PDE1D

Generated by Doxygen 1.8.1

Fri May 3 2013 09:06:30

Contents

Chapter 1

Todo List

Class SolvWidget

make all variables accessible for plotting by name

make the new solver a shared object and load it with dlopen so pde1d does not need to be modified.

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 ColorDelegate Class Reference

Public Member Functions

- **ColorDelegate** (QObject ∗parent=0)
- virtual QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- virtual void **setEditorData** (QWidget ∗editor, const QModelIndex &index) const
- virtual void **setModelData** (QWidget ∗editor, QAbstractItemModel ∗model, const QModelIndex &index) const

Static Public Member Functions

- static const QIcon **getIcon** (QColor col)
- static bool initColors ()

Static Public Attributes

- static QList< QColor > **colors**
- static bool **cinit** = ColorDelegate::initColors()

4.1.1 Member Function Documentation

4.1.1.1 static bool ColorDelegate::initColors ( ) [inline], [static]

#name

#name

#name

#name

#name

#name

darkGray

gray #name

darkGreen

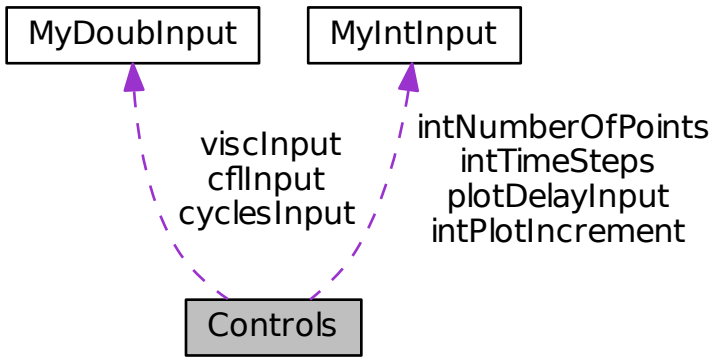
black #name

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

- colordelegate.h
- colordelegate.cpp

4.2 Controls Class Reference

Collaboration diagram for Controls:



Public Member Functions

- **Controls** (const QString &title="Controls", QWidget *parent=0, Qt::WindowFlags flags=0)
- **Controls** (const Controls &other)
- virtual Controls & **operator=** (const Controls &other)
- virtual bool **operator==** (const Controls &other) const
- void **setupUi** ()
- void **retranslateUi** ()

Public Attributes

- QWidget * **dockWidgetContents**
- QVBoxLayout * **verticalLayout**
- QLabel * **pdeLabel**
- QComboBox * **pdeBox**
- QComboBox * **addSolvCombo**
- QLabel * **sizeLabel**

- MyIntInput * **intNumberOfPoints**
- QLabel * **cyclesLabel**
- MyDoubInput * **cyclesInput**
- QGridLayout * **gridLayout**
- QLabel * **cflLabel**
- QLabel * **viscLabel**
- MyDoubInput * **cflInput**
- MyDoubInput * **viscInput**
- QLabel * **stepsLabel**
- QLabel * **incrementLabel**
- MyIntInput * **intTimeSteps**
- MyIntInput * **intPlotIncrement**
- QLabel * **label**
- MyIntInput * **plotDelayInput**
- QPushButton * **savePlotButton**
- QPushButton * **saveImageButton**
- QCheckBox * **animationCheck**
- QCheckBox * **checkBox**
- QSpacerItem * **verticalSpacer_2**
- QHBoxLayout * **horizontalLayout**
- QHBoxLayout * **horizontalLayout2**
- QPushButton * **resetButton**
- QPushButton * **runButton**
- QPushButton * **stopButton**

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

- controls.h
- controls.cpp

4.3 CurvesModel Class Reference

Signals

- void **newdata** ()

Public Member Functions

- **CurvesModel** (QObject *parent=0)
- virtual QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- virtual int **columnCount** (const QModelIndex &parent=QModelIndex()) const
- virtual int **rowCount** (const QModelIndex &parent=QModelIndex()) const
- virtual bool setData (const QModelIndex &index, const QVariant &value, int role=Qt::EditRole)
- virtual QVariant **headerData** (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const
- virtual bool **insertRows** (int row, int count, const QModelIndex &parent=QModelIndex())
- virtual bool **removeRows** (int row, int count, const QModelIndex &parent=QModelIndex())
- virtual Qt::ItemFlags **flags** (const QModelIndex &index) const
- void **setSolvers** (QList< SolvWidget * > *eeWidgets)
- QColor **getSymbolColor** (int row) const
- int **getSymbolSize** (int row) const
- void **solverAdded** ()
- void **removeSolver** (int row)

Static Public Member Functions

- static QString **colorName** (QColor col)
- static QIcon **penIcon** (QPen pen)

Protected Attributes

- QList< SolvWidget * > * **solvers**
- QStringList **penStyles**

4.3.1 Member Function Documentation

4.3.1.1 bool CurvesModel::setData (const QModelIndex & *index*, const QVariant & *value*, int *role* = `Qt::EditRole`) [virtual]

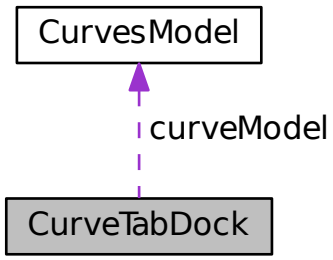
This makes a copy of Pen

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

- curvesmodel.h
- curvesmodel.cpp

4.4 CurveTabDock Class Reference

Collaboration diagram for CurveTabDock:



Public Member Functions

- **CurveTabDock** (const QString &title, QWidget *parent=0, Qt::WindowFlags flags=0)
- **CurveTabDock** (QWidget *parent=0, Qt::WindowFlags flags=0)
- void **setCurvesModel** (CurvesModel *newModel)

Protected Member Functions

- void **setupUi** (const CurveTabDock *ctd=NULL)

Protected Attributes

- QWidget * **dockWidgetContents**
- QHBoxLayout * **verticalLayout**
- QTableView * **curveTable**
- CurvesModel * **curveModel**

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

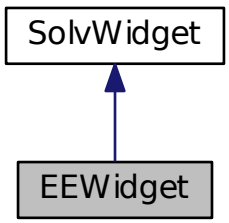
- curvetabdock.h
- curvetabdock.cpp

4.5 EEWidget Class Reference

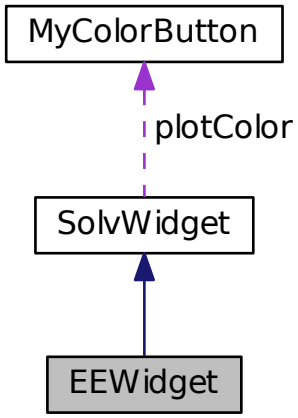
First Order Euler Explicit.

`#include <eewidget.h>`

Inheritance diagram for EEWidget:



Collaboration diagram for EEWidget:



Public Member Functions

- **EEWidget** (const EEWidget &other)
- virtual EEWidget & **operator=** (const EEWidget &other)
- virtual bool **operator==** (const EEWidget &other) const
- virtual void step (const size_t nStep)
- virtual void setSize (const size_t size=100)

  todo - remove from hear - the default should work after fixing step to use the new variables
- virtual bool canSolve (int equ)
- void **advection** (const int nStep)
- void **burger** (const int nStep)

Protected Attributes

- double **upwind**
- bool nulimit

  degree of upwinding for convection, 0 = central, 1.0 = 1st order upwind
- bool finVol

  if true, reduce nu to account for upwind effective viscosity

Additional Inherited Members

4.5.1 Detailed Description

First Order Euler Explicit.

The Euler Explicit method is one the simplest numerical initial value Ordinary Differential Equation solvers. It is based on an approximation to the derivative where the change in independent variable is small but not approaching zero.

The derivative or slope of a function u at a location, x_i , and time, t_n , with respect to time is defined as

$$\frac{\partial u}{\partial t} = \lim_{\delta t \to 0} \frac{u(x_i, t_n + \delta t) - u(x_i, t_n)}{\delta t}$$

where x_i is the ith x point and t_n is the nth time step. By replacing δt with a small Δt on the right side and solving for $u(x_i, t_n + \Delta t)$ an approximation for the value of u at the new time can be obtained.

$$u(x_i, t_n + \Delta t) = u(x_i, t_n) + \Delta t \frac{\partial u}{\partial t}$$

Time Component

The Euler Explicit method uses the partial derivative with respect to time at the current time to predict the value at the next time step, that is

$$u_i^{n+1} = u_i^n + \Delta t u_t$$

where u_i^{n+1} is notation for $u(x_i, t_n + \Delta t)$ and u_t is shorthand notation for $\frac{\partial u}{\partial t}$. Thoughout this discusion subscripts of t or x will represent partial derivatives while supscripts of i, n or k will represent indecies.

The temporal partial derivative, u_t , is solved from the partial differential equation(PDE). Where the spacial derivatives are approximated from finite differences.

Spacial Components

The spacial components used here all assume uniform spacing, Δx. The first spacial derivatives $\frac{\partial u}{\partial x} \equiv u_x$ are approximated as upwind differences based on the sign of it's coefficient. That is $cu_x \approx c\left(\frac{u_i - u_{i-1}}{\Delta x}\right)$ if c is positive and $cu_x \approx c\left(\frac{u_{i+1} - u_i}{\Delta x}\right)$ if c is negative. Another way to express this is

$$cu_x \approx \frac{c}{2\Delta x}(u_{i+1} - u_{i-1}) + \frac{|c|}{2\Delta x}(-u_{i+1} + 2u_i - u_{i-1})$$

where $|c|$ is the absolute value of c.

For second derivative terms, a central difference is used, $\nu u_{xx} \approx \nu \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$.

Advection/Diffusion Equation

Substituting the above approximations into the Advection/Diffusion equation, $u_t + cu_x = \nu u_{xx}$, results in

$$u_i^{n+1} = u_i^n + \Delta t \quad \left(\quad -\frac{c}{2\Delta x}(u_{i+1} - u_{i-1}) - \frac{|c|}{2\Delta x}(-u_{i+1} + 2u_i - u_{i-1}) \right.$$
$$\left. + \nu \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} \right)$$

or combining a few terms give the difference equation

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x} \quad \left(\quad \frac{-c}{2}(u_{i+1} - u_{i-1}) \right.$$
$$\left. + (\frac{\nu}{\Delta x} + \frac{|c|}{2})(u_{i+1} - 2u_i + u_{i-1}) \right.$$

.

Note

The term with $|c|$ is added to the diffusion term, ν , which shows that upwinding has the same form as adding additional diffusion. The 'nulimit' option described below reduces ν to account for this artificial dissipation.

If x is not uniform $\Delta x = \frac{x_{i+1}-x_{i-1}}{2}$ which is derived as follows.

Finite Volume Form

In one dimension we will call the line from midpoints $x_{i-1/2}$ to $x_{i+1/2}$ the volume. Form a box in space and time with corners at $(x_{i-1/2},t_n),(x_{i+1/2},t_n),(x_{i-1/2},t_{n+1}),(x_{i+1/2},t_{n+1})$. Then rewritting the advection diffusion equation as $u_t + f_x = 0$ where $f = cu - \nu u_x$, since ν is a constant. Integrating the above equation in time gives $u(x,t_n+\Delta t) - u(x,t_n) = -\int_{t_n}^{t_n+\Delta t} f_x dt$. Then the 'volume' average of u , $\bar{u}$, at the new time level becomes

$$\begin{aligned}
\overline{u^{n+1}} &= \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n+\Delta t)\,dx \\
&= \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n) - \int_{t_n}^{t_n+\Delta t} f_x\,dt\,dx \\
&= \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t_n)\,dx - \int_{t_n}^{t_n+\Delta t}\int_{x_{i-1/2}}^{x_{i+1/2}} f_x\,dx\,dt
\end{aligned}$$

The integrals can be reversed because we assume that f_x is continuous and differentiable. Also, from the Fundamental theorem of calculus (the divergence theorem)

$$\int_{x_{i-1/2}}^{x_{i+1/2}} f_x\,dx = f(x_{i+1/2},t) - f(x_{i-1/2},t)$$

which can be substituted into the above equation to yield

$$\overline{u_i^{n+1}} = \overline{u_i^n} - \frac{1}{\Delta x}\int_{t_n}^{t_n+\Delta t} f(x_{i+1/2},t) - f(x_{i-1/2},t)dt$$

Then assuming f is constant with respect to time over the interval from t to $t+\Delta t$ and solving for $\bar{u}$ we obtain

$$\begin{aligned}
\overline{u_i^{n+1}} &= \overline{u_i^n} + \frac{\Delta t}{\Delta x}\left(f(x_{i-1/2}) - f(x_{i+1/2})\right) \\
&= \overline{u_i^n} + \frac{\Delta t}{\Delta x}\left(cu_{i-1/2} - \nu u_x|_{i-1/2} - cu_{i+1/2} + \nu u_x|_{i+1/2}\right)
\end{aligned}$$

Again upwinding for the advection term gives

$$cu_{i-1/2} = \frac{c}{2}(u_{i-1}+u_i) + \frac{|c|}{2}(u_{i-1}-u_i)$$

and

$$cu_{i+1/2} = \frac{c}{2}(u_i+u_{i+1}) + \frac{|c|}{2}(u_i-u_{i+1})$$

. Also, use the approximation

$$u_x|_{i-1/2} = \frac{u_i - u_{i-1}}{\Delta x}$$

and

$$u_x|_{i+1/2} = \frac{u_{i+1} - u_i}{\Delta x}$$

. Dropping the overbar for the average of u gives

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x}\left(\frac{c}{2}(u_{i-1}-u_{i+1})\right) + \frac{\Delta t}{\Delta x^2}\left(\nu + \frac{|c|\Delta x}{2}\right)(u_{i+1} - 2u_i + u_{i-1})$$

. This is the same result as the finite difference equation above.

Note

For conservation law PDEs the derivation usually involves the reverse process. That is the finite volume form of the conservation equations is converted into the finite difference form using the divergence theorm. The finite volume form is then a more natural expression of the conservation equations. However, accuracy and stability analysis is simpler in the finite difference form.

Accuracy (Convergence)

To verify the accuracy of this equation substitute the Taylor Series expansion of u about x_i.

$$u_{i-1} = u(x_i - \Delta x, t_n) = u(x_i) - \Delta x u_x(x_i,t_n) + \frac{\Delta x^2}{2} u_{xx}(x_i,t_n) - \frac{\Delta x^3}{3!} u_{xxx}(x_i,t_n) + R_k(x)$$

$$u_{i+1} = u(x_i + \Delta x, t_n) = u(x_i) + \Delta x u_x(x_i,t_n) + \frac{\Delta x^2}{2} u_{xx}(x_i,t_n) + \frac{\Delta x^3}{3!} u_{xxx}(x_i,t_n) + R_k(x)$$

where $R_k(x)$ is a remander or error term and for functions u that are $k+1$ times differentiable on the open interval and continuous on the closed interval between x and $x \pm \Delta x$ is

$$R_k(x) = \frac{u_{x^{(k+1)}}(\xi_L)}{(k+1)!} \Delta x^{k+1}$$

for some ξ_L in the interval between x and $x \pm \Delta x$. In the above equations $k = 3$ and $u_{x^{(k+1)}}$ is the (k+1)$^{\text{th}}$ partial derivative of u with respect to x.

With these relations, the difference terms become

$$u_{i+1} - u_{i-1} = 2\Delta x u_x + 2\frac{\Delta x^3}{3!} u_{xxx} + R_k(x)$$

and

$$u_{i+1} - 2u_i + u_{i-1} = \Delta x^2 u_{xx} + 2\frac{\Delta x^4}{4!} u_{x^{(4)}} + R_k(x)$$

and for the time derivative

$$u_i^{n+1} = u_i^n + \Delta t u_t + \frac{\Delta t^2}{2} u_{tt} + \frac{\Delta t^3}{3!} u_{ttt} + R_k(t)$$

. Then substituting into the differnce equation above gives

$$\begin{aligned} u_i^n + \Delta t u_t \quad &+ \quad \frac{\Delta t^2}{2} u_{tt} + \frac{\Delta t^3}{3!} u_{ttt} \\ &= \quad u_i^n + \frac{\Delta t}{\Delta x}\left(-\frac{c}{2}(2\Delta x u_x + 2\frac{\Delta x^3}{3!} u_{xxx}) + \left(\frac{\nu}{\Delta x} + \frac{|c|}{2}\right)(\Delta x^2 u_{xx} + 2\frac{\Delta x^4}{4!} u_{x^{(4)}})\right) + \ldots \end{aligned}$$

. Collecting some terms, keeping only terms up to first order deltas gives

$$u_t + c u_x - \nu u_{xx} = \frac{|c|\Delta x}{2} u_{xx} - \frac{\Delta t}{2} u_{tt}$$

which approaches the advection/diffusion equation as Δx and Δt approach zero. Because the error term on the right hand side involves first order deltas the method is first order in time and space.

The 'nulimit' option reduces the viscosity by the aparent viscosity do to upwinding through the absolute value of wave speed, $|c|$.

$$\nu_{reduced} = max(0.0, \nu - \frac{|c|\Delta x}{2})$$

Stability

Stability is one of the major cornerstones to numerical methods. Indeed if a method is not stable it will not give the correct solution to the partial differential equation. When a numerical method gives the correct solution is called consistancy. The are two requirements for consistancy, convergence and stability. Convergence for the Explicit Euler method was shown above under accuracy. Convergence indicates that the numerical difference equation approaches the partial differential equation as the step size in both time and space approach zero. However, convergence may be limitted by roundoff or truncation errors introduced by the finite precision of computer hardware.

Stability analysis for non-linear equation and including boundary conditions becomes very difficult. However, for linear systems with periodic bounday conditions the stability analysis can give insight into the stability requirements of the numerical method. Linear stability is a requirement for stability but it does not in the general case, prove stability. Non-linear equations can be linearized for stability analysis but non-linear interactions may still be unstable which would not be indicated in the linear analysis.

For linear partial differential systems the solution is a sum of solutions that depend on the initial and boundary conditions. The solutions can be based on sum of sine and cosine waves since derivatives of sines and cosines are sines and cosines. Again, since the system is linear we can examine one component, a sine and a cosine wave at a frequency, independently. If any component is unstable, the method is unstable.

Linear Diffusion Stability

First, examine the diffusion part of the equation

$$\begin{aligned} u_t &= \nu\, u_{xx} \\ \frac{u_i^{n+1}-u_i^n}{\Delta t} &= \nu\, \frac{u_{i-1}^n - 2u_i^n + u_{n+1}^n}{\Delta x^2} \end{aligned}$$

where one component of $u_k^n = a\sin(\omega_k x) + b\cos(\omega_k x)$. Where k indicates a solution component that is continuous in x at a frequency of $\omega_k = k\,\omega$. Then using the trigonometric angle addition formulas for sine and cosine

See also

http://mathworld.wolfram.com/TrigonometricAdditionFormulas.html, http://www.-themathpage.com/atrig/sum-proof.htm or http://en.wikipedia.org/wiki/Proofs_-of_trigonometric_identities for angle sum formulas and proofs

$$\begin{aligned} u_k^n|_{i+1} &= a\sin(\omega_k x + \omega_k \Delta x) + b\cos(\omega_k x + \omega_k \Delta x) \\ &= a(\cos(\omega_k x)\sin(\omega_k \Delta x) + \sin(\omega_k x)\cos(\omega_k \Delta x)) \\ &\quad + b(\cos(\omega_k x)\cos(\omega_k \Delta x) - \sin(\omega_k x)\sin(\omega_k \Delta x)) \\ u_k^n|_{i-1} &= a(-\cos(\omega_k x)\sin(\omega_k \Delta x) + \sin(\omega_k x)\cos(\omega_k \Delta x)) \\ &\quad + b(\cos(\omega_k x)\cos(\omega_k \Delta x) + \sin(\omega_k x)\sin(\omega_k \Delta x)) \end{aligned}$$

results in

$$\begin{aligned} u_{k,i-1}^n - 2u_{k,i}^n + u_{k,i+1}^n &= 2a\,\sin(\omega_k x)\cos(\omega_k \Delta x) + 2b\cos(\omega_k x)\cos(\omega_k \Delta x) \\ &\quad -2a\,\sin(\omega_k x) - 2b\cos(\omega_k x) \\ &= 2(a\,\sin(\omega_k x)(\cos(\omega_k \Delta x) - 1) \\ &\quad + b\cos(\omega_k x)(\cos(\omega_k \Delta x) - 1) \\ &= 2u_k^n(\cos(\omega_k \Delta x) - 1) \end{aligned}$$

Substituting into the PDE gives

$$\begin{aligned} u_k^{n+1} &= u_k^n + 2\frac{\nu\,\Delta t}{\Delta x^2}u_k^n(\cos(\omega_k\Delta x)-1) \\ &= u_k^n\left(1+2\frac{\nu\,\Delta t}{\Delta x^2}(\cos(\omega_k\Delta x)-1)\right) \end{aligned}$$

This indicates the component is multiplied by $g = 1+2\frac{\nu\,\Delta t}{\Delta x^2}(\cos(\omega_k\Delta x)-1)$ each time step. If $|g|>1$ the magnitude of that component will increase without bound as *n* approaches infinity. Since $-2 \le (\cos(\omega_k\Delta x)-1) \le 0$,

$$1-4\frac{\nu\,\Delta t}{\Delta x^2} \le g \le 1\ .$$

Then $|g| \le 1$ requires

$$1-4\frac{\nu\,\Delta t}{\Delta x^2} \ge -1$$

or, since all terms, $(\nu,\Delta t,\Delta x)$, are positive, $\frac{\nu\,\Delta t}{\Delta x^2} \le \frac{1}{2}$ or $\Delta t \le \frac{\Delta x^2}{\nu 2}$. This indicates a conditional stability where the time step is limitted by stability constraints.

Note

A series of values at equally spaced points can be interpolated with a series of sine and cosine functions. The process of converting from point values to sine and cosine functions is a Discrete Fourier Transform. A special form of Discrete Fourier Transform is the Fast Fourier Transform that is both computationally efficient and less susceptible to round off errors than straight forward Discrete Fourier Transform computation.

That is for each i

$$u(x_i) = \sum_{k=0}^{K} a_k\,\sin(\omega_k x_i) + b_k\,\cos(\omega_k x_i)$$

or introducing complex numbers with $I=\sqrt{-1}$, and Euler's Formula, $e^{Ix} = \cos(x) + I\,\sin(x)$

$$\begin{aligned} u(x_i) &= \sum_{k=-K}^{K} c_k e^{\omega_k x_i} \\ &= \sum_{k=-K}^{K} c_k(I\,\sin(\omega_k x_i) + \cos(\omega_k x_i)) \end{aligned}$$

where $c_k = \frac{b_k}{2} - I\,\frac{a_k}{2}$, $a_{-k} = -a_k$ and $b_{-k} = b_k$.

To show that the above equations for $u(x_i)$ are equivalent look at the sum of a $\pm k$ pair.

$$\begin{aligned} u_k + u_{-k} &= c_k(I\,\sin(\omega_k x_i) + \cos(\omega_k x_i)) \\ &\quad + c_{-k}(I\,\sin(-\omega_k x_i) + \cos(-\omega_k x_i)) \\ &= (\frac{b_k}{2} - I\,\frac{a_k}{2})(I\,\sin(\omega_k x_i) + \cos(\omega_k x_i)) \\ &\quad + (\frac{b_k}{2} + I\,\frac{a_k}{2})(-I\,\sin(\omega_k x_i) + \cos(\omega_k x_i)) \\ &= b_k\,\cos(\omega_k x_i) - I^2 a_k\,\sin(\omega_k x_i) + 0I \end{aligned}$$

which, with the addition of $c_0 = b_0$, results in the first equation.

Advection Stability

Burgers' Equation

Burgers' equation, $u_t + \frac{1}{2}(u^2)_x = \nu u_{xx}$ or $u_t + u\,u_x = \nu u_{xx}$ is a non-linear PDE. The wave speed is a function of the dependent variable, u. The greater u the faster the movement. An initial line with a negative slope (u decreasing with x) will steepen as time moves forward while an initial line with a positive slope will flatten as time moves forward.

With the finite difference option, the upwinding is implimented as

$u_i^{n+1} = u_i^n + \Delta t \left(\frac{u_{i-1}^2 - u_i^2}{2\Delta x} + \nu \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \right)$ - if $u_i > 0$,

$u_i^{n+1} = u_i^n + \Delta t \left(\frac{u_i^2 - u_{i+1}^2}{2\Delta x} + \nu \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \right)$ - if $u_i < 0$ and

$u_i^{n+1} = u_i^n + \Delta t \left(+ \nu \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2} \right)$ - if $u_i = 0$.

Applying the Taylor Series expansion to all terms for the case $u_i > 0$ and rearranging gives

$$u_t + uu_x - \nu u_{xx} = \frac{\Delta x}{2}((u_x)^2 + uu_{xx}) - \frac{\Delta t}{2}u_{tt} + ...HigherOrderTerms...$$

which again shows first order convergence.

Remarks

This Taylor Series was computed using sage, http::www.sagemath.org. Sage is a Python math program with symbolic calculus capabilities. The code follows

```
#Euler Explicit Burgers' Equation Finite Difference Form
var('x,t,Dx,Dt,nu')
U = function('U',x,t)
T = taylor(U,( x, 0),(t,0), 3)
S = taylor(U^2,(x, 0),(t,0), 3)
rem = (T.substitute(x=0,t=0) -T.substitute(x=0,t=Dt))/Dt - ( S.substitute(x=0,t
      =0) - S.substitute(x=-Dx,t=0))/Dx/2
rem += nu*(T.substitute(x=-Dx,t=0) - 2*T.substitute(x=0,t=0) + T.substitute(x=
      Dx,t=0))/Dx/Dx
print rem.expand()
```

The finite difference nulimit option is $\nu_{reduced} = max(0.0, \nu - \frac{|u_i|\Delta x}{2})$.

The finite volume form is implimented with upwinding from the flux through the faces (or mid points) based on the sign of $\overline{u_{i+1/2}} = \frac{u_i + u_{i+1}}{2}$.

$F_{i+1/2} = \overline{u_{i+1/2}} \frac{u_i}{2} + \nu \frac{u_i - u_{i+1}}{\Delta x}$ - if $\overline{u_{i+1/2}} > 0$,

$F_{i+1/2} = \overline{u_{i+1/2}} \frac{u_{i+1}}{2} + \nu \frac{u_i - u_{i+1}}{\Delta x}$ - if $\overline{u_{i+1/2}} < 0$ and

$F_{i+1/2} = \nu \frac{u_i - u_{i+1}}{\Delta x}$ - if $\overline{u_{i+1/2}} = 0$.

Then

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x}(F_{i-1/2} - F_{i+1/2})$$

is the full solution equation.

Applying the Taylor Series expansion to all terms for the case $\overline{u_{i+1/2}} > 0$ and $\overline{u_{i-1/2}} > 0$

$$u_t + uu_x - \nu u_{xx} = \frac{\Delta t}{2}u_{tt} + \frac{\Delta x}{4}((u_x)^2 + uu_{xx}) + ...HigherOrderTerms...$$

This indicates a slightly lower error for the finite volume form but still only first order convergence. The sage code for computing the Taylor Series expansions follows:

```
#Euler Explicit Burgers' Equation Finite Volume Form
var('x,t,Dx,Dt,nu')
U = function('U',x,t)
T = taylor(U,( x, 0),(t,0), 3)
ubm =( T.substitute(x=-Dx,t=0) + T.substitute(x=0,t=0))/2
ubp =( T.substitute(x=Dx,t=0) + T.substitute(x=0,t=0))/2
fp = ubp*T.substitute(x=0,t=0)/2 + nu*(T.substitute(x=0,t=0) - T.substitute(x=
      Dx,t=0))/Dx
fm = ubm*T.substitute(x=-Dx,t=0)/2 + nu*(T.substitute(x=-Dx,t=0) - T.substitute
      (x=0,t=0))/Dx
rem =  (T.substitute(x=0,t=0) -T.substitute(x=0,t=Dt))/Dt + ( fm - fp )/Dx
print rem.expand()
```

The finite volume nulimit option is $\nu_{reduced} = max(0.0, \nu - \frac{|\overline{u_{i+1/2}}|\Delta x}{2})$.

Summary

The Euler Explicit Method is the simplest method for numerical solution of ordinary or partial differential equations. However, it is not very accurate and has limitted stability. This method is good for demonstrating basic properties of numerical methods for partial differential equations, such as convergence and stability.

4.5.2 Member Function Documentation

4.5.2.1 bool EEWidget::canSolve (int *equ*) [virtual]

indicate whether equation number equ can be solved in this class

Reimplemented from SolvWidget.

4.5.2.2 void EEWidget::step (const size_t *nStep*) [virtual]

Compute nStep time steps using the current parameters

Parameters

nStep	the number of steps to computer

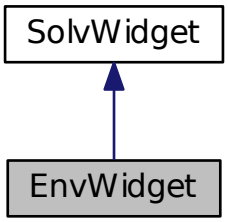
Implements SolvWidget.

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

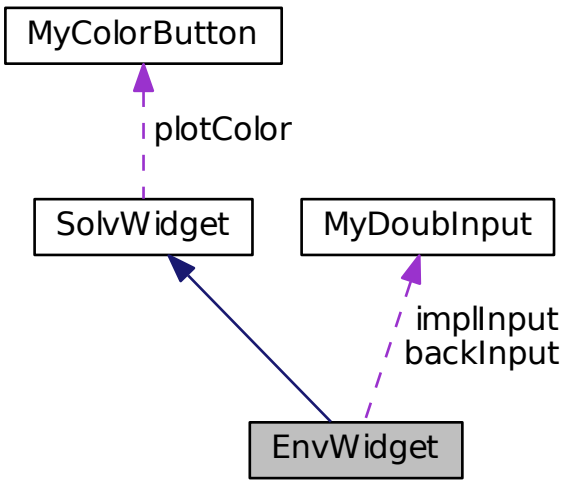
- eewidget.h
- eewidget.cpp

4.6 EnvWidget Class Reference

Inheritance diagram for EnvWidget:



Collaboration diagram for EnvWidget:



Public Slots

- void **setImplicit** (double value=5/12.0)
- void **setBackward** (double value=-1/12.0)
- void **setBasis** (int index)
- void **setMethod** (int index)

Public Member Functions

- **EnvWidget** (QWidget *parent=0)
- **EnvWidget** (const EnvWidget &other)
- virtual EnvWidget & **operator=** (const EnvWidget &other)
- virtual bool **operator==** (const EnvWidget &other) const
- virtual void **step** (const size_t nStep)
- virtual void **setSize** (const size_t size=100)
- virtual void **setCFL** (const double value=1.0)
- const double **getImplicit** ()
- virtual void **initSin** (const double value)
- virtual double * **getU** ()
- bool **canSolve** (int equ)

Protected Member Functions

- void **updateCoef** (int value)
- void **lspc** ()
- void **lspa** ()
- void **femc** ()
- void **fema** ()
- void **rk** ()
- void **setupTrans** (int size)
- void **allocate_gsl** (int size)

Protected Attributes

- QLabel * **implLabel**
- MyDoubInput * **implInput**
- QLabel * **backLabel**
- MyDoubInput * **backInput**
- QLabel * **weightLabel**
- QComboBox * **weightBox**
- QLabel * **methodLabel**
- QComboBox * **methodBox**
- QStandardItemModel * **methodModel**
- int **method**
- double **impl**
- double **beta**
- double **back**
- int **ibase**
- bool **dirty**
- int **ipad**
- int **winwid**
- int **winoff**
- double * **weights**
- size_t **nbas**
- int **ntime**
- double * **Ub**
- size_t **nub**

- double * **Usum**
- gsl_vector_view **UbView**
- gsl_vector * **TranVec**
- gsl_vector * **UVec**
- gsl_vector * **BVec**
- gsl_vector * **CVec**
- gsl_matrix * **Left**
- gsl_permutation * **lpermut**
- gsl_matrix * **Right**
- gsl_matrix * **FarRight**
- gsl_matrix * **Mforw**
- gsl_matrix * **Mback**
- gsl_permutation * **permut**
- int **signum**

Additional Inherited Members

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

- envwidget.h
- envwidget.cpp

4.7 ErrTabDock Class Reference

Public Member Functions

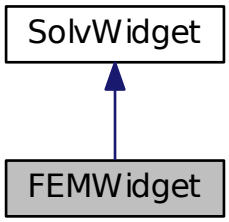
- **ErrTabDock** (const QString &title, QWidget *parent=0, Qt::WindowFlags flags=Qt::Widget)
- **ErrTabDock** (QWidget *parent=0, Qt::WindowFlags flags=Qt::Widget)
- void **errTab** ()

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

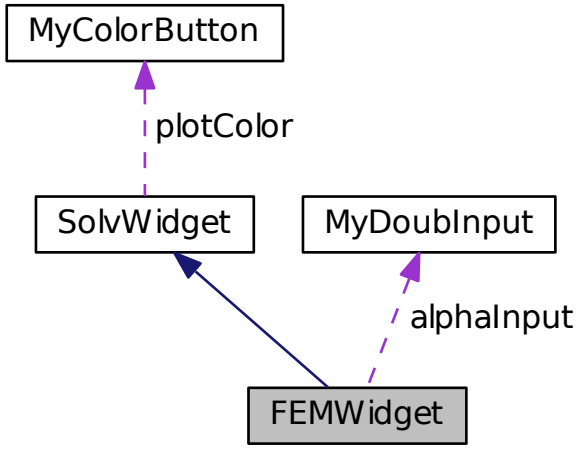
- errtabdock.h
- errtabdock.cpp

4.8 FEMWidget Class Reference

Inheritance diagram for FEMWidget:



Collaboration diagram for FEMWidget:



Public Slots

- void **setImplicit** (double value=0.5)
- void **setBasis** (int index)

Public Member Functions

- **FEMWidget** (QWidget ∗parent=0)

- **FEMWidget** (const FEMWidget &other)
- virtual FEMWidget & **operator=** (const FEMWidget &other)
- virtual bool **operator==** (const FEMWidget &other) const
- virtual void **step** (const size_t nStep)
- virtual void **setSize** (const size_t size=100)
- const double **getImplicit** ()
- bool **canSolve** (int equ)

Protected Attributes

- QLabel * **alphaLabel**
- MyDoubInput * **alphaInput**
- QLabel * **weightLabel**
- QComboBox * **weightBox**
- double **a**
- double **b**
- bool **cosBas**
- gsl_vector * **X**
- gsl_vector * **DIAG**
- gsl_vector * **E**
- gsl_vector * **F**
- gsl_vector * **B**

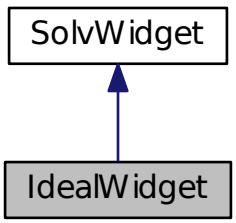
Additional Inherited Members

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

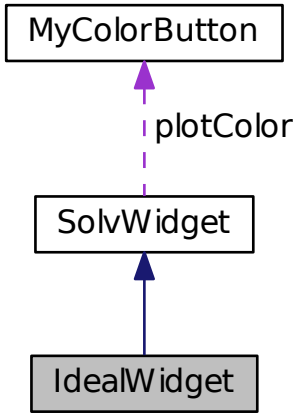
- femwidget.h
- femwidget.cpp

4.9 IdealWidget Class Reference

Inheritance diagram for IdealWidget:



Collaboration diagram for IdealWidget:



Public Member Functions

- **IdealWidget** (QWidget ∗parent=0)
- virtual void **step** (const size_t nStep)
- virtual void **setSize** (const size_t size=100)
- virtual void **initSin** (const double cycles=1.0)
- virtual double ∗ **getU** ()
- virtual void **setEquation** (int index)
- virtual bool **canSolve** (int equ)
- void **toMagPhase** (double ∗FFT)
- void **Phaser** (double ∗FFT)

Protected Attributes

- double ∗ **FFT_0**
- gsl_fft_real_wavetable ∗ **real_g**
- gsl_fft_halfcomplex_wavetable ∗ **hc_g**
- gsl_fft_real_workspace ∗ **work_g**
- double **totVisc**

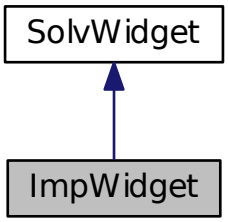
Additional Inherited Members

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

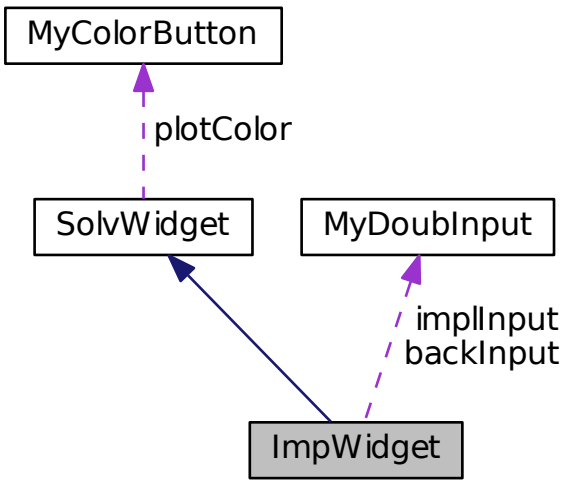
- idealwidget.h
- idealwidget.cpp

4.10 ImpWidget Class Reference

Inheritance diagram for ImpWidget:



Collaboration diagram for ImpWidget:



Public Slots

- void **setImplicit** (double value=5/12.0)
- void **setBackward** (double value=-1/12.0)
- void **setBasis** (int index)
- void **setMethod** (int index)

Public Member Functions

- **ImpWidget** (QWidget *parent=0)
- **ImpWidget** (const ImpWidget &other)
- virtual ~ImpWidget ()
- virtual ImpWidget & **operator=** (const ImpWidget &other)
- virtual bool **operator==** (const ImpWidget &other) const
- virtual void step (const size_t nStep)
- virtual void setSize (const size_t size=100)
- virtual void **setCFL** (const double value=1.0)
- const double **getImplicit** ()
- virtual void **initSin** (const double value)
- virtual double * **getU** ()
- bool **canSolve** (int equ)

Protected Member Functions

- void **updateCoef** (int value)
- void **fillA** ()
- void **fillB** ()
- void **blockFillA** ()
- void **lspc** ()
- void **lspa** ()
- void **femc** ()
- void **fema** ()
- void **reversTrans** (double *from, double *to)
- void **forwardTrans** (double *from, double *to)
- void **setupTrans** ()

Protected Attributes

- QLabel * **implLabel**
- MyDoubInput * **implInput**
- QLabel * **backLabel**
- MyDoubInput * **backInput**
- QLabel * **weightLabel**
- QComboBox * **weightBox**
- QLabel * **methodLabel**
- QComboBox * **methodBox**
- int **method**
- double * **coef**
- int **ncoef**
- int **nblock**
- int **ivar**
- int **nvar**
- int **ntime**
- int **nup**
- int **ndn**
- double **impl**
- double **beta**

- double **back**
- int **ibase**
- double * **Ub**
- bool **aexist**
- bool **dirty**
- char **equed** [1]
- yes_no_t **equil**
- trans_t **trans**
- SuperMatrix **A**
- SuperMatrix **L**
- SuperMatrix **Up**
- SuperMatrix **B**
- SuperMatrix **X**
- double * **a**
- int * **asub**
- int * **xa**
- int * **perm_c**
- int * **perm_r**
- int * **etree**
- void * **work**
- int **info**
- int **lwork**
- int **nrhs**
- int **i**
- int **m**
- int **n**
- int **nnz**
- double * **rhsb**
- double * **rhsx**
- double * **R**
- double * **C**
- double * **ferr**
- double * **berr**
- double **u**
- double **rpg**
- double **rcond**
- mem_usage_t **mem_usage**
- superlu_options_t **options**
- SuperLUStat_t **stat**
- bool **transform**
- double * **Utran**
- gsl_vector * **TranVec**
- gsl_vector * **UVec**
- gsl_matrix * **Mforw**
- gsl_matrix * **Mback**
- int **matSize_gsl**
- gsl_permutation * **permut**
- int **signum**

Additional Inherited Members

4.10.1 Constructor & Destructor Documentation

4.10.1.1 ImpWidget::~ImpWidget ( ) [virtual]

???

??? Destroy_CompCol_Matrix(&A);

4.10.2 Member Function Documentation

4.10.2.1 void ImpWidget::setSize (const size_t *size* = 100) [virtual]

???

??? Destroy_CompCol_Matrix(&A);

Reimplemented from SolvWidget.

4.10.2.2 void ImpWidget::step (const size_t *nStep*) [virtual]

set B matrix

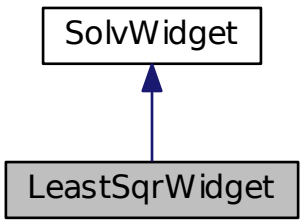
Implements SolvWidget.

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

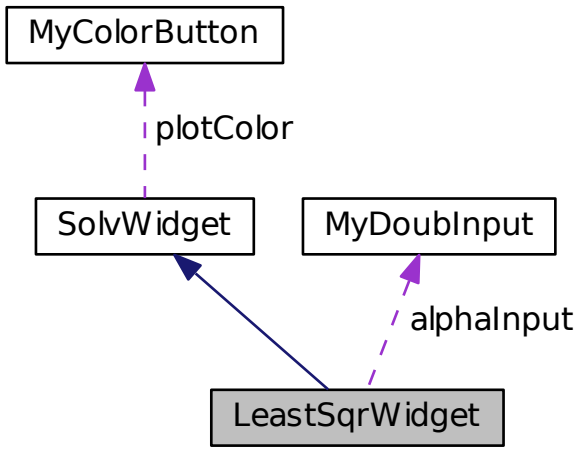
- impwidget.h
- impwidget.cpp

4.11 LeastSqrWidget Class Reference

Inheritance diagram for LeastSqrWidget:



Collaboration diagram for LeastSqrWidget:



Public Slots

- void **setImplicit** (double value=0.5)
- void **setBasis** (int index)

Public Member Functions

- **LeastSqrWidget** (QWidget ∗parent=0)
- **LeastSqrWidget** (const LeastSqrWidget &other)
- virtual LeastSqrWidget & **operator=** (const LeastSqrWidget &other)
- virtual bool **operator==** (const LeastSqrWidget &other) const
- virtual void **step** (size_t nStep=1)
- virtual void **setSize** (const size_t value)
- const double **getImplicit** ()
- bool **canSolve** (int equ)

Protected Attributes

- QLabel ∗ **alphaLabel**
- MyDoubInput ∗ **alphaInput**
- QLabel ∗ **weightLabel**
- QComboBox ∗ **weightBox**
- double **a**
- double **b**
- bool **cosBas**
- gsl_vector ∗ **X**

- gsl_vector * **DIAG**
- gsl_vector * **E**
- gsl_vector * **F**
- gsl_vector * **B**

Additional Inherited Members

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

- leastsqrwidget.h
- leastsqrwidget.cpp

4.12 LineWidthDelegate Class Reference

Public Member Functions

- **LineWidthDelegate** (QObject *parent=0)
- virtual QWidget * **createEditor** (QWidget *parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- virtual void **setEditorData** (QWidget *editor, const QModelIndex &index) const
- virtual void **setModelData** (QWidget *editor, QAbstractItemModel *model, const QModelIndex &index) const

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

- itemdelegate.h
- itemdelegate.cpp

4.13 MyColorButton Class Reference

Public Slots

- void **setColor** ()
- void **setValue** (QColor value)

Signals

- void **valueChanged** (QColor value)

Public Member Functions

- **MyColorButton** (QWidget *parent=0)
- **MyColorButton** (const QColor &, QWidget *parent=0)
- QColor **getValue** ()

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

- myinputs.h
- myinputs.cpp

4.14 MyDoubInput Class Reference

Public Slots

- void **done** ()
- void **setValue** (double value)

Signals

- void **valueChanged** (double value)

Public Member Functions

- **MyDoubInput** (QWidget ∗parent=0)
- **MyDoubInput** (const QString &, QWidget ∗parent=0)
- **MyDoubInput** (double value, QWidget ∗parent=0, double lower=-1.7e+308, double upper=1.7e+308, double singleStep=0.01, int precision=6)
- double **getValue** ()

Public Attributes

- QDoubleValidator ∗ **val**

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

- myinputs.h
- myinputs.cpp

4.15 MyIntInput Class Reference

Public Slots

- void **done** ()
- void **setValue** (int value)

Signals

- void **valueChanged** (int value)

Public Member Functions

- **MyIntInput** (QWidget ∗parent=0)
- **MyIntInput** (const QString &value, QWidget ∗parent=0)
- int **getValue** ()

Public Attributes

- QIntValidator * **val**
- int **myValue**

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

- myinputs.h
- myinputs.cpp

4.16 NV Struct Reference

Public Attributes

- QString **name**
- QwtSymbol::Style **val**

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

- itemdelegate.h

4.17 pde1d Class Reference

Public Slots

- void **setSize** (int ivalue)
- void **setCycles** (double value=1.0)
- void **reset** ()
- void **setCFL** (double value=1.0)
- void **setEquation** (int value=0)
- void **setViscosity** (double value=0.001)
- void **setStop** (int value)
- void **setStep** (int value)
- void **setDelay** (int value=10)
- void **run** ()
- void **stopRun** ()
- void **saveImage** ()
- void **savePlot** ()
- void **addSolver** (int index=0)
- void **removeSolver** (int index=0)
- void **refresh** ()

Public Member Functions

- pde1d ()

4.17.1 Constructor & Destructor Documentation

4.17.1.1 pde1d::pde1d ( )

Main window with a qwt plot area and a control dock widget

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

- pde1d.h
- pde1d.cpp

4.18 PenStyleDelegate Class Reference

Public Member Functions

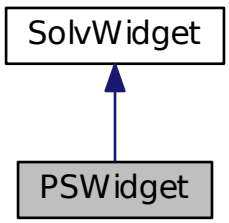
- **PenStyleDelegate** (QObject ∗parent=0)
- virtual QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- virtual void **setEditorData** (QWidget ∗editor, const QModelIndex &index) const
- virtual void **setModelData** (QWidget ∗editor, QAbstractItemModel ∗model, const QModelIndex &index) const

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

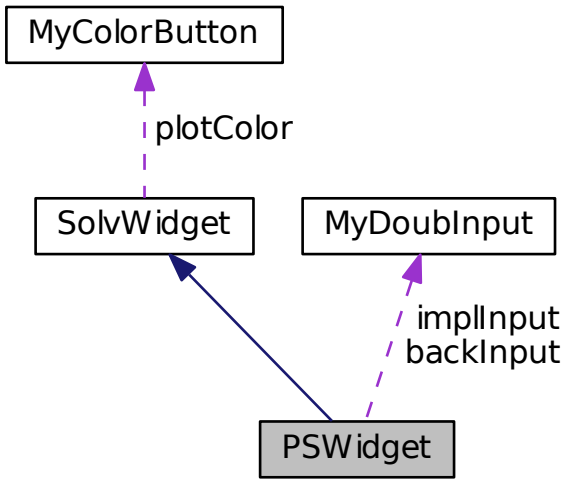
- itemdelegate.h
- itemdelegate.cpp

4.19 PSWidget Class Reference

Inheritance diagram for PSWidget:



Collaboration diagram for PSWidget:



Public Slots

- void **setImplicit** (double value=5/12.0)
- void **setBackward** (double value=-1/12.0)
- void **setMethod** (int index)

Public Member Functions

- **PSWidget** (QWidget ∗parent=0)
- **PSWidget** (const PSWidget &other)
- virtual PSWidget & **operator=** (const PSWidget &other)
- virtual bool **operator==** (const PSWidget &other) const
- virtual void **step** (const size_t nStep)
- virtual void setSize (const size_t size=100)
- virtual void **setCFL** (const double value=1.0)
- const double **getImplicit** ()
- virtual void **initSin** (const double value)
- virtual double ∗ **getU** ()
- void **setNStage** (int arg1)
- virtual void **setEquation** (int index)
- bool **canSolve** (int equ)

Protected Member Functions

- void **allocateData** (size_t rksize)
- void **freeData** ()
- void **lspc** ()
- void **lspa** ()
- void **femc** ()
- void **fema** ()
- void **rk** ()
- void **setupTrans** ()
- void **phaser** ()
- void **solvModel** (QString tr, QStandardItem *arg2)

Protected Attributes

- QLabel * **implLabel**
- MyDoubInput * **implInput**
- QLabel * **backLabel**
- MyDoubInput * **backInput**
- QLabel * **weightLabel**
- QComboBox * **weightBox**
- QLabel * **methodLabel**
- QComboBox * **methodBox**
- QLabel * **pdeLabel**
- QComboBox * **pdeBox**
- QStandardItemModel * **methodModel**
- int **method**
- double **impl**
- double **beta**
- double **back**
- double * **Ideal**
- int **ntime**
- gsl_fft_real_wavetable * **real_g**
- gsl_fft_halfcomplex_wavetable * **hc_g**
- gsl_fft_real_workspace * **work_g**
- double ** **data**
- size_t **narrays**
- size_t **nrk**
- size_t **dsize**
- double * **b_a**
- double * **b_b**
- double * **b_c**
- int **nStage**
- int **n_b_k**

Additional Inherited Members

4.19.1 Member Function Documentation

4.19.1.1 void PSWidget::setSize (const size_t *size* = 100) [virtual]

???

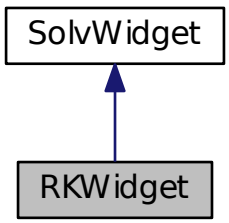
Reimplemented from SolvWidget.

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

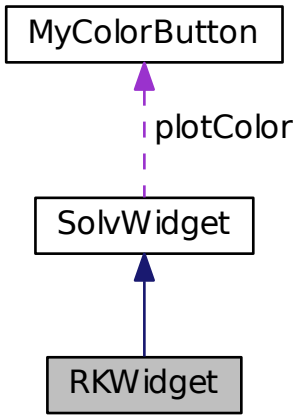
- pswidget.h
- pswidget.cpp

4.20 RKWidget Class Reference

Inheritance diagram for RKWidget:



Collaboration diagram for RKWidget:



Public Slots

- void setBasis (int index)
- void **setMethod** (int index)

Public Member Functions

- **RKWidget** (QWidget ∗parent=0)
- **RKWidget** (const RKWidget &other)
- virtual ∼RKWidget ()
- virtual RKWidget & **operator=** (const RKWidget &other)
- virtual bool **operator==** (const RKWidget &other) const
- virtual void step (const size_t nStep)
- virtual void **setSize** (const size_t size=100)
- virtual void **setCFL** (const double value=1.0)
- const double **getImplicit** ()
- bool **canSolve** (int equ)

Protected Member Functions

- void **updateCoef** (int value)
- void **fillA** ()
- void **fillB** (int stage)
- void **blockFillA** ()
- void **setNStage** (int arg1)

Protected Attributes

- QLabel * **weightLabel**
- QComboBox * **weightBox**
- QLabel * **methodLabel**
- QComboBox * **methodBox**
- int **method**
- double * **coef**
- int **ncoef**
- int **nblock**
- int **ivar**
- int **nvar**
- int **nup**
- int **ndn**
- double * **b_a**
- double * **b_b**
- double * **b_c**
- double * **b_k**
- int **nStage**
- int **n_b_k**
- int **ibase**
- bool **aexist**
- char **equed** [1]
- yes_no_t **equil**
- trans_t **trans**
- SuperMatrix **A**
- SuperMatrix **L**
- SuperMatrix **Up**
- SuperMatrix **B**
- SuperMatrix **X**
- double * **a**
- int * **asub**
- int * **xa**
- int * **perm_c**
- int * **perm_r**
- int * **etree**
- void * **work**
- int **info**
- int **lwork**
- int **nrhs**
- int **i**
- int **m**
- int **n**
- int **nnz**
- double * **rhsb**
- double * **rhsx**
- double * **R**
- double * **C**
- double * **ferr**
- double * **berr**
- double **u**

- double **rpg**
- double **rcond**
- mem_usage_t **mem_usage**
- superlu_options_t **options**
- SuperLUStat_t **stat**

Additional Inherited Members

4.20.1 Constructor & Destructor Documentation

4.20.1.1 RKWidget::~RKWidget ( ) [virtual]

???

??? Destroy_CompCol_Matrix(&A);

4.20.2 Member Function Documentation

4.20.2.1 void RKWidget::setBasis (int *index*) [slot]

todo - setupTrans() and friends – error wrong approach – new envwidget

4.20.2.2 void RKWidget::step (const size_t *nStep*) [virtual]

set B matrix

solve factored system

update U_

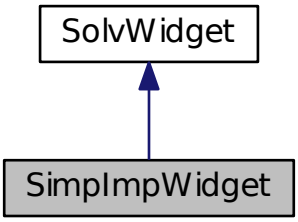
Implements SolvWidget.

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

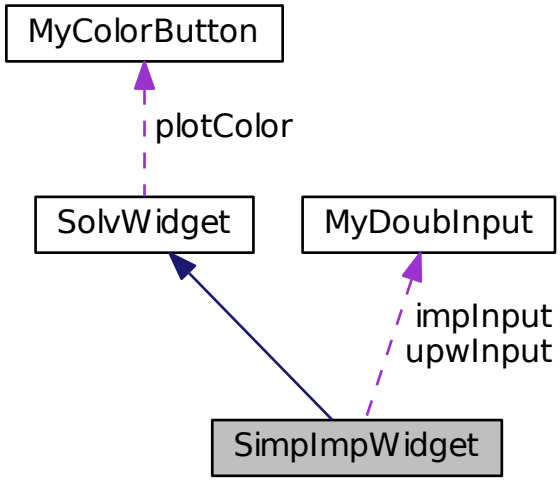
- rkwidget.h
- rkwidget.cpp

4.21 SimpImpWidget Class Reference

Inheritance diagram for SimpImpWidget:



Collaboration diagram for SimpImpWidget:



Public Slots

- void **setImpl** (double value=0.0)
- void **setUpwind** (double value=0.5)

Public Member Functions

- **SimpImpWidget** (QWidget ∗parent=0)
- **SimpImpWidget** (const SimpImpWidget &other)
- virtual SimpImpWidget & **operator=** (const SimpImpWidget &other)
- virtual bool **operator==** (const SimpImpWidget &other) const
- double **getUpwind** ()
- double **getImpl** ()
- virtual void **step** (size_t nStep=1)
- virtual void **setSize** (const size_t value)
- bool **canSolve** (int equ)

Protected Attributes

- double **impl**
- double **upwind**
- QLabel ∗ **implLabel**
- MyDoubInput ∗ **implnput**
- QLabel ∗ **upwLabel**
- MyDoubInput ∗ **upwlnput**
- gsl_vector ∗ **X**
- gsl_vector ∗ **DIAG**
- gsl_vector ∗ **E**
- gsl_vector ∗ **F**
- gsl_vector ∗ **B**

Additional Inherited Members

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

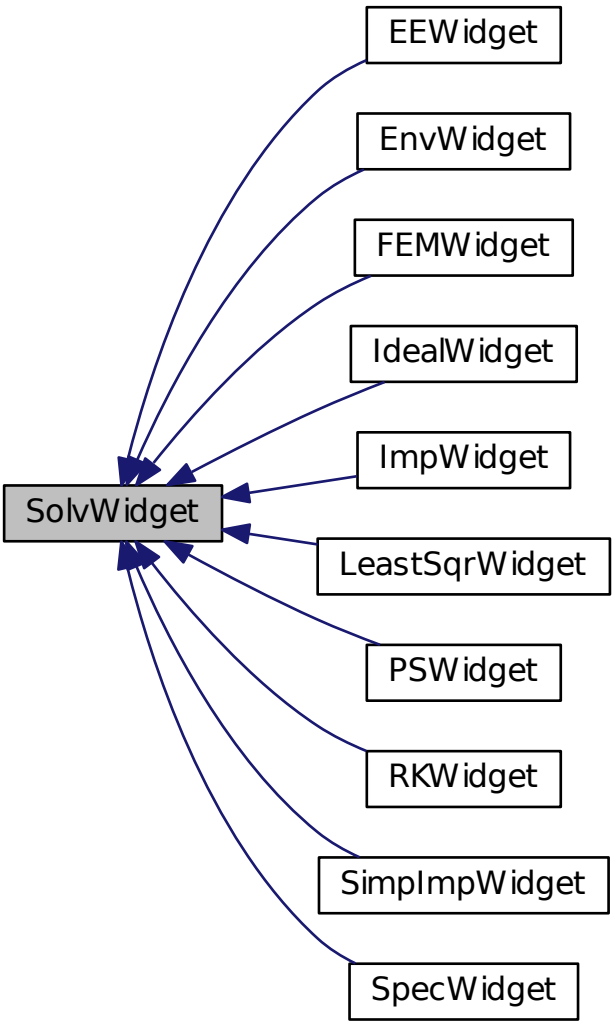
- simpimpwidget.h
- simpimpwidget.cpp

4.22 SolvWidget Class Reference

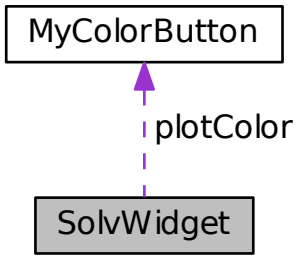
Base widget to be inherited to add a numerical solver.

`#include <solvwidget.h>`

Inheritance diagram for SolvWidget:



Collaboration diagram for SolvWidget:



Public Slots

- void **setColor** (QColor color)
- void **setTitle** (QString title)
- void **setViscosity** (double value=0.0)

Signals

- void **dockClose** (int id)

Public Member Functions

- **SolvWidget** (QWidget ∗parent=0)
- **SolvWidget** (const SolvWidget &other)
- virtual SolvWidget & **operator=** (const SolvWidget &other)
- virtual bool **operator==** (const SolvWidget &other) const
- QString & **getTitle** ()
- QwtPlotCurve ∗ **getCurve** (QString xvalue=tr("X"), QString value=tr("U"))
- QColor **getPenColor** ()
- const size_t **getSize** ()
- void **resize** (int value)
- virtual void **initSin** (const double cycles=1.0)
- virtual void **setCFL** (const double value=1.0)
- const double **getCFL** ()
- void **setSpeed** (const double value=1.0)
- const double **getSpeed** ()
- double ∗ **getX** ()
- virtual double ∗ **getU** ()
- double ∗ **getIdeal** ()
- void **setup** (const size_t size=100, const double cycles=1.0, const double cfl=1.0)
- int **getCurrentStep** ()
- double **getCycles** ()

- double **getTravel** ()
- virtual bool **canSolve** (int equ)
- bool **isUnstable** ()
- bool **isOK** ()
- int **getId** ()
- void **setId** (const int value)
- void **closeEvent** (QCloseEvent ∗ev)
- virtual void **setSize** (const size_t size=100)
- virtual void **step** (const size_t nStep)=0
- void **setupUi** ()
- virtual void **setEquation** (int index)

Public Attributes

- QWidget ∗ **dockWidgetContents**
- QVBoxLayout ∗ **verticalLayout**
- QLabel ∗ **plotNameLabel**
- QLineEdit ∗ **plotNameEdit**
- QLabel ∗ **plotColorLabel**
- MyColorButton ∗ **plotColor**
- QSpacerItem ∗ **verticalSpacer**
- QwtPlotCurve ∗ **curve**

Protected Member Functions

- void **Efunc** (double ∗Udat)
- void **Dfunc** (double ∗Ddat)

Protected Attributes

- QString **title**
- QHash< QString, double ∗ > **data**
- QStringList **dataNames**
- QHash< QString, double ∗ > **fluxes**
- QStringList **fluxNames**
- size_t **N_**
- double ∗ **U_**
- double ∗ **X_**
- double ∗ E_

  U_t + e_∗E_(U)_x + d_∗D_(U)_xx = 0.0.
- double ∗ **J_**
- double ∗ **D_**
- double ∗ **Ideal_**
- double **d_**
- double **e_**
- double **visc_**
- double **speed_**
- double **dt**
- double **dx**

- double **CFL**
- double **cycles**
- int **cStep**
- double **pi**
- double **totCFL**
- int **id**
- bool **dirty**
- int **equation**
- bool **unstable**
- bool **samset**
- int **nghost**
- int **nfghost**

4.22.1 Detailed Description

Base widget to be inherited to add a numerical solver.

This base class provides much of the structure for interaction with the remainder of the code. To add a new solver inherit from this class and override the functions.

```
void step(const size_t nStep ) {  -- to solve for nStep times }
bool canSolve(int equ); // indicates the solver can solve equation number equ
```

The constructor should add variable names to dataNames for data at x_i nodes and fluxNames for data at midpoints, $x_i - 1/2$. e.g.

```
dataNames.append(tr("MyVariableName"));
```

This will make the variable double pointer available as

```
double *myVariable;
myVariable = data.value(tr("MyVariableName"));
```

Todo make all variables accessible for plotting by name

default names include "X" // locations "U" // default dependant variable "E" // convective flux - e.g. c∗U or 1/2 U^2 "D" // diffusive flux - e.g. nu∗U "Jac" // Jacobian dE/dU

To add custom widgets/controls for the solver, create QWidget items, connect the signals to slots and add them to verticalLayout within the constructor.

```
    upwLabel = new QLabel(tr("Upwinding"));
    upwInput = new MyDoubInput(0.0,this,-0.1,1.1,0.01,6);// (Initial
       Value, this, minValue, maxValue, increment, precision)
    connect(upwInput,SIGNAL(valueChanged(double)),this,SLOT(setUpwind(double)))
      ;
    verticalLayout->addWidget(upwLabel);
    verticalLayout->addWidget(upwInput);
    .
    .
    .
    verticalLayout->addItem(verticalSpacer);
```

Also, initialize any custom parameters in the constructor and set the initial values in any custom widgets.

For additional data that does not fit the form as a double array (double pointer) override setSize(int value) to allocate the custom data items.

Note

at the end of the setSize() method make sure to call resize(int value) to perform the automatic size updates. resize() will allocate all named variables in dataNames and fluxNames and reset the array size variable N_.

Override the destructor to delete any custom data types.

The new solver must currently be added to pde1d.cpp Include the new headers

```
#include "specwidget.h"
#include "pswidget.h"
```

in the constructor add the new solvers to the combobox, e.g.

```
solvModel->appendRow( new QStandardItem ( tr("Spectral FFT") ));    //
   addSolver(8)
solvModel->appendRow( new QStandardItem ( tr("Pseudo Spectral") )); //
   addSolver(9)
```

and in addSolver (int index) add the additionnal case, e.g.

```
case 8:
    SpecWidget *specw;
    specw= new SpecWidget ( this );
    addIt(specw);
    break;
case 9:
    PSWidget *psw;
    psw = new PSWidget ( this );
    addIt(psw);
    break;
```

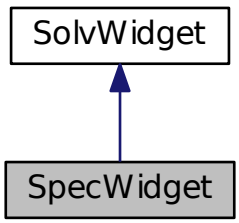
Todo make the new solver a shared object and load it with dlopen so pde1d does not need to be modified.

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

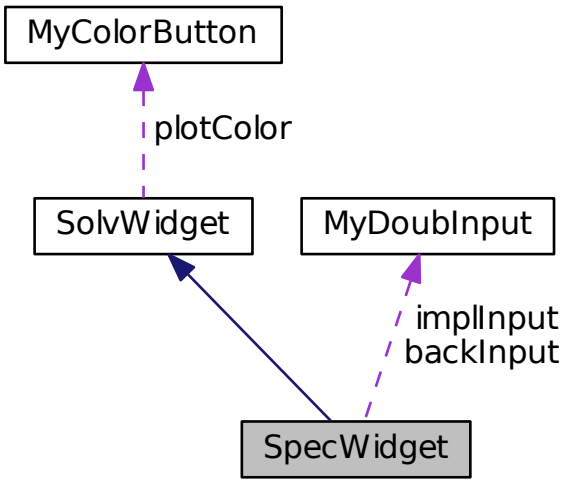
- solvwidget.h
- solvwidget.cpp

4.23 SpecWidget Class Reference

Inheritance diagram for SpecWidget:



Collaboration diagram for SpecWidget:



Public Slots

- void **setImplicit** (double value=5/12.0)
- void **setBackward** (double value=-1/12.0)
- void **setMethod** (int index)

Public Member Functions

- **SpecWidget** (QWidget ∗parent=0)
- **SpecWidget** (const SpecWidget &other)
- virtual SpecWidget & **operator=** (const SpecWidget &other)
- virtual bool **operator==** (const SpecWidget &other) const
- virtual void **step** (const size_t nStep)
- virtual void **setSize** (const size_t size=100)
- virtual void **setCFL** (const double value=1.0)
- const double **getImplicit** ()
- virtual void **initSin** (const double value)
- virtual double ∗ **getU** ()
- void **setNStage** (int arg1)
- virtual void **setEquation** (int index)
- bool **canSolve** (int equ)

Protected Member Functions

- void **allocateData** (size_t rksize)
- void **freeData** ()
- void **lspc** ()
- void **lspa** ()
- void **femc** ()
- void **fema** ()
- void **rk** (double *out, double *in)
- void **setupTrans** ()
- void phaser ()

Protected Attributes

- QLabel * **implLabel**
- MyDoubInput * **implInput**
- QLabel * **backLabel**
- MyDoubInput * **backInput**
- QLabel * **weightLabel**
- QComboBox * **weightBox**
- QLabel * **methodLabel**
- QComboBox * **methodBox**
- QStandardItemModel * **methodModel**
- int **method**
- double **impl**
- double **beta**
- double **back**
- int **ntime**
- gsl_fft_real_wavetable * **real_g**
- gsl_fft_halfcomplex_wavetable * **hc_g**
- gsl_fft_real_workspace * **work_g**
- double ** **data**
- size_t **narrays**
- size_t **nrk**
- size_t **dsize**
- double * **b_a**
- double * **b_b**
- double * **b_c**
- double * **b_k**
- int **nStage**
- int **n_b_k**

Additional Inherited Members

4.23.1 Member Function Documentation

4.23.1.1 void SpecWidget::phaser ( ) [protected]

??? positive rotation

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

- specwidget.h
- specwidget.cpp

4.24 SymbolSizeDelegate Class Reference

Public Member Functions

- **SymbolSizeDelegate** (QObject ∗parent=0)
- virtual QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- virtual void **setEditorData** (QWidget ∗editor, const QModelIndex &index) const
- virtual void **setModelData** (QWidget ∗editor, QAbstractItemModel ∗model, const QModelIndex &index) const

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

- itemdelegate.h
- itemdelegate.cpp

4.25 SymbolStyleDelegate Class Reference

Public Member Functions

- **SymbolStyleDelegate** (QObject ∗parent=0)
- virtual QWidget ∗ **createEditor** (QWidget ∗parent, const QStyleOptionViewItem &option, const QModelIndex &index) const
- virtual void **setEditorData** (QWidget ∗editor, const QModelIndex &index) const
- virtual void **setModelData** (QWidget ∗editor, QAbstractItemModel ∗model, const QModelIndex &index) const

Static Public Member Functions

- static bool **initSyms** ()
- static QString **symName** (QwtSymbol::Style style)
- static const QIcon **getIcon** (QwtSymbol &sym)

Static Public Attributes

- static QList< NV > **syms**
- static bool **sinit** = SymbolStyleDelegate::initSyms()

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

- itemdelegate.h
- itemdelegate.cpp

Index