HighOrderDGTransportSolver

Generated by Doxygen 1.8.14

Contents

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

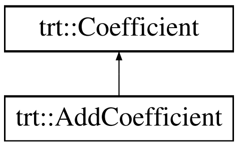
Class Documentation

3.1 trt::AddCoefficient Class Reference

add two coefficients together

```
#include <Coefficient.hpp>
```

Inheritance diagram for trt::AddCoefficient:



Public Member Functions

- AddCoefficient (Coefficient ∗c1, Coefficient ∗c2)

  constructor. adds c1 and c2
- double Eval (ElTrans &trans, double xref) const

  evaluate

Private Attributes

- Coefficient ∗ **_c1**
- Coefficient ∗ **_c2**

Additional Inherited Members

3.1.1 Detailed Description

add two coefficients together

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

- /home/sam/trt/fem/Coefficient.hpp

3.2 trt::Array< T > Class Template Reference

Template class for wrapping std::vector. Mostly provides out of range checks.

```
#include <Array.hpp>
```

Public Member Functions

- Array ()
  default constructor
- Array (int N)
  initialize
- Array (int N, T val)
  initialize and set values
- void operator= (const Array< T > &a)
  copy assignment
- Array (std::initializer_list< T > list)
  construct from initializer list
- void operator= (std::initializer_list< T > list)
  set from initializer list
- int Size () const
  return the size of the array
- void Resize (int N)
  resize the array
- T & operator[ ] (int ind)
  access to the array
- const T & operator[ ] (int ind) const
  const access to the array
- void operator= (T val)
  set all elements to val
- void Append (T val)
  add to end of Array
- void Append (const Array< T > &a)
  add an array to the back of this
- void Clear ()
  clear contents of vector
- void Intersection (const Array< T > &x, Array< T > &r) const
  return the intersection of two arrays
- bool operator== (const Array< T > &a) const
  test if two arrays are the same
- void Transpose ()
  reverse order of array
- void Print (std::ostream &out=std::cout) const
  print the Array
- double ∗ Data ()
  direct access to the data
- const double ∗ Data () const
  const direct access to the data
- double & Last ()
  access to the last element

Private Attributes

- std::vector< T > _vector

  vector that stores all the data

3.2.1 Detailed Description

template<class T = int>
class trt::Array< T >

Template class for wrapping std::vector. Mostly provides out of range checks.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Array() [1/2]

```
template<class T = int>
trt::Array< T >::Array (
              int N )  [inline]
```

initialize

Parameters

N	size of array

3.2.2.2 Array() [2/2]

```
template<class T = int>
trt::Array< T >::Array (
              int N,
              T val )  [inline]
```

initialize and set values

Parameters

N	size of array
val	initial value

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

- /home/sam/trt/general/Array.hpp

3.3 trt::Basis Class Reference

represent a collection of basis functions (in reference space) of an arbitrary polynomial order

```
#include <Basis.hpp>
```

Public Member Functions

- Basis (int order)

  constructor
- const Poly1D & operator[ ] (int i) const

  access to ith basis function
- const Poly1D & Derivative (int i) const

  access to ith derivative
- int Size () const

  return the number of basis functions

Private Attributes

- int _order

  polynomial order
- Array< Poly1D > _p

  polynomials
- Array< Poly1D > _dp

  derivatives of polynomials

3.3.1 Detailed Description

represent a collection of basis functions (in reference space) of an arbitrary polynomial order

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Basis()

```
trt::Basis::Basis (
            int order )
```

constructor

Parameters

order	polynomial order

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

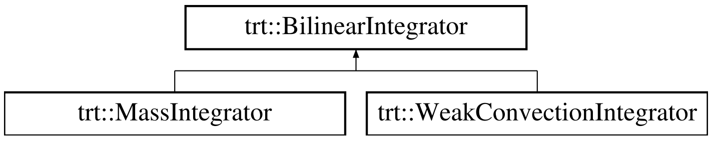
- /home/sam/trt/fem/Basis.hpp
- /home/sam/trt/fem/Basis.cpp

3.4 trt::BilinearIntegrator Class Reference

abstract class for bilinear forms

`#include <BilinearIntegrator.hpp>`

Inheritance diagram for trt::BilinearIntegrator:



Public Member Functions

- BilinearIntegrator ()

    default constructor
- virtual void Assemble (Element &el, Matrix &elmat)

    assemble a local matrix

3.4.1 Detailed Description

abstract class for bilinear forms

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

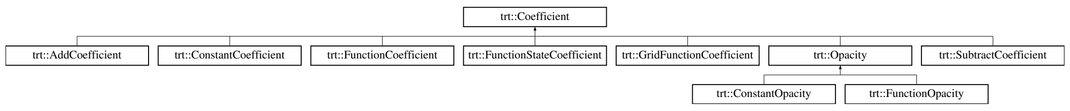
- /home/sam/trt/fem/BilinearIntegrator.hpp

3.5 trt::Coefficient Class Reference

abstract class for evaluating things

`#include <Coefficient.hpp>`

Inheritance diagram for trt::Coefficient:



Public Member Functions

- virtual double Eval (ElTrans &trans, double xref) const

  interface for evaluating with a transformation
- void SetState (double state)

  set the state

Protected Attributes

- double _state

  store a constant parameter that can be used in evaluating 2D functions

3.5.1 Detailed Description

abstract class for evaluating things

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

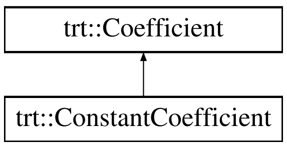
- /home/sam/trt/fem/Coefficient.hpp

3.6 trt::ConstantCoefficient Class Reference

evaluates to a constant value

```
#include <Coefficient.hpp>
```

Inheritance diagram for trt::ConstantCoefficient:



Public Member Functions

- ConstantCoefficient (double c)

  constructor
- double Eval (ElTrans &trans, double xref) const

  evaluate

Private Attributes

- double _c

  constant value

Additional Inherited Members

3.6.1 Detailed Description

evaluates to a constant value

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

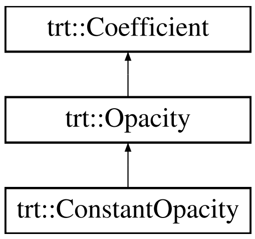
- /home/sam/trt/fem/Coefficient.hpp

3.7 trt::ConstantOpacity Class Reference

constant opacity

```
#include <Opacity.hpp>
```

Inheritance diagram for trt::ConstantOpacity:



Public Member Functions

- ConstantOpacity (double c)
  constructor
- double Eval (ElTrans &trans, double xref) const
  evaluate

Private Attributes

- double _c
  constant value

Additional Inherited Members

3.7.1 Detailed Description

constant opacity

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

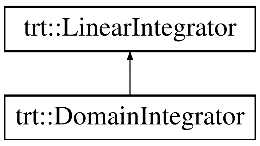
- /home/sam/trt/trt/Opacity.hpp

3.8 trt::DomainIntegrator Class Reference

domain integrator $B_i q dx$

```
#include <LinearIntegrator.hpp>
```

Inheritance diagram for trt::DomainIntegrator:



Public Member Functions

- DomainIntegrator ()

  constructor
- DomainIntegrator (Coefficient ∗c)

  construct with coefficient
- void Assemble (Element &el, Vector &elvec)

  assemble

Private Attributes

- Coefficient ∗ _c

  store the function to integrate
- Vector _shape

  store shape function evaluations

3.8.1 Detailed Description

domain integrator $B_i q dx$

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

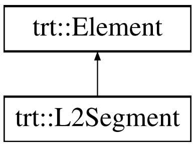
- /home/sam/trt/fem/LinearIntegrator.hpp
- /home/sam/trt/fem/LinearIntegrator.cpp

3.9 trt::Element Class Reference

abstract class for elements

```
#include <Element.hpp>
```

Inheritance diagram for trt::Element:



Public Member Functions

- Element (Node left, Node right, int order)

  constructor
- ∼Element ()

  destructor
- virtual void CalcShape (double x, Vector &shape) const

  evaluate all basis functions at an integration point
- virtual void CalcGradShape (double x, Vector &shape) const

  evaluate derivative of all basis functions at an integration point
- void CalcPhysGradShape (double x, Vector &shape) const

  evaluate derivatives in physical space (divide by Jacobian)
- int NumNodes () const

  return the number of nodes in this element
- const Node & GetNode (int i) const

  return the ith FEM node
- Node & GetNode (int i)

  return the ith FEM node
- ElTrans & GetTrans ()

  return the element transformation
- double Interpolate (double x, const Vector &u) const

  interpolate to a point
- void GetVDofs (Array< int > &vdofs) const

  get the ids of all nodes in the element
- int GetOrder () const

  return the polynomial order of this element

Protected Attributes

- Node & _left

  left end point node
- Node & _right

  right end point node
- int _order

  polynomial order
- int _bcl

  left boundary condition
- int _bcr

  right boundary condition
- Basis _basis

  basis object
- ElTrans ∗ _trans

  element transformation for ∗this
- Array< Node ∗ > _nodes

  store FEM nodes

3.9.1 Detailed Description

abstract class for elements

3.9.2 Constructor & Destructor Documentation

3.9.2.1 Element()

```
trt::Element::Element (
            Node left,
            Node right,
            int order )
```

constructor

Parameters

xl	left end point
xr	right end point
order	polynomial order
bcl	left boundary condition
bcr	right boundary condition

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

- /home/sam/trt/fem/Element.hpp
- /home/sam/trt/fem/Element.cpp

3.10 trt::ElTrans Class Reference

reference to physical space transformation

```
#include <ElTrans.hpp>
```

Public Member Functions

- ElTrans (Element ∗el)

  constructor
- double Jacobian (double xref)

  evaluate jacobian in reference space
- double Transform (double xref)

  transform reference to physical space
- Element & GetElement ()

  return the element

Private Attributes

- Element ∗ _el

  store the element

3.10.1 Detailed Description

reference to physical space transformation

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

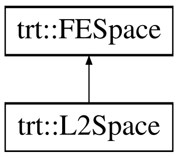
- /home/sam/trt/fem/ElTrans.hpp
- /home/sam/trt/fem/ElTrans.cpp

3.11 trt::FESpace Class Reference

abstract class for finite element spaces

```
#include <FESpace.hpp>
```

Inheritance diagram for trt::FESpace:



Public Member Functions

- FESpace (int Ne, double xb, int order)

    constructor
- int GetVSize () const

    return the number of unknowns
- int GetNumElements () const

    return the number of elements
- int GetNumNodes () const

    return the number of nodes
- Element & GetElement (int e)

    access to element e
- const Node & GetNode (int i) const

    access to node i

Protected Attributes

- Array< Element * > _els

    store the elements in the FESpace
- Array< Node * > _nodes

    store their nodes

3.11.1 Detailed Description

abstract class for finite element spaces

3.11.2 Constructor & Destructor Documentation

3.11.2.1 FESpace()

```
trt::FESpace::FESpace (
             int Ne,
             double xb,
             int order )
```

constructor

Parameters

Ne	number of elements
order	polynomial order

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

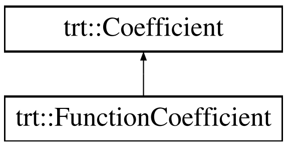
- /home/sam/trt/fem/FESpace.hpp
- /home/sam/trt/fem/FESpace.cpp

3.12 trt::FunctionCoefficient Class Reference

evaluate a function

`#include <Coefficient.hpp>`

Inheritance diagram for trt::FunctionCoefficient:



Public Member Functions

- FunctionCoefficient (double(∗f)(double x))
  constructor
- double Eval (ElTrans &trans, double xref) const
  evaluate

Private Attributes

- double(∗ _f)(double)
  store the function

Additional Inherited Members

3.12.1 Detailed Description

evaluate a function

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

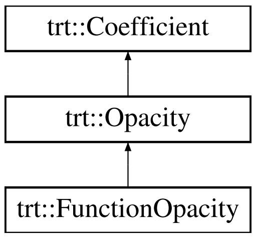
- /home/sam/trt/fem/Coefficient.hpp

3.13 trt::FunctionOpacity Class Reference

space, temperature dependent opacity

```
#include <Opacity.hpp>
```

Inheritance diagram for trt::FunctionOpacity:



Public Member Functions

- FunctionOpacity (double(∗f)(double x, double T))

  constructor
- FunctionOpacity (double(∗f)(double x))

  constructor for space only dependence
- double Eval (ElTrans &trans, double xref) const

  evaluate

Private Attributes

- double(∗ _f)(double x, double T)

  store function of space, temperature
- double(∗ _g)(double x)

  store function of space only

Additional Inherited Members

3.13.1 Detailed Description

space, temperature dependent opacity

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

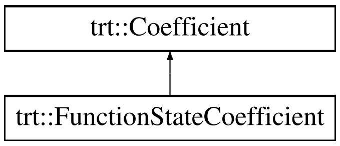
- /home/sam/trt/trt/Opacity.hpp
- /home/sam/trt/trt/Opacity.cpp

3.14 trt::FunctionStateCoefficient Class Reference

evaluate a 2D function through setting the state

```
#include <Coefficient.hpp>
```

Inheritance diagram for trt::FunctionStateCoefficient:



Public Member Functions

- FunctionStateCoefficient (double(∗f)(double, double))

  constructor
- double Eval (ElTrans &trans, double xref) const

  evaluate

Private Attributes

- double(∗ _f)(double, double)

  store the 2D function

Additional Inherited Members

3.14.1 Detailed Description

evaluate a 2D function through setting the state

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

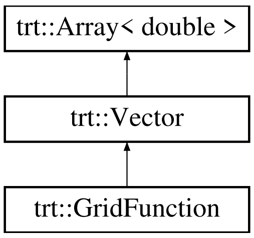
- /home/sam/trt/fem/Coefficient.hpp

3.15 trt::GridFunction Class Reference

represent a solution vector on an FESpace

```
#include <GridFunction.hpp>
```

Inheritance diagram for trt::GridFunction:



Public Member Functions

- GridFunction (FESpace ∗space)

    constructor
- FESpace ∗ GetSpace () const

    return the FESpace associated with the solution vector
- double L2Error (Coefficient ∗exact)

    return the L2 error
- void Project (double(∗f)(double))

    evaluate a function and store in this
- void Project (Coefficient ∗c)

    evaluate at a coefficient at every node
- void operator= (double val)

    assign all elements to a value

Private Attributes

- FESpace ∗ _space

    space associated with the solution vector

3.15.1 Detailed Description

represent a solution vector on an FESpace

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

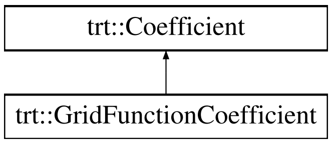
- /home/sam/trt/fem/GridFunction.hpp
- /home/sam/trt/fem/GridFunction.cpp

3.16 trt::GridFunctionCoefficient Class Reference

evaluate a GridFunction as a coefficient

```
#include <Coefficient.hpp>
```

Inheritance diagram for trt::GridFunctionCoefficient:



Public Member Functions

- GridFunctionCoefficient (GridFunction &gf)

  constructor
- double Eval (ElTrans &trans, double xref) const

  evaluate the grid function coefficient

Private Attributes

- GridFunction ∗ _gf

  store the grid function

Additional Inherited Members

3.16.1 Detailed Description

evaluate a GridFunction as a coefficient

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

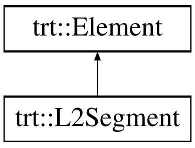
- /home/sam/trt/fem/Coefficient.hpp
- /home/sam/trt/fem/Coefficient.cpp

3.17 trt::L2Segment Class Reference

1D discontinuous galerkin element

```
#include <L2Segment.hpp>
```

Inheritance diagram for trt::L2Segment:



Public Member Functions

- L2Segment (Node left, Node right, int order)

    constructor
- void CalcShape (double x, Vector &shape) const

    evaluate basis functions
- void CalcGradShape (double x, Vector &gshape) const

    evaluate derivatives of basis functions

Additional Inherited Members

3.17.1 Detailed Description

1D discontinuous galerkin element

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

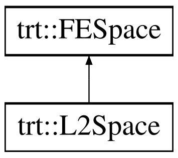
- /home/sam/trt/fem/L2Segment.hpp
- /home/sam/trt/fem/L2Segment.cpp

3.18 trt::L2Space Class Reference

Discontinuous Galerkin finite element space.

`#include <FESpace.hpp>`

Inheritance diagram for trt::L2Space:



Public Member Functions

- L2Space (int Ne, double xb, int order)

    constructor

Additional Inherited Members

3.18.1 Detailed Description

Discontinuous Galerkin finite element space.

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

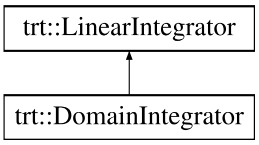
- /home/sam/trt/fem/FESpace.hpp
- /home/sam/trt/fem/FESpace.cpp

3.19 trt::LinearIntegrator Class Reference

abstract class for linear forms

```
#include <LinearIntegrator.hpp>
```

Inheritance diagram for trt::LinearIntegrator:



Public Member Functions

- LinearIntegrator ()

  default constructor
- virtual void Assemble (Element &el, Vector &elvec)

  assemble a local right hand side vector

3.19.1 Detailed Description

abstract class for linear forms

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

- /home/sam/trt/fem/LinearIntegrator.hpp

3.20 trt::LuaReader Class Reference

read in and access a Lua script

```
#include <Lua.hpp>
```

Public Member Functions

- LuaReader (std::string filename)

  constructor
- LuaReader ()

  default constructor
- void SetLuaFile (std::string filename)

  set the lua script file name
- void Parse (int argc, char ∗argv[ ], std::string def="none")

  parse argc and argv for Lua file name
- void VectorFunction (const char ∗field_name, double x, Vector &v) const

  return a vector valued function named field_name
- double ScalarFunction (const char ∗field_name, double x) const

  return a scalar valued function name field_name
- double ScalarFunction (const char ∗field_name, double x, double mu) const

  return a scalar valued function named field_name that takes two arguments
- double Double (const char ∗field_name) const

  return a double named field_name from lua script
- double Double (const char ∗field_name, double def) const

  return a double with default value if not found
- int Int (const char ∗field_name) const

  return a required integer named field_name
- int Int (const char ∗field_name, int def) const

  return an optional integer name field_name. Returns def if not found
- std::string String (const char ∗field_name) const

  read a string field_name from lua script
- std::string String (const char ∗field_name, std::string def) const

  read an optional string return def if not found
- bool Bool (const char ∗field_name) const

  return a bool value. Returns false if not found
- double SourceFunction (double x)

  evaluate source_function from lua script. Lua keyword = source_function
- double InitialConditions (double x)

  evaluate initial conditions. Lua keyword = initial_function
- std::string OutputFile () const

  return the output name from lua script. Lua keyword = output_file
- double EndTime () const

  return the end time. Lua keyword = end_time
- double TimeStep () const

  return the time step. Lua keyword = time_step
- int WriteFreq () const

  return the writer frequency. Lua keyword write_freq
- int FEOrder () const

  return the fe order. Lua keyword = fe_order

Private Attributes

- lua_State ∗ _state

  lua parser

3.20.1 Detailed Description

read in and access a Lua script

3.20.2 Constructor & Destructor Documentation

3.20.2.1 LuaReader()

```
trt::LuaReader::LuaReader (
             std::string filename )
```

constructor

Parameters

filename	name of lua script

3.20.3 Member Function Documentation

3.20.3.1 Parse()

```
void trt::LuaReader::Parse (
             int argc,
             char * argv[],
             std::string def = "none" )
```

parse argc and argv for Lua file name

Parameters

argc	number of command line arguments
argv	command line arguments array
def	default lua script to load if Lua script not specified

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

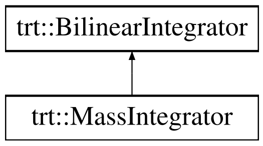
- /home/sam/trt/utils/Lua.hpp
- /home/sam/trt/utils/Lua.cpp

3.21 trt::MassIntegrator Class Reference

integrate a mass matrix $\int B_i B_j dx$

```
#include <BilinearIntegrator.hpp>
```

Inheritance diagram for trt::MassIntegrator:



Public Member Functions

- MassIntegrator ()

  default constructor
- MassIntegrator (Coefficient ∗c)

  constructor with coefficient
- void Assemble (Element &el, Matrix &elmat)

  assemble local mass matrix

Private Attributes

- Coefficient ∗ _c

  coefficient
- Vector _shape

  store shape evaluations
- Matrix _op

  store outer product of shape functions

3.21.1 Detailed Description

integrate a mass matrix $\int B_i B_j dx$

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

- /home/sam/trt/fem/BilinearIntegrator.hpp
- /home/sam/trt/fem/BilinearIntegrator.cpp

3.22 trt::Matrix Class Reference

store a matrix

```
#include <Matrix.hpp>
```

Public Member Functions

- Matrix ()
  default constructor
- Matrix (int m, int n=-1)
  constructor. defaults to square
- Matrix (const Matrix &m)
  copy constructor
- Matrix & operator= (const Matrix &m)
  copy assignment
- void Resize (int m, int n=-1)
  set the size
- int Height () const
  return the number of rows
- int Width () const
  return the number of columns
- const double $*$ Data () const
  const access to data
- double $*$ Data ()
  direct access to the data
- double & operator() (int i, int j)
  index into matrix. Hides COLUMN MAJOR bs behind the scenes for stupid LAPACK
- double operator() (int i, int j) const
  const index into matrix
- void Solve (const Vector &b, Vector &x) const
  solve the system $Ax = b$ using lapack's dgesv (LU factor and solve)
- void operator$*$= (double val)
  scale all elements by val
- void operator+= (const Matrix &mat)
  add a matrix to this
- void Add (const Matrix &a, Matrix &sum) const
  add two matrices together
- void Mult (double alpha, const Vector &x, double beta, Vector &b) const
  do $b = alphathisx + beta * b$
- void Mult (const Vector &x, Vector &b) const
  just matrix vector product
- std::ostream & Print (std::ostream &out=std::cout) const
  print to output

Private Attributes

- Array$<$ double $>$ _data
  store the data
- int _m
  number of rows
- int _n
  number of cols

3.22.1 Detailed Description

store a matrix

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

- /home/sam/trt/linalg/Matrix.hpp
- /home/sam/trt/linalg/Matrix.cpp

3.23 trt::Node Class Reference

represent an FEM node

```
#include <Node.hpp>
```

Public Member Functions

- Node (double x, double xref, int gid, int bc)

  constructor
- void SetGlobalID (int id)

  set the global id
- int GlobalID () const

  return the global id
- int BC () const

  return the boundary type
- double X () const

  return node location
- double XRef () const

  return location in reference space

Private Attributes

- double _x

  location in physical space
- double _xref

  location in reference space
- int _gid

  global id of this node
- int _bc

  boundary type

3.23.1 Detailed Description

represent an FEM node

3.23.2 Constructor & Destructor Documentation

3.23.2.1 Node()

```
trt::Node::Node (
            double x,
            double xref,
            int gid,
            int bc ) [inline]
```

constructor

Parameters

x	node location in physical space
xref	node location in reference space
gid	global id
bc	boundary condition type

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

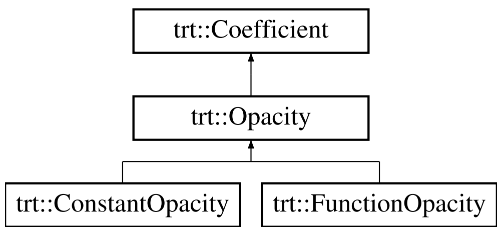
- /home/sam/trt/fem/Node.hpp

3.24 trt::Opacity Class Reference

abstract representation of space and temperature dependent opacity

```
#include <Opacity.hpp>
```

Inheritance diagram for trt::Opacity:



Public Member Functions

- void SetTemperature (const Vector &T)

  set the temperature Vector for evaluating temperature dependence

Protected Attributes

- Vector _T

  store the tempearture Vector

3.24.1 Detailed Description

abstract representation of space and temperature dependent opacity

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

- /home/sam/trt/trt/Opacity.hpp

3.25 trt::Poly1D Class Reference

represent a polynomial of one variable

`#include <Basis.hpp>`

Public Member Functions

- Poly1D ()

  default constructor
- Poly1D (const Array< double > &c)

  constructor
- double Eval (double x) const

  evaluate at a point
- Poly1D Derivative () const

  return the derivative of ∗this
- std::ostream & Print (std::ostream &out=std::cout) const

  print the polynomial to the ofstream

Private Attributes

- Array< double > _c

  store the polynomial coefficients

3.25.1 Detailed Description

represent a polynomial of one variable

3.25.2 Constructor & Destructor Documentation

3.25.2.1 Poly1D()

```
trt::Poly1D::Poly1D (
            const Array< double > & c )  [inline]
```

constructor

provide coefficients in ascending powers

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

- /home/sam/trt/fem/Basis.hpp

3.26 trt::Quadrature Class Reference

arbitrary order Gauss Legendre quadrature

```
#include <Quadrature.hpp>
```

Public Member Functions

- Quadrature (int p, double a=0, double b=1)

  constructor
- int NumPoints () const

  return the number of integration points
- double Point (int i) const

  return the ith integration point
- double Weight (int i) const

  return the ith weight
- int Order () const

  return the integration order

Private Attributes

- int _p

  integration order
- double _a

  lower limit
- double _b

  upper limit
- Array< double > _x

  integration points
- Array< double > _w

  integration weights

3.26.1 Detailed Description

arbitrary order Gauss Legendre quadrature

3.26.2 Constructor & Destructor Documentation

3.26.2.1 Quadrature()

```
trt::Quadrature::Quadrature (
             int p,
             double a = 0,
             double b = 1 )
```

constructor

Parameters

p	integration order
a	lower limit of integration
b	upper limit of integration

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

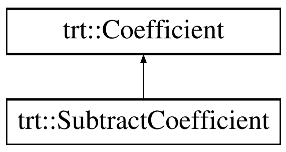
- /home/sam/trt/general/Quadrature.hpp
- /home/sam/trt/general/Quadrature.cpp

3.27 trt::SubtractCoefficient Class Reference

subtract a coefficient from another one

```
#include <Coefficient.hpp>
```

Inheritance diagram for trt::SubtractCoefficient:



Public Member Functions

- SubtractCoefficient (Coefficient ∗c1, Coefficient ∗c2)

  constructor. evaluates c1 - c2
- double Eval (ElTrans &trans, double xref) const

  evaluate the two coefficients and subtract

Private Attributes

- Coefficient ∗ **_c1**
- Coefficient ∗ **_c2**

Additional Inherited Members

3.27.1 Detailed Description

subtract a coefficient from another one

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

- /home/sam/trt/fem/Coefficient.hpp

3.28 trt::Sweeper Class Reference

performs direct inversion of transport equation

```
#include <Sweeper.hpp>
```

Public Member Functions

- Sweeper (FESpace ∗space, Quadrature &quad, Coefficient ∗inflow)

  constructor
- void Solve (Coefficient ∗sig_s, Coefficient ∗sig_t, Coefficient ∗q, TVector ∗dq, const Vector &phi, TVector &psi) const

  perform a sweep for all angles
- void SweepLR (double mu, Coefficient ∗sig_s, Coefficient ∗sig_t, Coefficient ∗q, Vector ∗dq, const Vector &phi, Vector &psi_n) const

  sweep from left to right (mu $>$ 0)
- void SweepRL (double mu, Coefficient ∗sig_s, Coefficient ∗sig_t, Coefficient ∗q, Vector ∗dq, const Vector &phi, Vector &psi_n) const

  sweep from right to left (mu $<$ 0)
- void SweepLR (double mu, Coefficient ∗sig_s, Coefficient ∗sig_t, Coefficient ∗q, const Vector &phi, Vector &psi_n) const

  sweep without the discrete source
- void SweepRL (double mu, Coefficient ∗sig_s, Coefficient ∗sig_t, Coefficient ∗q, const Vector &phi, Vector &psi_n) const

  sweep without discrete source

Private Attributes

- FESpace ∗ _space

  store FE space for psi
- Quadrature _quad

  store quadrature object for angular integration
- Coefficient ∗ _inflow

  store the inflow function

3.28.1 Detailed Description

performs direct inversion of transport equation

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

- /home/sam/trt/trt/Sweeper.hpp
- /home/sam/trt/trt/Sweeper.cpp

3.29 trt::Timer Class Reference

stopwatch class that wraps std::chrono

`#include <Timer.hpp>`

Public Member Functions

- Timer ()

  constructor
- void Start ()

  start the clock
- void Stop ()

  stop the clock
- double GetDuration () const

  return the duration

Private Attributes

- std::chrono::time_point< std::chrono::system_clock > _start

  store the start time
- std::chrono::duration< double > _el

  store the elapsed time

3.29.1 Detailed Description

stopwatch class that wraps std::chrono

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

- /home/sam/trt/utils/Timer.hpp
- /home/sam/trt/utils/Timer.cpp

3.30 trt::TransportOperator Class Reference

driver for transport solver

`#include <TransportOperator.hpp>`

Public Member Functions

- TransportOperator (FESpace ∗space, int Nangles, Opacity ∗sig_s, Opacity ∗sig_t, Coefficient ∗q, Coefficient ∗inflow, Opacity ∗cv=NULL)

  constructor
- void SourceIteration (TVector &psi, int niter, double tol) const

  source iteration but with opacities from constructor
- int SourceIteration (Coefficient ∗sig_t, Coefficient ∗sig_s, Coefficient ∗q, TVector ∗dq, int niter, double tol, TVector &psi, bool LOUD=false) const

  perform source iteration with a discrete source
- void NewtonIteration (const TVector &psi_p, const Vector &T_p, int n_outer, double t_outer, int n_inner, double t_inner, double dt, TVector &psi, Vector &T)

  perform one newton temperature iteration
- void BackwardEuler (const TVector &psi_p, int niter, double tol, double dt, TVector &psi)

  perform one time step
- void ComputeScalarFlux (const TVector &psi, Vector &phi) const

  compute the scalar flux
- void SetA (double a)

  set the radiation temperature thing (a)

Private Member Functions

- void FormSource (double dt, const TVector &psi_p, const Vector &T_p, const Vector &Ts, TVector &dq) const

  form the source term for backward euler Newton Iteration
- void FormScattering (double dt, const Vector &Ts, Vector &scattering) const

  form the scattering coefficient for backward euler Newton iteration
- void UpdateTemperature (double dt, const Vector &T_old, const Vector &T_p, const Vector &phi, Vector &T) const

  update the temperature vector

Private Attributes

- FESpace ∗ _space

  store the FESpace
- int _Nangles

  number of angles
- Coefficient ∗ _sig_a

  abs cross section
- Opacity ∗ _sig_s

  scattering cross section
- Opacity ∗ _sig_t

  total cross section
- Coefficient ∗ _q

  source function
- Coefficient ∗ _inflow

  inflow function
- Opacity ∗ _cv

  heat capacity
- double _c

  speed of light

- double _a

  radiation temperature thing
- Quadrature _quad

  Sn angular quadrature object.
- Sweeper _sweeper

  sweeper to invert each source iteration

3.30.1 Detailed Description

driver for transport solver

3.30.2 Constructor & Destructor Documentation

3.30.2.1 TransportOperator()

```
trt::TransportOperator::TransportOperator (
             FESpace * space,
             int Nangles,
             Opacity * sig_s,
             Opacity * sig_t,
             Coefficient * q,
             Coefficient * inflow,
             Opacity * cv = NULL )
```

constructor

Parameters

finite	element space for transport
Nangles	number of angles (S_?)
sig_s	scattering cross section
sig_t	total cross section
q	source
cv	heat capacity

3.30.3 Member Function Documentation

3.30.3.1 BackwardEuler()

```
void trt::TransportOperator::BackwardEuler (
             const TVector & psi_p,
             int niter,
             double tol,
             double dt,
             TVector & psi )
```

perform one time step

Parameters

in	*psi←↩ _p*	angular flux from previous time step
in	*niter*	maximum number of source iterations to try
in	*tol*	iterative tolerance to stop at
in	*dt*	time step size
out	*psi*	angular flux at next time step

3.30.3.2 NewtonIteration()

```
void trt::TransportOperator::NewtonIteration (
             const TVector & psi_p,
             const Vector & T_p,
             int n_outer,
             double t_outer,
             int n_inner,
             double t_inner,
             double dt,
             TVector & psi,
             Vector & T )
```

perform one newton temperature iteration

Parameters

in	*psi_p*	angular flux from previous iteration
in	*T_p*	temperature from previous iteration
in	*n_outer*	max number of newton iterations
in	*t_outer*	newton iteration tolerance
in	*n_inner*	max number of source iterations
in	*t_inner*	source iteration tolerance
in	*dt*	time step size
out	*psi*	converged angular flux
out	*T*	converged temperature

3.30.3.3 SourceIteration()

```
int trt::TransportOperator::SourceIteration (
             Coefficient * sig_t,
             Coefficient * sig_s,
             Coefficient * q,
             TVector * dq,
             int niter,
             double tol,
             TVector & psi,
             bool LOUD = false ) const
```

perform source iteration with a discrete source

Parameters

in	*sig↵_t*	total interaction (term that multiplies psi on LHS)
in	*sig↵_s*	scattering (phi term that is lagged)
in	*q*	fixed source on rhs
in	*dq*	discrete source term. can be null
in	*niter*	maximum number of iterations
in	*tol*	relative tolerance before stopping
in,out	*initial*	guess for psi. final solution returned in psi
in	*LOUD*	print iteration info to terminal

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

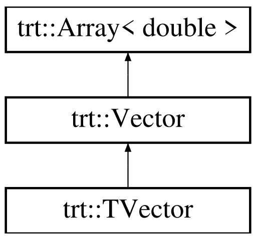
- /home/sam/trt/trt/TransportOperator.hpp
- /home/sam/trt/trt/TransportOperator.cpp

3.31 trt::TVector Class Reference

store psi. strides in space then angle

```
#include <TVector.hpp>
```

Inheritance diagram for trt::TVector:



Public Member Functions

- TVector (FESpace *space, int Nangles)

  constructor
- double & operator() (int angle, int i)

2D indexing
- double operator() (int angle, int i) const

    const indexing
- void operator= (double val)

    set all values
- void GetAngle (int angle, Vector &psi_n) const

    get an angle
- void SetAngle (int angle, const Vector &psi_n)

    set an angle
- FESpace ∗ GetSpace () const

    return the FESpace

Private Attributes

- FESpace ∗ _space

    Finite element space this is build on.
- int _Nangles

    number of angles

3.31.1 Detailed Description

store psi. strides in space then angle

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

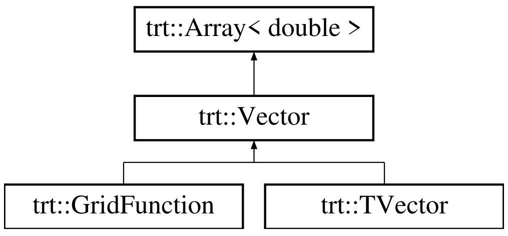
- /home/sam/trt/trt/TVector.hpp
- /home/sam/trt/trt/TVector.cpp

3.32 trt::Vector Class Reference

class for storing a vector of doubles

```
#include <Vector.hpp>
```

Inheritance diagram for trt::Vector:



Public Member Functions

- Vector (int N, double val=0)

    constructor
- Vector ()

    default constructor
- void operator= (const Vector &v)

    copy assignment
- void operator= (double val)

    set all elements to a value
- void operator/= (double val)

    divide all entries by val
- void operator∗= (double val)

    multiply all entries by val
- double operator∗ (const Vector &v) const

    dot product
- void OuterProduct (const Vector &v, Matrix &mat) const

    outer product
- void GetSubVector (const Array< int > &vdofs, Vector &subv) const

    return the subvector corresponding to the ordering in vdofs
- void operator+= (const Vector &v)

    add a vector to this
- void Subtract (const Vector &v, Vector &diff) const

    subtract from this
- bool IsFinite () const

    check if all entries in vector are finite

3.32.1 Detailed Description

class for storing a vector of doubles

3.32.2 Constructor & Destructor Documentation

3.32.2.1 Vector()

```
trt::Vector::Vector (
             int N,
             double val = 0 )
```

constructor

Parameters

N	size of vector
val	initial value for all elements

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

- /home/sam/trt/linalg/Vector.hpp
- /home/sam/trt/linalg/Vector.cpp

3.33 trt::WallTimer Class Reference

singleton class for wall timer using std::chrono

```
#include <WallTimer.hpp>
```

Public Member Functions

- ∼WallTimer ()

    destructor: prints wall time

Static Public Member Functions

- static WallTimer & instance ()

    returns a static instance so only one can exist in the program

Private Member Functions

- WallTimer ()

    private constructor prevents instantiating more than one WallTimer

Private Attributes

- std::chrono::time_point< std::chrono::system_clock > _start

    store start time
- std::chrono::duration< double > _el

    store the elapsed time
- bool _init

    true if initialized

3.33.1 Detailed Description

singleton class for wall timer using std::chrono

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

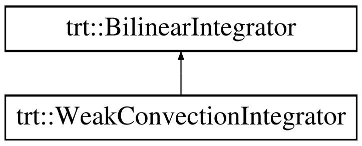
- /home/sam/trt/utils/WallTimer.hpp
- /home/sam/trt/utils/WallTimer.cpp

3.34 trt::WeakConvectionIntegrator Class Reference

integrate weak convection matrix $\int -B_j \frac{dB_i}{dx} dx$

```
#include <BilinearIntegrator.hpp>
```

Inheritance diagram for trt::WeakConvectionIntegrator:



Public Member Functions

- WeakConvectionIntegrator ()

  constructor
- WeakConvectionIntegrator (Coefficient ∗c)

  constructor with coefficient
- void Assemble (Element &el, Matrix &elmat)

  assemble local matrix

Private Attributes

- Coefficient ∗ _c

  store coefficient
- Vector _shape

  store shape evals
- Vector _gshape

  store grad shape evals
- Matrix _op

  outer product

3.34.1 Detailed Description

integrate weak convection matrix $\int -B_j \frac{dB_i}{dx} dx$

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

- /home/sam/trt/fem/BilinearIntegrator.hpp
- /home/sam/trt/fem/BilinearIntegrator.cpp

3.35 trt::Writer Class Reference

stores pointers to GridFunctions and writes them to file

```
#include <Writer.hpp>
```

Public Member Functions

- Writer (std::string name="solution")

  constructor. provide base name for output files
- void Add (GridFunction &gf, std::string name)

  add a solution variable to the output list
- void SetFreq (int f)

  set the output frequency for time dependent calculations
- void Write (bool force=false)

  write to file

Private Attributes

- int _f

  output frequency
- Array< GridFunction ∗ > _gf

  store pointers to GridFunctions
- Array< std::string > _names

  store their corresponding names
- std::string _base_name

  store the base name
- int _count

  number of times Write has been called
- int _writes

  number of files written

3.35.1 Detailed Description

stores pointers to GridFunctions and writes them to file

3.35.2 Member Function Documentation

3.35.2.1 Write()

```
void trt::Writer::Write (
            bool force = false )
```

write to file

Parameters

force	writes regardless of frequency if true

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

- /home/sam/trt/utils/Writer.hpp
- /home/sam/trt/utils/Writer.cpp

Index