

## Static Private Attributes

- static quint32 **smNumInstances** = 0

## Additional Inherited Members

### 4.28.1 Detailed Description

Vector data object.

### 4.28.2 Member Function Documentation

#### 4.28.2.1 addItem()

```
DataItemType & VectorDataObject::addItem (
            DataValueType key )  [override], [virtual]
```

Insert a new item into VectorDataObject.

Implements RSE::Core::AbstractDataObject.

#### 4.28.2.2 clone()

```
AbstractDataObject * VectorDataObject::clone ( ) const  [override], [virtual]
```

Clone a vector data object.

Implements RSE::Core::AbstractDataObject.

#### 4.28.2.3 import()

```
void VectorDataObject::import (
             QTextStream & stream )  [override], [virtual]
```

Import a vector data object from a file.

Implements RSE::Core::AbstractDataObject.

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

- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/vectordataobject.h
- /home/qinterfly/Library/Projects/Current/RodSystemEstimator/src/core/vectordataobject.cpp

# Chapter 5

# File Documentation

## 5.1 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/central/doublespinboxitemdelegate.cpp File Reference

DoubleSpinBoxItemDelegate.

```
#include <QDoubleSpinBox>
#include "doublespinboxitemdelegate.h"
```

### 5.1.1 Detailed Description

DoubleSpinBoxItemDelegate.

**Author**

**Date**

2022

## 5.2 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/central/doublespinboxitemdelegate.h File Reference

DoubleSpinBoxItemDelegate.

```
#include <QStyledItemDelegate>
```

### Classes

- class RSE::Models::DoubleSpinBoxItemDelegate

  *Class to specify how table values can be edited.*

### 5.2.1 Detailed Description

DoubleSpinBoxItemDelegate.

**Author**

**Date**

2022

## 5.3 doublespinboxitemdelegate.h

Go to the documentation of this file.

```

#ifndef DOUBLESPINBOXITEMDELEGATE_H
#define DOUBLESPINBOXITEMDELEGATE_H

#include <QStyledItemDelegate>

namespace RSE::Models
{

class DoubleSpinBoxItemDelegate : public QStyledItemDelegate
{
public:
    DoubleSpinBoxItemDelegate(QObject* parent = nullptr);
    QWidget* createEditor(QWidget* parent, const QStyleOptionViewItem& option, const QModelIndex& index)
    const override;
    void setEditorData(QWidget* pEditor, const QModelIndex& index) const override;
    void setModelData(QWidget* pEditor, QAbstractItemModel* pModel, const QModelIndex& index) const
    override;
    void updateEditorGeometry(QWidget* pEditor, const QStyleOptionViewItem& option, const QModelIndex&
    index) const override;
};

}

#endif // DOUBLESPINBOXITEMDELEGATE_H
```

## 5.4 /home/qinterfly/Library/Projects/Current/RodSystem←Estimator/src/central/mainwindow.cpp File Reference

Definition of the MainWindow class.

```
#include <QVBoxLayout>
#include <QGridLayout>
#include <QLabel>
#include <QTextEdit>
#include <QDoubleSpinBox>
#include <QSpinBox>
#include <QSpacerItem>
#include <QSettings>
#include <QTableView>
#include <QHeaderView>
#include <QToolBar>
#include <QComboBox>
#include <QFileDialog>
#include <QMessageBox>
```

```
#include "DockManager.h"
#include "DockWidget.h"
#include "DockAreaWidget.h"
#include "ads_globals.h"
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "uiconstants.h"
#include "rodsystemtablemodel.h"
#include "doublespinboxitemdelegate.h"
#include "core/project.h"
#include "core/solutionoptions.h"
#include "core/solutionmanager.h"
#include "core/io.h"
#include "viewers/convergenceviewer.h"
```

### Functions

- QDoubleSpinBox ∗ **createDoubleField** (double value, double maxValue=1e3, int numDecimals=3)
  *Create a field to input a floating-point number.*
- QSpinBox ∗ **createIntegerField** (int value, int maxValue=1000)
  *Create a field to input an integer.*

### 5.4.1 Detailed Description

Definition of the MainWindow class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.5 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/central/mainwindow.h File Reference

Declaration of the MainWindow class.

```
#include <QMainWindow>
```

### Classes

- class RSE::App::MainWindow
  *Central window of the program.*

### 5.5.1 Detailed Description

Declaration of the MainWindow class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.6 mainwindow.h

Go to the documentation of this file.

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui
{
class MainWindow;
}
class QSettings;
class QDoubleSpinBox;
class QSpinBox;
class QTableView;
class QTextEdit;
class QProcess;
class QComboBox;
QT_END_NAMESPACE

namespace ads
{
class CDockManager;
class CDockWidget;
}

namespace RSE
{

namespace Core
{
class Project;
class IO;
}

namespace Solution
{
class SolutionManager;
class SolutionOptions;
}

namespace Models
{
class RodSystemTableModel;
}

namespace App
{

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget* parent = nullptr);
    ~MainWindow();

private:
    // Content
    void initializeWindow();
    void createContent();
    void createDefaultProject();
    void createDefaultSolutionOptions();
    void closeEvent(QCloseEvent* pEvent) override;
    ads::CDockWidget* createDamperWidget();
    ads::CDockWidget* createRodSystemWidget();
    ads::CDockWidget* createSupportWidget();
    ads::CDockWidget* createCalculationWidget();
    ads::CDockWidget* createConsole();
    // Signals & Slots
    void specifyMenuConnections();

private slots:
    // Settings
    void saveSettings();
    void restoreSettings();
    // Parameters of a damper
    void setMassCable(double);
    void setMassLoadedCable(double);
    void setWorkingLength(double);
    void setBouncerLength(double);
    void setSpringLength(double);
    void setSpringStiffness(double);
    void computeSpring();
    // Parameters of a rod system
    void setCable(QString const& name);
    void setForce(double);
    void computeSpans();
    // Parameters of supports
    void setLongitudinalStiffness(double);
    void setVerticalStiffness(double);
    // Controlling the solution process
    void runRodSystemSolution();
    void runOptimizationSolution();
    void appendOutputData(QByteArray);
    void showConvergence();
    // Creating a project
    void createProject();
    // Saving a project
    void saveAsProject();
    void saveProject();
    // Open a project
    void openProjectDialog();
    void openProject(QString const&);
    void setProjectTitle();
    // Set project data
    void setProjectData();
    void setSolutionOptions();
    void setCurrentCable();
    void setBlockedSignals(bool);
    void aboutProgram();

private:
    // GUI
    Ui::MainWindow* mpUi;
    ads::CDockManager* mpDockManager;
    Models::RodSystemTableModel* mpRodSystemTableModel;
    // Parameters of a damper
    QDoubleSpinBox* mpMassCable;
    QDoubleSpinBox* mpMassLoadedCable;
    QDoubleSpinBox* mpWorkingLength;
    QDoubleSpinBox* mpBouncerLength;
    QDoubleSpinBox* mpSpringLength;
    QDoubleSpinBox* mpSpringStiffness;
    // Parameters of a rod system
    QComboBox* mpNameCable;
    QDoubleSpinBox* mpForce;
    // Parameters of supports
    QDoubleSpinBox* mpLongitudinalStiffness;
    QDoubleSpinBox* mpVerticalStiffness;
    // Options of computational process
    QSpinBox* mpNumCalcModes;
    QSpinBox* mpNumDampModes;
    QSpinBox* mpStepModes;
    QDoubleSpinBox* mpTolTrunc;
    QTextEdit* mpConsole;
    // Project
    RSE::Core::Project* mpProject;
    RSE::Solution::SolutionManager* mpSolutionManager;
    RSE::Solution::SolutionOptions* mpSolutionOptions;
    RSE::Core::IO* mpIO;
    // Settings
    QSharedPointer<QSettings> mpSettings;
};

}

```

```
}

#endif // MAINWINDOW_H
```

## 5.7 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/central/rodsystemtablemodel.cpp File Reference

Definition of the RodSystemTableModel class.

```
#include <QTableView>
#include "rodsystemtablemodel.h"
#include "core/rodsystem.h"
```

### 5.7.1 Detailed Description

Definition of the RodSystemTableModel class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.8 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/central/rodsystemtablemodel.h File Reference

Declaration of the RodSystemTableModel class.

```
#include <QStandardItemModel>
```

### Classes

- class RSE::Models::RodSystemTableModel

  *Table model to set and represent data of a rod system.*

### 5.8.1 Detailed Description

Declaration of the RodSystemTableModel class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.9 rodsystemtablemodel.h

Go to the documentation of this file.

```
#ifndef RODSYSTEMTABLEMODEL_H
#define RODSYSTEMTABLEMODEL_H

#include <QStandardItemModel>

namespace RSE
{

namespace Core
{
class RodSystem;
}

namespace Models
{

class RodSystemTableModel : public QStandardItemModel
{
    Q_OBJECT

public:
    RodSystemTableModel(QObject* parent = nullptr);
    ~RodSystemTableModel() = default;
    void setRodSystem(Core::RodSystem* pRodSystem);
    void updateContent();
    void insertAfterSelected();
    void removeSelected();

signals:
    void modified();

private:
    void clearContent();
    void setChangedData(QStandardItem* pItem);

private:
    Core::RodSystem* mpRodSystem = nullptr;
};

}

}



#endif // RODSYSTEMTABLEMODEL_H
```

## 5.10 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/central/uiconstants.h File Reference

Graphical constants shared between several widgets.

```
#include <QString>
```

### Variables

- const QString **RSE::UiConstants::Settings::skGeometry** = "geometry"
- const QString **RSE::UiConstants::Settings::skState** = "state"
- const QString **RSE::UiConstants::Settings::skDockingState** = "dockingState"

### 5.10.1 Detailed Description

Graphical constants shared between several widgets.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.11 uiconstants.h

Go to the documentation of this file.

```
#ifndef UICONSTANTS_H
#define UICONSTANTS_H

#include <QString>

namespace RSE::UiConstants
{

namespace Settings
{
const QString skGeometry     = "geometry";
const QString skState        = "state";
const QString skDockingState = "dockingState";
}

}

#endif // UICONSTANTS_H
```

## 5.12 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/abstractdataobject.cpp File Reference

Implementation of the AbstractDataObject class.

```
#include "abstractdataobject.h"
#include "constants.h"
```

### 5.12.1 Detailed Description

Implementation of the AbstractDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.13 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/abstractdataobject.h File Reference

Declaration of the AbstractDataObject class.

```
#include <QObject>
#include <QString>
#include <QDataStream>
#include <map>
#include "array.h"
#include "aliasdata.h"
```

### Classes

- class RSE::Core::AbstractDataObject

    *Data object which is designied in the way to be represented in a table easily.*

### Typedefs

- using **RSE::Core::DataItemType** = Array< DataValueType >
- using **RSE::Core::DataHolder** = std::multimap< DataKeyType, DataItemType >

### Functions

- QDataStream & **RSE::Core::operator**<< (QDataStream &stream, AbstractDataObject const &obj)

    *Print a data object to a binary stream.*

### 5.13.1 Detailed Description

Declaration of the AbstractDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.14 abstractdataobject.h

Go to the documentation of this file.

```
#ifndef ABSTRACTDATAOBJECT_H
#define ABSTRACTDATAOBJECT_H

#include <QObject>
#include <QString>
#include <QDataStream>
#include <map>
#include "array.h"
#include "aliasdata.h"

namespace RSE::Core
{

using DataItemType = Array<DataValueType>;
using DataHolder = std::multimap<DataKeyType, DataItemType>;

class AbstractDataObject : public QObject
{
public:
    enum ObjectType
    {
        kScalar,
        kVector,
        kMatrix,
        kSurface
    };
    AbstractDataObject(ObjectType type, QString const& name);
    virtual ~AbstractDataObject() = 0;
    virtual AbstractDataObject* clone() const = 0;
    virtual DataItemType& addItem(DataKeyType key) = 0;
    void removeItem(DataValueType key);
    bool changeItemKey(DataKeyType oldKey, DataKeyType newKey, DataHolder* items = nullptr);
    bool setArrayValue(DataKeyType key, DataValueType newValue, IndexType iRow = 0, IndexType iColumn =
    0);
    DataValueType arrayValue(DataKeyType key, IndexType iRow = 0, IndexType iColumn = 0);
    std::vector<DataKeyType> keys() const;
    quint32 numberItems() const { return mItems.size(); }
    DataHolder const& getItems() { return mItems; }
    DataIDType id() const { return mID; }
    ObjectType type() const { return mkType; }
    QString const& name() const { return mName; }
    void setName(QString const& name) { mName = name; }
    static DataIDType maxObjectID() { return smMaxObjectID; }
    static void setMaxObjectID(DataIDType iMaxObjectID) { smMaxObjectID = iMaxObjectID; }
    virtual void serialize(QDataStream& stream) const;
    virtual void deserialize(QDataStream& stream);
    friend QDataStream& operator«(QDataStream& stream, AbstractDataObject const& obj);
    virtual void import(QTextStream& stream) = 0;
    void write(QTextStream& stream) const;

protected:
    const ObjectType mkType;
    QString mName;
    DataIDType mID;
    DataHolder mItems;

private:
    static DataIDType smMaxObjectID;
};

inline QDataStream& operator«(QDataStream& stream, AbstractDataObject const& obj)
{
    obj.serialize(stream);
    return stream;
}

}

#endif // ABSTRACTDATAOBJECT_H
```

## 5.15 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/aliasdata.h File Reference

Specification of data types used in a project.

```
#include <QtGlobal>
```

### Typedefs

- using **RSE::Core::DataValueType** = double
- using **RSE::Core::DataKeyType** = double
- using **RSE::Core::DataIDType** = qint64

### 5.15.1 Detailed Description

Specification of data types used in a project.

**Author**

Pavel Lakiza

**Date**

May 2021

## 5.16 aliasdata.h

Go to the documentation of this file.

```

#ifndef ALIASDATA_H
#define ALIASDATA_H

#include <QtGlobal>

namespace RSE::Core
{

using DataValueType = double;
using DataKeyType = double;
using DataIDType = qint64;

}

#endif // ALIASDATA_H
```

## 5.17 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/array.cpp File Reference

Implementation of the Array class.

```
#include "array.h"
```

### 5.17.1 Detailed Description

Implementation of the Array class.

**Author**

Pavel Lakiza

**Date**

March 2021

## 5.18 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/core/array.h File Reference

Declaration of the Array class.

```
#include <QDebug>
#include "constants.h"
```

### Classes

- class RSE::Core::Array< T >

  *Numerical array class.*
- class RSE::Core::Array< T >::Row< U >

  *Proxy class to acquire a row by index.*

### Typedefs

- using **RSE::Core::IndexType** = quint32

### Functions

- template<typename K >
  QDebug **RSE::Core::operator**<< (QDebug stream, Array< K > &array)

  *Print all array values using the matrix format.*
- template<typename K >
  QDataStream & **RSE::Core::operator**<< (QDataStream &stream, Array< K > const &array)

  *Write an array to a binary stream.*
- template<typename K >
  QDataStream & **RSE::Core::operator**>> (QDataStream &stream, Array< K > &array)

  *Read an array from a stream.*
- template<typename K >
  QTextStream & **RSE::Core::operator**<< (QTextStream &stream, Array< K > const &array)

  *Write an array to a text stream.*

### 5.18.1 Detailed Description

Declaration of the Array class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.19 array.h

Go to the documentation of this file.

```

#ifndef ARRAY_H
#define ARRAY_H

#include <QDebug>
#include "constants.h"

namespace RSE::Core
{

using IndexType = quint32;

template<typename T>
class Array
{
private:
    template <typename U> class Row;

public:
    Array(IndexType numRows = 0, IndexType numCols = 0);
    Array(Array<T> const& another);
    Array(Array<T>&& another);
    ~Array();
    T* data() { return mpData; }
    void resize(IndexType numRows, IndexType numCols);
    void removeColumn(IndexType iRemoveColumn);
    void swapColumns(IndexType iFirstColumn, IndexType iSecondColumn);
    void clear();
    IndexType rows() const { return mNumRows; };
    IndexType cols() const { return mNumCols; };
    IndexType size() const { return mNumRows * mNumCols; }
    Row<T> operator[](IndexType iRow) { return Row<T>(&mpData[mNumCols * iRow]); };
    Row<T> operator[](IndexType iRow) const { return Row<T>(&mpData[mNumCols * iRow]); };
    Array& operator=(Array<T> const& another);
    template<typename K> friend QDebug operator«(QDebug stream, Array<K>& array);
    template<typename K> friend QDataStream& operator«(QDataStream& stream, Array<K> const& array);
    template<typename K> friend QDataStream& operator»(QDataStream& stream, Array<K>& array);
    template<typename K> friend QTextStream& operator«(QTextStream& stream, Array<K> const& array);

private:
    IndexType mNumRows;
    IndexType mNumCols;
    T* mpData = nullptr;
    template <typename U>
    class Row
    {
    public:
        Row() = delete;
        Row(T* pData) : mpRow(pData) { };
        ~Row() { }
        T& operator[](IndexType iCol) { return mpRow[iCol]; }
        T const& operator[](IndexType iCol) const { return mpRow[iCol]; }
        T* data() { return mpRow; }
    private:
        T* mpRow;
    };
};

template<typename K>
inline QDebug operator«(QDebug stream, Array<K>& array)
{
    IndexType const& nRows = array.mNumRows;
    IndexType const& nCols = array.mNumCols;
    stream = stream.noquote();
    stream « QString("Array size: %1 x %2").arg(QString::number(nRows), QString::number(nCols));
    stream « Qt::endl;
    for (IndexType iRow = 0; iRow != nRows; ++iRow)
    {
        for (IndexType jCol = 0; jCol != nCols; ++jCol)
            stream « QString::number(array[iRow][jCol]);
        stream « Qt::endl;
    }
    return stream;
}

template<typename K>
inline QDataStream& operator«(QDataStream& stream, Array<K> const& array)
{
    stream « array.mNumRows « array.mNumCols;
    IndexType const& size = array.size();
    for (IndexType i = 0; i != size; ++i)
        stream « array.mpData[i];
    return stream;
}

template<typename K>
inline QDataStream& operator»(QDataStream& stream, Array<K>& array)
{
    delete[] array.mpData;
    stream » array.mNumRows » array.mNumCols;
    IndexType const& size = array.size();
    array.mpData = new K[size];
    for (IndexType i = 0; i != size; ++i)
        stream » array.mpData[i];
    return stream;
}

template<typename K>
inline QTextStream& operator«(QTextStream& stream, Array<K> const& array)
{
    IndexType const& numRows = array.mNumRows;
    IndexType const& numCols = array.mNumCols;
    for (IndexType iRow = 0; iRow != numRows; ++iRow)
    {
        for (IndexType jCol = 0; jCol != numCols; ++jCol)
            stream « QString::number(array[iRow][jCol], 'g', RSE::Constants::kWritingPrecision);
        stream « Qt::endl;
    }
    return stream;
}

}

#endif // ARRAY_H
```

## 5.20 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/constants.h File Reference

Computational constants.

### Variables

- const double **RSE::Constants::kGravitationalAcceleration** = 9.8067

    *Gravitational acceleration, m/s^2.*

- const int **RSE::Constants::kWritingPrecision** = 15

    *Number of digits to be written to a file.*

### 5.20.1 Detailed Description

Computational constants.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.21 constants.h

Go to the documentation of this file.

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

namespace RSE::Constants
{

const double kGravitationalAcceleration = 9.8067;

const int kWritingPrecision = 15;

}

#endif // CONSTANTS_H
```

## 5.22 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/damper.cpp File Reference

Definition of the Damper class.

```
#include "damper.h"
#include "constants.h"
```

### 5.22.1 Detailed Description

Definition of the Damper class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.23 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/core/damper.h File Reference

Declaration the Damper class.

```
#include <QPair>
```

### Classes

- class RSE::Core::Damper

    *Class to compute and collect properties of a damper.*

### 5.23.1 Detailed Description

Declaration the Damper class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.24 damper.h

Go to the documentation of this file.

```

#ifndef DAMPER_H
#define DAMPER_H

#include <QPair>

namespace RSE::Core
{

class Damper
{
public:
    Damper(double massCable, double massLoadedCable, double workingLength, double bouncerLength,
           double springLength = 0, double springStiffness = 0);
    ~Damper() = default;
    // Get parameteres of damper
    double massCable() const { return mMassCable; }
    double massLoadedCable() const { return mMassLoadedCable; }
    double workingLength() const { return mWorkingLength; }
    double bouncerLength() const { return mBouncerLength; }
    double springLength() const { return mSpringLength; }
    double springStiffness() const { return mSpringStiffness; }
    // Set parameters of a damper
    void setMassCable(double massCable) { mMassCable = massCable; }
    void setMassLoadedCable(double massLoadedCable) { mMassLoadedCable = massLoadedCable; }
    void setWorkingLength(double workingLength) { mWorkingLength = workingLength; }
    void setBouncerLength(double bouncerLength) { mBouncerLength = bouncerLength; }
    void setSpringLength(double springLength) { mSpringLength = springLength; }
    void setSpringStiffness(double springStiffness) { mSpringStiffness = springStiffness; }
    // Compute characteristics of a damper
    void computeSpring();

private:
    double mMassCable;
    double mMassLoadedCable;
    double mWorkingLength;
    double mBouncerLength;
    double mSpringLength = 0.0;
    double mSpringStiffness = 0.0;
};

}

#endif // DAMPER_H
```

## 5.25 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/databasecables.cpp File Reference

Definition of the DataBaseCables class.

```
#include <QFile>
#include <QTextStream>
#include "databasecables.h"
```

### 5.25.1 Detailed Description

Definition of the DataBaseCables class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.26 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/databasecables.h File Reference

Declaration of the DataBaseCables class.

```
#include <QString>
#include <unordered_map>
```

### Classes

- struct RSE::Core::Cable

  *Mechanical properties of a cable.*
- class RSE::Core::DataBaseCables

  *Aggregate data of cables.*

### 5.26.1 Detailed Description

Declaration of the DataBaseCables class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.27 databasecables.h

Go to the documentation of this file.

```
#ifndef DATACABLES_H
#define DATACABLES_H

#include <QString>
#include <unordered_map>

namespace RSE::Core
{

struct Cable
{
    std::string name;
    double bendingStiffness;
    double torsionalStiffness;
    double massPerLength;
    double youngsModulus;
    double area;
};

class DataBaseCables
{
public:
    DataBaseCables(QString const& directory, QString const& fileName);
    ~DataBaseCables() = default;
    std::vector<std::string> names() const;
    Cable const& getItem(std::string const& name) const { return mData.at(name); }

private:
    bool readDataBase(QString const& pathFile);

private:
    std::unordered_map<std::string, Cable> mData;
};

}

#endif // DATACABLES_H
```

## 5.28 /home/qinterfly/Library/Projects/Current/RodSystem↩Estimator/src/core/io.cpp File Reference

Definition of the IO class.

```
#include <QFile>
#include <QFileInfo>
#include <QDir>
#include "io.h"
#include "project.h"
```

### 5.28.1 Detailed Description

Definition of the IO class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.29 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/io.h File Reference

Declaration of the IO class.

```
#include <QString>
#include <QPair>
#include "solutionoptions.h"
```

### Classes

- class RSE::Core::IO

  *Class to save the project and solution data.*

### Typedefs

- using **RSE::Core::IOPair** = QPair< Project ∗, RSE::Solution::SolutionOptions ∗ >

### 5.29.1 Detailed Description

Declaration of the IO class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.30 io.h

Go to the documentation of this file.

```

#ifndef IO_H
#define IO_H

#include <QString>
#include <QPair>
#include "solutionoptions.h"

namespace RSE
{

namespace Core
{

class Project;
class DataBaseCables;

using IOPair = QPair<Project*, RSE::Solution::SolutionOptions*>;

class IO
{
public:
    IO(QString const& lastPath);
    ~IO() = default;
    QString const& lastPath() const { return mLastPath; }
    QString const& extension() const { return mkProjectExtension; }
    void saveAs(QString const& pathFile, Project& project, Solution::SolutionOptions& options);
    IOPair open(QString const& pathFile, DataBaseCables const& dataBaseCables);

private:
    const QString mkProjectExtension = ".rse";
    QString mLastPath;
};

}

}

#endif // IO_H
```

## 5.31 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/core/project.cpp File Reference

Definition of the Project class.

```
#include <QFile>
#include <QLocale>
#include "project.h"
#include "scalardataobject.h"
#include "vectordataobject.h"
#include "utilities.h"
#include "solutionoptions.h"
```

### Functions

- QStringList **readAllLines** (QString const &path, QString const &fileName)

  *Read all the lines from a file.*
- void **replaceStringEntry** (QString &string, int numSkipEntries, QString subString)

  *Replace a substring after specified number of skips.*
- void **writeAllLines** (QStringList const &lines, QString const &path, QString const &fileName)

  *Write all the lines to a file.*

### 5.31.1 Detailed Description

Definition of the Project class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.32 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/project.h File Reference

Declaration of the Project class.

```
#include <QString>
#include "abstractdataobject.h"
#include "damper.h"
#include "rodsystem.h"
#include "support.h"
#include "databasecables.h"
```

### Classes

- class RSE::Core::Project

### Typedefs

- using **RSE::Core::DataObjects** = std::vector< AbstractDataObject ∗ >

### 5.32.1 Detailed Description

Declaration of the Project class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.33 project.h

Go to the documentation of this file.

```

#ifndef PROJECT_H
#define PROJECT_H

#include <QString>
#include "abstractdataobject.h"
#include "damper.h"
#include "rodsystem.h"
#include "support.h"
#include "databasecables.h"

namespace RSE
{

namespace Solution
{
class SolutionOptions;
}

namespace Core
{
class ScalarDataObject;
class VectorDataObject;

using DataObjects = std::vector<AbstractDataObject*>;

class Project
{
public:
    Project(QString const& name, DataBaseCables dataBaseCables, Damper damper, RodSystem rodSystem,
    Support support);
    QString const& name() const { return mName; }
    void setName(QString const& name) { mName = name; }
    Damper& damper() { return mDamper; }
    RodSystem& rodSystem() { return mRodSystem; }
    Support& support() { return mSupport; }
    DataBaseCables const& dataBaseCables() const { return mDataBaseCables; }
    // IO
    void readTemplateData(QString const& path);
    void writeCalcData(QString const& path, Solution::SolutionOptions const& options);

private:
    AbstractDataObject* addDataObject(AbstractDataObject::ObjectType type);
    void importDataObjects(QString const& path, QString const& fileName);
    void readProjectID(QString const& path);
    // Modify data objects
    void modifyScalarDataObjects();
    void modifyVectorDataObjects(Spans const& spans);
    // IO
    void writeDataObjects(DataObjects const& dataObjects, QString const& path, QString const& fileName);
    void writeRods(QString const& path, QString const& fileName);
    void writeProgram(QString const& path, QString const& fileName, int numRods, int numCalcModes);

private:
    QString mName;
    Damper mDamper;
    RodSystem mRodSystem;
    Support mSupport;
    DataBaseCables mDataBaseCables;
    DataObjects mScalarDataObjects;
    DataObjects mVectorDataObjects;
    int mProjectID;
    QStringList mRods;
    QStringList mProgram;
    static const QString skProjectExtension;
};

}

}

#endif // PROJECT_H
```

## 5.34 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/core/rodsystem.cpp File Reference

Definition of the RodSystem class.

```
#include <functional>
#include <stdlib.h>
#include <stdio.h>
#include <gsl/gsl_multiroots.h>
#include "rodsystem.h"
#include "constants.h"
#include "databasecables.h"
```

### Typedefs

- using **IntegralFun** = std::function< double(double)>

## Functions

- double **integrate** (IntegralFun const &f, double const &a, double const &b, int const &n)

  *Compute integral using the MidPoint rule.*
- double **x1** (double u, double u0, double uL)
- double **x2** (double u, double u0, double uL)
- double **Q1** (double u, double u0, double uL)
- double **Q2** (double u, double u0, double uL)
- double **Nf** (double u, double u0, double uL)
- double **LL** (double L, double u0, double uL, RodSystemParameters const ∗pParameters)
- double **projForce** (double u0, double uL, double L, RodSystemParameters const ∗pParameters)
- int **equations** (const gsl_vector ∗pState, void ∗pVoidParameters, gsl_vector ∗pFun)

  *System of equations.*

### 5.34.1 Detailed Description

Definition of the RodSystem class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.35 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/rodsystem.h File Reference

Declaration of the RodSystem class.

```
#include <QString>
#include <vector>
#include <gsl/gsl_vector.h>
```

### Classes

- struct RSE::Core::Spans

  *Computed parameters of spans.*
- struct RSE::Core::RodSystemParameters

  *Parameters of a rod system.*
- class RSE::Core::RodSystem

### 5.35.1 Detailed Description

Declaration of the RodSystem class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.36 rodsystem.h

Go to the documentation of this file.

```
#ifndef RODSYSTEM_H
#define RODSYSTEM_H

#include <QString>
#include <vector>
#include <gsl/gsl_vector.h>

namespace RSE::Core
{

struct Cable;

struct Spans
{
    Spans(int numRods) : u0(numRods), uL(numRods), L(numRods) { }

    std::vector<double> u0;
    std::vector<double> uL;
    std::vector<double> L;
    double projectedForce;
};

struct RodSystemParameters
{
    std::vector<double> distances;
    double massPerLength;
    double youngsModulus;
    double area;
    double force;
    int numRods = 0;
};

class RodSystem
{
public:
    RodSystem(std::vector<double> distances, Cable const& cable, double force);
    // Get parameters of a system
    std::vector<double> const& distances() const { return mParameters.distances; }
    std::string const& nameCable() const { return mNameCable; }
    double force() const { return mParameters.force; }
    int numRods() const { return mParameters.numRods; }
    double massPerLength() const { return mParameters.massPerLength; }
    // Set parameters of a system
    void setDistances(std::vector<double> const& distances);
    void setCable(Cable const& cable);
    void setForce(double force) { mParameters.force = force; };
    // Compute parameters of spans
    Spans computeSpans();

private:
    RodSystemParameters mParameters;
    std::string mNameCable;
};

}

#endif // RODSYSTEM_H
```

## 5.37 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/scalardataobject.cpp File Reference

Implementation of the ScalarDataObject class.

```
#include "scalardataobject.h"
```

### 5.37.1 Detailed Description

Implementation of the ScalarDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.38 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/scalardataobject.h File Reference

Declaration of the ScalarDataObject class.

```
#include "abstractdataobject.h"
```

### Classes

- class RSE::Core::ScalarDataObject

  *Scalar data object.*

### 5.38.1 Detailed Description

Declaration of the ScalarDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.39 scalardataobject.h

Go to the documentation of this file.

```
#ifndef SCALARDATAOBJECT_H
#define SCALARDATAOBJECT_H

#include "abstractdataobject.h"

namespace RSE::Core
{

class ScalarDataObject : public AbstractDataObject
{
public:
    ScalarDataObject(QString const& name);
    ~ScalarDataObject();
    AbstractDataObject* clone() const override;
    DataItemType& addItem(DataValueType key) override;
    static quint32 numberInstances() { return smNumInstances; }
    virtual void import(QTextStream& stream) override;
private:
    static quint32 smNumInstances;
};

}

#endif // SCALARDATAOBJECT_H
```

## 5.40 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/solutionmanager.cpp File Reference

Definition of the SolutionManager class.

```
#include <QFileInfo>
#include <QDir>
#include "solutionoptions.h"
#include "solutionmanager.h"
```

### 5.40.1 Detailed Description

Definition of the SolutionManager class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.41 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/solutionmanager.h File Reference

Declaration of the SolutionManager class.

```
#include <QString>
#include <QProcess>
#include <QObject>
#include <QTextStream>
#include "project.h"
```

## Classes

- class RSE::Solution::SolutionManager

  *Class to control the solution process.*

### 5.41.1 Detailed Description

Declaration of the SolutionManager class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.42 solutionmanager.h

Go to the documentation of this file.

```
#ifndef SOLUTIONMANAGER_H
#define SOLUTIONMANAGER_H

#include <QString>
#include <QProcess>
#include <QObject>
#include <QTextStream>
#include "project.h"

namespace RSE::Solution
{

class SolutionOptions;

class SolutionManager : public QObject
{
    Q_OBJECT

public:
    SolutionManager(QString const& rootPath, QString const& relativeInputPath, QString const&
    relativeOutputPath);
    ~SolutionManager();
    void solveRodSystem(Core::Project& project, SolutionOptions const& options);
    void solveOptimization(Core::Project& project, SolutionOptions const& options);
    void runVisualizer();

signals:
    void outputSent(QByteArray);
    void rodSystemSolved();
    void optimizationSolved();
    void optimizationStepPerformed();

public slots:
    void stopSolution();

private:
    void processRodSystemStream();
    void processOptimizationStream();
    void runParserProcess();
    void writeOptimizationInput(QString const& pathFile, int numDampers, SolutionOptions const& options);
    int getRodSystemStatus();

private:
    QString mRootPath;
    QString mInputPath;
    QString mOutputPath;
    QProcess* mpRodSystemSolver = nullptr;
    QProcess* mpOptimizationSolver = nullptr;
};

}



#endif // SOLUTIONMANAGER_H
```

## 5.43 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/solutionoptions.cpp File Reference

Definition of the SolutionOptions class.

```
#include "solutionoptions.h"
```

### 5.43.1 Detailed Description

Definition of the SolutionOptions class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.44 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/solutionoptions.h File Reference

Declaration of the SolutionOptions class.

### Classes

- class RSE::Solution::SolutionOptions

### 5.44.1 Detailed Description

Declaration of the SolutionOptions class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.45 solutionoptions.h

Go to the documentation of this file.

```cpp

#ifndef SOLUTIONOPTIONS_H
#define SOLUTIONOPTIONS_H

namespace RSE
{

namespace Solution
{

class SolutionOptions
{
public:
    SolutionOptions() = default;
    SolutionOptions(int numCalcModes, int numDampModes, int stepModes, double tolTrunc);
    ~SolutionOptions() = default;
    // Get parameters
    int numCalcModes() const { return mNumCalcModes; }
    int numDampModes() const { return mNumDampModes; }
    int stepModes() const { return mStepModes; }
    double tolTrunc() const { return mTolTrunc; }
    // Set parameters
    void setNumCalcModes(int numCalcModes) { mNumCalcModes = numCalcModes; }
    void setNumDampModes(int numDampModes) { mNumDampModes = numDampModes; }
    void setStepModes(int stepModes) { mStepModes = stepModes; }
    void setTolTrunc(double tolTrunc) { mTolTrunc = tolTrunc; }

private:
    int mNumCalcModes;
    int mNumDampModes;
    int mStepModes;
    double mTolTrunc;
};

}

}


#endif // SOLUTIONOPTIONS_H
```

## 5.46 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/core/support.cpp File Reference

Definition of the Support class.

```cpp
#include "support.h"
```

### 5.46.1 Detailed Description

Definition of the Support class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.47 /home/qinterfly/Library/Projects/Current/RodSystem←Estimator/src/core/support.h File Reference

Declaration of the Support class.

### Classes

- class RSE::Core::Support

  *Class to aggregate data of supports.*

### 5.47.1 Detailed Description

Declaration of the Support class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.48 support.h

Go to the documentation of this file.

```
#ifndef SUPPORT_H
#define SUPPORT_H

namespace RSE::Core
{

class Support
{
public:
    Support(double longitudinalStiffness, double verticalStiffness);
    ~Support() = default;
    // Get characteristics
    double longitudinalStiffness() const { return mLongitudinalStiffness; }
    double verticalStiffness() const { return mVerticalStiffness; }
    // Set characteristics
    void setLongitudinalStiffness(double longitudinalStiffness) { mLongitudinalStiffness =
    longitudinalStiffness; }
    void setVerticalStiffness(double verticalStiffness) { mVerticalStiffness = verticalStiffness; }

private:
    double mLongitudinalStiffness;
    double mVerticalStiffness;
};

}

#endif // SUPPORT_H
```

## 5.49 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/utilities.cpp File Reference

Definition of utilites.

```
#include <QDebug>
#include <QString>
#include <QFile>
#include <QDir>
#include <QPair>
#include "utilities.h"
```

### 5.49.1 Detailed Description

Definition of utilites.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.50 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/utilities.h File Reference

Declaration of utilities.

```
#include <QSharedPointer>
#include "abstractdataobject.h"
```

### Functions

- QPair< Core::AbstractDataObject::ObjectType, QSharedPointer< QFile > > **RSE::Utilities::File::get↩ DataObjectFile** (QString const &path, QString const &fileName)

  *Retrieve a pair consisted of a data object file and its type.*
- QString **RSE::Utilities::File::loadFileContent** (QString const &path)

  *Load all the content of a file.*

### 5.50.1 Detailed Description

Declaration of utilities.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.51 utilities.h

Go to the documentation of this file.

```

#ifndef UTILITIES_H
#define UTILITIES_H

#include <QSharedPointer>
#include "abstractdataobject.h"

class QFile;
class QString;

namespace RSE
{

namespace Utilities
{

namespace File
{

QPair<Core::AbstractDataObject::ObjectType, QSharedPointer<QFile>> getDataObjectFile(QString const& path,
    QString const& fileName);
QString loadFileContent(QString const& path);

}

}

}

#endif // UTILITIES_H
```

## 5.52 /home/qinterfly/Library/Projects/Current/RodSystem←Estimator/src/core/vectordataobject.cpp File Reference

Implementation of the VectorDataObject class.

```
#include "vectordataobject.h"
```

### Variables

- const IndexType **skNumElements** = 3

### 5.52.1 Detailed Description

Implementation of the VectorDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.53 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/core/vectordataobject.h File Reference

Declaration of the VectorDataObject class.

```
#include "abstractdataobject.h"
```

### Classes

- class RSE::Core::VectorDataObject

  *Vector data object.*

### 5.53.1 Detailed Description

Declaration of the VectorDataObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.54 vectordataobject.h

Go to the documentation of this file.

```
#ifndef VECTORDATAOBJECT_H
#define VECTORDATAOBJECT_H

#include "abstractdataobject.h"

namespace RSE::Core
{

class VectorDataObject : public AbstractDataObject
{
public:
    VectorDataObject(QString const& name);
    ~VectorDataObject();
    AbstractDataObject* clone() const override;
    DataItemType& addItem(DataValueType key) override;
    static quint32 numberInstances() { return smNumInstances; }
    virtual void import(QTextStream& stream) override;

private:
    static quint32 smNumInstances;
};

}

#endif // VECTORDATAOBJECT_H
```

## 5.55 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/framecollection.h File Reference

Collection of the data associated with the specified frame.

```
#include "frameobject.h"
```

### Classes

- struct KLP::EnergyFrame

  *Energy quantities associated with a frame.*
- struct KLP::StateFrame

  *Kinematic and dynamic quantities associated with a frame.*
- struct KLP::FrameCollection

  *Set of all quantities belonged to a frame.*

### Typedefs

- using **KLP::FloatFrameObject** = FrameObject< float >

### Variables

- const int **KLP::kNumDirections** = 3

### 5.55.1 Detailed Description

Collection of the data associated with the specified frame.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.56 framecollection.h

Go to the documentation of this file.

```

#ifndef FRAMECOLLECTION_H
#define FRAMECOLLECTION_H

#include "frameobject.h"

namespace KLP
{

const int kNumDirections = 3;

using FloatFrameObject = FrameObject<float>;

struct EnergyFrame
{
    FloatFrameObject kinetic;
    FloatFrameObject potential;
    FloatFrameObject full;
};

struct StateFrame
{
    FloatFrameObject displacements[kNumDirections];
    FloatFrameObject rotations[kNumDirections];
    FloatFrameObject forces[kNumDirections];
    FloatFrameObject moments[kNumDirections];
};

struct FrameCollection
{
    int numRods;
    FloatFrameObject parameter;
    FloatFrameObject naturalLength;
    FloatFrameObject accumulatedNaturalLength;
    FloatFrameObject coordinates[kNumDirections];
    StateFrame state;
    StateFrame projectedState;
    StateFrame firstDerivativeState;
    StateFrame secondDerivativeState;
    StateFrame errorState;
    FloatFrameObject strain;
    std::vector<StateFrame> modalStates;
    FloatFrameObject frequencies;
    EnergyFrame energy;
};

}

#endif // FRAMECOLLECTION_H
```

## 5.57 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/frameobject.cpp File Reference

Definition of the FrameObject class.

`#include "frameobject.h"`

### 5.57.1 Detailed Description

Definition of the FrameObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.58 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/frameobject.h File Reference

Declaration of the FrameObject class.

```
#include <QDebug>
#include "frameobjectiterator.h"
```

### Classes

- class KLP::FrameObject< T >

### Functions

- template<typename K >
  QDebug **KLP::operator**<< (QDebug stream, FrameObject< K > &frameObject)

### 5.58.1 Detailed Description

Declaration of the FrameObject class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.59 frameobject.h

Go to the documentation of this file.

```
#ifndef FRAMEOBJECT_H
#define FRAMEOBJECT_H

#include <QDebug>
#include "frameobjectiterator.h"

namespace KLP
{

template <typename T>
class FrameObject
{
public:
    using iterator = FrameObjectIterator<T>;

public:
    FrameObject(T const* pData = nullptr, T normFactor = 1.0, qint64 size = 0, qint64 step = 1);
    ~FrameObject() = default;
    bool isEmpty() const { return !mpData; }
    qint64 size() const { return mSize; }
    iterator begin() { return iterator(&mpData[0], mNormFactor, mStep); }
    iterator end() { return iterator(&mpData[mSize], mNormFactor, mStep); }
    iterator operator[](int index) { return begin() + index; }
    template<typename K> friend QDebug operator«(QDebug stream, FrameObject<K>& frameObject);

private:
    T const* mpData;
    T mNormFactor;
    qint64 mSize;
    qint64 mStep;
};

template<typename K>
inline QDebug operator«(QDebug stream, FrameObject<K>& frameObject)
{
    stream = stream.noquote();
    for (auto it = frameObject.begin(); it != frameObject.end(); ++it)
        stream « QString::number(*it) « Qt::endl;
    return stream;
}

}

#endif // FRAMEOBJECT_H
```

## 5.60 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/frameobjectiterator.cpp File Reference

Definition of the FrameObjectIterator class.

`#include "frameobjectiterator.h"`

### 5.60.1 Detailed Description

Definition of the FrameObjectIterator class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.61 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/frameobjectiterator.h File Reference

Declaration of the FrameObjectIterator class.

`#include <QtGlobal>`

### Classes

- class KLP::FrameObjectIterator< T >

  *Class to iterate through data of a record.*

### 5.61.1 Detailed Description

Declaration of the FrameObjectIterator class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.62 frameobjectiterator.h

Go to the documentation of this file.

```
#ifndef FRAMEOBJECTITERATOR_H
#define FRAMEOBJECTITERATOR_H

#include <QtGlobal>

namespace KLP
{

template <typename T>
class FrameObjectIterator
{
public:
    using self_type          = FrameObjectIterator<T>;
    using iterator_category = std::random_access_iterator_tag;
    using difference_type   = std::ptrdiff_t;
    using value_type        = T;
    using pointer           = T const*;
    using reference         = T const&;

public:
    FrameObjectIterator(pointer pData, T normFactor, qint64 step);
    // Access
    value_type operator*() { return *mpData * mNormFactor; }
    // Operators
    self_type& operator++() { mpData += mStep; return *this; }
    self_type operator++(int) { self_type temp = *this; ++(*this); return temp; }
    self_type operator+(const difference_type& movement) { auto pOldData = mpData; mpData += movement *
    mStep; self_type temp = *this; mpData = pOldData; return temp; }
    difference_type operator-(const FrameObjectIterator& another) const { return mpData - another.mpData;
    }
    // Comparison
    friend bool operator== (self_type const& first, self_type const& second) { return first.mpData ==
    second.mpData; };
    friend bool operator!= (self_type const& first, self_type const& second) { return !(first == second);
    };

private:
    pointer mpData;
    T mNormFactor;
    qint64 const mStep;
};

}

#endif // FRAMEOBJECTITERATOR_H
```

## 5.63 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/index.h File Reference

Specification of a structure to index records.

```
#include <QtGlobal>
#include "types.h"
#include <vector>
```

### Classes

- struct KLP::IndexData

  *Data of each record.*
- struct KLP::Index

  *Structure to navigate through records.*

### 5.63.1 Detailed Description

Specification of a structure to index records.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.64 index.h

Go to the documentation of this file.

```
#ifndef INDEX_H
#define INDEX_H

#include <QtGlobal>
#include "types.h"
#include <vector>

namespace KLP
{

struct IndexData
{
    qint64 position = 0;
    qint64 size = 0;
    qint64 step = 1;
    qint64 partSize = 0;
};

struct Index
{
    Index() { data.resize(RecordType::MAX_RECORD); }
    std::vector<IndexData> data;
    quint64 recordShift = 0;
    quint64 relativeDataShift = 0;
};

}

#endif // INDEX_H
```

## 5.65 /home/qinterfly/Library/Projects/Current/RodSystem↵ Estimator/src/klp/result.cpp File Reference

Definition of the Result class.

```
#include <QFile>
#include "result.h"
```

### 5.65.1 Detailed Description

Definition of the Result class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.66 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/result.h File Reference

Declaration of the Result class.

```
#include <QString>
#include "index.h"
#include "framecollection.h"
```

### Classes

- class KLP::Result

  *Class to aggregate all the records.*

### 5.66.1 Detailed Description

Declaration of the Result class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.67 result.h

Go to the documentation of this file.

```
#ifndef RESULT_H
#define RESULT_H

#include <QString>
#include "index.h"
#include "framecollection.h"

namespace KLP
{

class Result
{
public:
    Result(QString const& pathFile);
    ~Result() = default;
    bool isEmpty() const { return mContent.isEmpty(); }
    int numRods(qint64 iFrame) const;
    FloatFrameObject getFrameObject(qint64 iFrame, RecordType type, float normFactor = 1.0f, qint64 shift
    = 0) const;
    FrameCollection getFrameCollection(qint64 iFrame) const;
    void update();

private:
    bool read();
    void buildIndex();
    void setStateFrameData(StateFrame& state, RecordType type, qint64 iFrame, qint64 iStartData,
    std::vector<float> const& normFactors) const;

private:
    QString const mkPathFile;
    QByteArray mContent;
    std::vector<Index> mIndex;
    qint64 mNumRecords;
    std::vector<float> mTime;
    char mNumBytesRod;
};

}

#endif // RESULT_H
```

## 5.68 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/klp/types.h File Reference

Specification of data types in a KLP file.

### Enumerations

- enum KLP::RecordType {
  **R** = 2 , **Xi** = 3 , **S** = 4 , **SS** = 5 ,
  **X1** = 6 , **X2** = 7 , **X3** = 8 , **U** = 9 ,
  **Ut** = 10 , **Utt** = 11 , **EPS** = 12 , **Ul** = 13 ,
  **BETA** = 15 , **Qm** = 16 , **qm** = 17 , **AE** = 18 ,
  **MF** = 19 , **MV** = 20 , **ND** = 21 , **FM** = 22 ,
  **ERR** = 23 , **MASS** = 24 , **RMASS** = 25 , **IP** = 26 ,
  **CSM** = 27 , **CS** = 28 , **CSP** = 29 , **CSE** = 30 ,
  **CSG** = 31 , **FI** = 32 , **FM2** = 33 , **EM** = 34 ,
  **EN** = 35 , **MAX_RECORD** }

  *Types of records.*
- enum KLP::NondimensionalType {
  **Time** = 0 , **Displacement** = 1 , **Force** = 2 , **Moment** = 3 ,
  **DistributedForce** = 7 , **DistributedMoment** = 8 , **Speed** = 9 , **Acceleration** = 10 ,
  **MAX_NONDIM** }

  *Types of nondimensional coefficients.*

### 5.68.1 Detailed Description

Specification of data types in a KLP file.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.69 types.h

Go to the documentation of this file.

```
#ifndef TYPES_H
#define TYPES_H

namespace KLP
{

enum RecordType
{
    R     = 2,  // Rods
    Xi    = 3,  // Parameter
    S     = 4,  // Natural length
    SS    = 5,  // Accumulated natural length
    X1    = 6,  // Coordinate X1
    X2    = 7,  // Coordinate X2
    X3    = 8,  // Coordinate X3
    U     = 9,  // State vector: [U1, U2, U3, w1, w2, w3, Q1, Q2, Q3, M1, M2, M3]
    Ut    = 10, // First-order derivative of the state vector with respect to time
    Utt   = 11, // Second-order derivative of the state vector with respect to time
    EPS   = 12, // Tensile-compressive strain
    Ul    = 13, // Projected state vector: [U1L, U2L, U3L, w1, w2, w3, Q1L, Q2L, Q3L, M1L, M2L, M3L]
    BETA  = 15, // Rotation matrix
    Qm    = 16, // Loads
    qm    = 17, // Distributed loads
    AE    = 18, // Aerodynamic
    MF    = 19, // Eigenfrequencies
    MV    = 20, // Eigenvectors
    ND    = 21, // Nondimensional coefficients [use NondimensionalType to navigate]
    FM    = 22, // Finite element model
    ERR   = 23, // Computational errors of the state vector
    MASS  = 24, // Total mass and the center of gravity
    RMASS = 25, // Masses of rods
    IP    = 26, // Cross sections
    CSM   = 27, // ?
    CS    = 28, // ?
    CSP   = 29, // ?
    CSE   = 30, // ?
    CSG   = 31, // ?
    FI    = 32, // Finite element image: set of coordinates (X, Y, Z) to plot lines
    FM2   = 33, // ?
    EM    = 34, // Effective masses
    EN    = 35, // Energy
    MAX_RECORD
};

enum NondimensionalType
{
    Time               = 0,
    Displacement       = 1,
    Force              = 2,
    Moment             = 3,
    DistributedForce   = 7,
    DistributedMoment  = 8,
    Speed              = 9,
    Acceleration       = 10,
    MAX_NONDIM
};

}

#endif // TYPES_H
```

## 5.70 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/main/main.cpp File Reference

Startup.

```
#include <QFile>
#include <QApplication>
#include <QFontDatabase>
#include "mainwindow.h"
#include "utilities.h"
```

### Functions

- int **main** (int argc, char ∗argv[ ])
  *Startup point.*

### 5.70.1 Detailed Description

Startup.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.71 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/viewers/convergenceviewer.cpp File Reference

Definition of the ConvergenceViewer class.

```
#include <QVBoxLayout>
#include "convergenceviewer.h"
```

### 5.71.1 Detailed Description

Definition of the ConvergenceViewer class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.72 /home/qinterfly/Library/Projects/Current/RodSystem↩ Estimator/src/viewers/convergenceviewer.h File Reference

Declaration of the ConvergenceViewer class.

```
#include <QWidget>
#include "array.h"
#include "qcustomplot.h"
```

### Classes

- class RSE::Viewers::ConvergenceViewer

  *Class to represent convergence of viscosities.*

### 5.72.1 Detailed Description

Declaration of the ConvergenceViewer class.

**Author**

Pavel Lakiza

**Date**

July 2022

## 5.73 convergenceviewer.h

Go to the documentation of this file.

```
#ifndef CONVERGENCEVIEWER_H
#define CONVERGENCEVIEWER_H

#include <QWidget>
#include "array.h"
#include "qcustomplot.h"

namespace RSE
{

namespace Viewers
{

class ConvergenceViewer : public QWidget
{
public:
    ConvergenceViewer(QString const& pathFile, QWidget* pParent = nullptr);
    ~ConvergenceViewer();
    void plot();

private:
    void initialize();
    bool read();

private:
    QString const mkPathFile;
    QCustomPlot* mpFigure;
    QStringList mAvailableColors;
    QVector<QCPScatterStyle::ScatterShape> mAvailableShapes;
    QVector<int> mCalcModes;
    Core::Array<double> mDampingValues;
};

}

}

#endif // CONVERGENCEVIEWER_H
```