Studienprojekt Modellbildung und Simulation 2015/16

Erzeugt von Doxygen 1.6.1

Tue Mar 22 19:22:43 2016

Inhaltsverzeichnis

Kapitel 1

Datenstruktur-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 2

Datenstruktur-Verzeichnis

2.1 Datenstrukturen

Hier folgt die Aufzählung aller Datenstrukturen mit einer Kurzbeschreibung:

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller dokumentierten Dateien mit einer Kurzbeschreibung:

Kapitel 4

Datenstruktur-Dokumentation

4.1 Icarus::assembleFem Klassenreferenz

Öffentliche Methoden

- **assembleFem** (double sh, int sx, int sy, int sz)
- void **assemble** (DistEllpackMatrix< double > &Matrix, SlicedVector< double > &rhs)

4.1.1 Ausführliche Beschreibung

Definiert in Zeile 15 der Datei assemblefem.hpp.

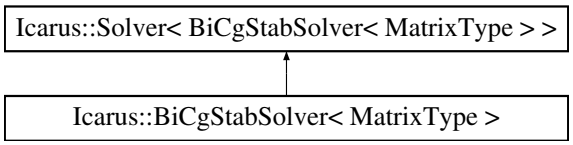
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/assemblefem.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/assemblyLGS.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/assemblyMatrixRow.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/assemblyRHSLoad.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/assemblyRHSNeumann.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/basis.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/getxyz.cpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/quadratur.cpp

4.2 Icarus::BiCgStabSolver< MatrixType > Template-Klassenreferenz

Löst das LGS Ax=b mithilfe der Methode der bikonjugierten Gradienten.

`#include <bicgstabsolver.hpp>`Klassendiagramm für Icarus::BiCgStabSolver< MatrixType >::



Öffentliche Typen

- typedef MatrixType::VectorType VectorType

  Mit der Matrix (bzgl. Speicherverteilung etc.) verträgliche Vektortyp. Die rechte Seite b muss diesen Typ besitzen, die Lösung besitzt ebenfalls diesen Typ.

- typedef MatrixType::ScalarType ScalarType

  Typ der Einträge der Matrix A.

- typedef MatrixType::RealType RealType

  Typ, den z.B. die Norm eines Vektors von Elementen vom Typ ScalarType hat.

Öffentliche Methoden

- BiCgStabSolver (const MatrixType &A, const VectorType &b, RealType tol=DEFAULT_TOL, const MatrixType ∗K1inv=nullptr, const MatrixType ∗K2inv=nullptr)

  Konstruktor.

Statische öffentliche Attribute

- static const long long MAX_ITER = 10000000000L

  Anzahl der Iterationen, nach der abgebrochen wird, wenn die Toleranz nicht erreicht werden kann.

- static const RealType DEFAULT_TOL

  Toleranz, die ohne explizite Angabe angenommen wird.

Freundbeziehungen

- class **Solver**< **BiCgStabSolver**< **MatrixType** > >

4.2.1 Ausführliche Beschreibung

template<**typename MatrixType**> **class Icarus::BiCgStabSolver**< **MatrixType** >

Löst das LGS Ax=b mithilfe der Methode der bikonjugierten Gradienten.

Template Parameters:

MatrixType Typ der Matrix A und ggf. der Vorkonditionierer K1 und K2.

Definiert in Zeile 15 der Datei bicgstabsolver.hpp.

4.2.2 Beschreibung der Konstruktoren und Destruktoren

4.2.2.1 template<typename MatrixType> Icarus::BiCgStabSolver< MatrixType >::BiCgStabSolver (const MatrixType & *A*, const VectorType & *b*, RealType *tol* = DEFAULT_TOL, const MatrixType * *K1inv* = `nullptr`, const MatrixType * *K2inv* = `nullptr`) `[inline]`

Konstruktor.

Parameter:

A Matrix A in Ax=b

b Rechte Seite b in Ax=b

tol Residuumsnorm, bei der abgebrochen werden soll

K1inv Linksvorkonditionierer (nullptr für keinen)

K2inv Rechtsvorkonditionierer (nullptr für keinen)

Definiert in Zeile 10 der Datei bicgstabsolver.tpp.

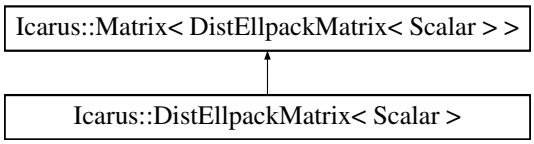
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/bicgstabsolver.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/bicgstabsolver.tpp

4.3 Icarus::DistEllpackMatrix< Scalar > Template-Klassenreferenz

Dünnbesetzte, quadratische Matrix, deren Zeilen gleichverteilt auf einer Menge von Nodes liegen.

`#include <distellpackmatrix.hpp>`Klassendiagramm für Icarus::DistEllpackMatrix< Scalar >::



Öffentliche Typen

- typedef MatrixTraits< DistEllpackMatrix< Scalar > >::VectorType VectorType

  Zugeordneter (d.h. bezüglich der Operatoren vertäglicher) Vektor-Typ.

Öffentliche Methoden

- DistEllpackMatrix (size_t dim_global, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- **DistEllpackMatrix** (DistEllpackMatrix &&other)
- **DistEllpackMatrix** (const DistEllpackMatrix &other)
- DistEllpackMatrix & **operator=** (DistEllpackMatrix &&other)
- DistEllpackMatrix & **operator=** (const DistEllpackMatrix &other)
- MPI_Comm get_comm () const

  Gibt den Kommunikator in die Prozessgruppe der Matrix zurück.

- size_t get_dim_local () const

  Gibt die lokale Dimension der Matrix, d.h. die Anzahl der auf der aufrufenden Node gespeicherten Zeilen zurück.

- size_t get_dim_local_nopad () const

  Gibt die Anzahl der Zeilen, wie sie auf einer der ersten N-1 Nodes liegen, zurück.

- size_t get_dim_global () const

  Gibt die globale Dimension, d.h. die Anzahl der Zeilen der Matrix, zurück.

- void prepare_sequential_fill (size_t max_row_length)

Bereitet den zeilenweisen Füllvorgang der Matrix vor.

- void sequential_fill (size_t colind, const Scalar &val)

  Fülle die Matrix zeilenweise.

- void end_of_row ()

  Beende eine Zeile beim zeilenweisen Füllen der Matrix.

- bool is_filled () const

  Prüft, ob die Matrix korrekt gefüllt wurde.

- size_t first_row_on_node () const

  Gibt den globalen Index der ersten auf der Node liegenden Zeile zurück.

- DistEllpackMatrix precond_equi () const

  Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

- DistEllpackMatrix precond_jacobi () const

  Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

- void print_local_data (std::ostream &os) const

  Schreibe den lokalen Inhalt des Block in den Stream out.

Öffentliche, statische Methoden

- static DistEllpackMatrix import_csr_file (const std::string &filename, MPI_-Comm new_comm=MPI_COMM_WORLD)

  Lese eine DistEllpackMatrix aus einem CSR-artigen Dateiformat ein.

Freundbeziehungen

- class **Matrix< DistEllpackMatrix< Scalar > >**

4.3.1 Ausführliche Beschreibung

template<typename Scalar> class Icarus::DistEllpackMatrix< Scalar >

Dünnbesetzte, quadratische Matrix, deren Zeilen gleichverteilt auf einer Menge von Nodes liegen. Die Zeilen dieser Matrix liegen (annähernd) gleichverteilt auf einer Menge von Nodes der zugeordneten Prozessgruppe. Lokal auf der Node werden die Zeilen im Ellpack-Format gespeichert für maximale Effizienz der MV-Multiplikation in CUDA.

Dieser Matrixtyp kann nur sequentiell zeilenweise gefüllt werden, wobei die maximale Zeilenlänge (node-weise) bekannt sein muss. Siehe dazu auch die Dokumtation der Funktion sequential_fill.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 29 der Datei distellpackmatrix.hpp.

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 template<typename Scalar > Icarus::DistEllpackMatrix< Scalar >::DistEllpackMatrix (size_t *dim_global*, MPI_Comm *my_comm* = MPI_COMM_WORLD) [inline]

Standardkonstruktor. Erzeugt einen Vektor der Dimension dim, der komplett auf jeder Node der Prozessgruppe my_comm liegt.

Parameter:

dim_global Dimension der Matrix.

my_comm Kommunikator in die Prozessgruppe der Matrix.

Definiert in Zeile 14 der Datei distellpackmatrix.tpp.

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 template<typename Scalar > void Icarus::DistEllpackMatrix< Scalar >::end_of_row () [inline]

Beende eine Zeile beim zeilenweisen Füllen der Matrix. Diese Funktion beendet die aktuelle Zeile beim Füllvorgang und setzt den Füllcursor auf die nächste Zeile, oder beendet den Füllvorgang, falls die letzte auf der Node vorhandene Zeile beendet wurde.

Vor dem erste Aufruf dieser Funktion muss mit prepare_sequential_fill die maximal auf der füllenden Node auftretende Zeilenlänge gesetzt werden.

Diese Funktion muss während eines Füllvorgangs auf der Node genau dim_local mal aufgerufen werden.

Definiert in Zeile 191 der Datei distellpackmatrix.tpp.

4.3.3.2 template<typename Scalar > DistEllpackMatrix< Scalar > Icarus::DistEllpackMatrix< Scalar >::import_csr_file (const std::string & *filename*, MPI_Comm *new_comm* = MPI_COMM_WORLD) [inline, static]

Lese eine DistEllpackMatrix aus einem CSR-artigen Dateiformat ein. Das benötigte Dateiformat wird von dem MATLAB-Skript /util/csrwrite.m erzeugt. Die Informatio-

nen werden in drei Dateien gespeichert, deren Namen aus einem gemeinsamen Präfix und verschiedenen Endungen bestehen. Dieser Funktion wird (wie auch csrwrite.m) dieser Präfix übergeben.

Parameter:

filename Präfix des Dateinamens des Dateitripels, das eingelesen werden soll.

new_comm Kommunikator in die Prozessgruppe, der die neu erzeugte Matrix gehören soll.

Rückgabe:

Gibt die aus dem Dateitripel erzeugte DistEllpackMatrix zurück.

Definiert in Zeile 238 der Datei distellpackmatrix.tpp.

4.3.3.3 template<typename Scalar> bool Icarus::DistEllpackMatrix< Scalar >::is_filled () const [inline]

Prüft, ob die Matrix korrekt gefüllt wurde. Ein positiver Rückgabewert dieser Funktion ist einerseits ein Indikator für einen erfolgreich abgeschlossenen Füllvorgang und andererseits die Voraussetzung für sämtliche algebraische Operationen mit der Matrix.

Rückgabe:

Gibt zurück, ob die Matrix korrekt gefüllt wurde.

Definiert in Zeile 149 der Datei distellpackmatrix.hpp.

4.3.3.4 template<typename Scalar > DistEllpackMatrix< Scalar > Icarus::DistEllpackMatrix< Scalar >::precond_equi () const [inline]

Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

Rückgabe:

Der Vorkonditionierer hat denselben Typ wie das Objekt, auf das die Funktion aufgerufen wird.

Definiert in Zeile 324 der Datei distellpackmatrix.tpp.

4.3.3.5 template<typename Scalar > DistEllpackMatrix< Scalar > Icarus::DistEllpackMatrix< Scalar >::precond_jacobi () const [inline]

Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer. Wenn in einer Zeile eine Null auf der Diagonalen steht, wird diese Zeile durch die Vorkonditionierung nicht verändert.

Rückgabe:

Der Vorkonditionierer hat denselben Typ wie das Objekt, auf das die Funktion aufgerufen wird.

Definiert in Zeile 344 der Datei distellpackmatrix.tpp.

4.3.3.6 template<typename Scalar > void Icarus::DistEllpackMatrix< Scalar >::prepare_sequential_fill (size_t *max_row_length*) [inline]

Bereitet den zeilenweisen Füllvorgang der Matrix vor. Diese Funktion muss von jeder Node genau einmal zu Beginn des Füllvorgangs aufgerufen werden. Anschließend können die Zeilen mit sequential_fill und end_of_row gefüllt werden, beginnend bei der lokal ersten Zeile. Die maximale auf dieser Node auftretende Zeilenlänge muss vorher bekannt sein.

Parameter:

max_row_length Maximal auf dieser Node auftretende Zeilenlänge.

Definiert in Zeile 161 der Datei distellpackmatrix.tpp.

4.3.3.7 template<typename Scalar > void Icarus::DistEllpackMatrix< Scalar >::print_local_data (std::ostream & *os*) const [inline]

Schreibe den lokalen Inhalt des Block in den Stream out. Für die Verwendung dieser Funktion muss eine entsprechende Überladung des Operators std::ostream::operator<<(Scalar) existieren.

Parameter:

out Stream, in den die Ausgabe geschrieben werden soll.

Definiert in Zeile 371 der Datei distellpackmatrix.tpp.

4.3.3.8 template<typename Scalar > void Icarus::DistEllpackMatrix< Scalar >::sequential_fill (size_t *colind*, const Scalar & *val*) [inline]

Fülle die Matrix zeilenweise. Diese Funktion fügt der aktuellen Zeile den Wert val mit Spaltenindex col hinzu. Vor dem erste Aufruf dieser Funktion muss mit prepare_sequential_fill die maximal auf der füllenden Node auftretende Zeilenlänge gesetzt werden. Nachdem der letzte Eintrag einer Zeile gesetzt wurde, wird mit end_of_row die Zeile beendet.

Parameter:

colind Spaltenindex des einzutregenden Werts. Es muss colind < dim_global gelten.

val Wert, der an die Position colind geschrieben werden soll.

Definiert in Zeile 179 der Datei distellpackmatrix.tpp.

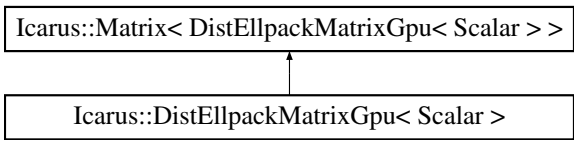
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrix.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrix.tpp

4.4 Icarus::DistEllpackMatrixGpu< Scalar > Template-Klassenreferenz

Dünnbesetzte, quadratische Matrix, deren Zeilen gleichverteilt auf einer Menge von Nodes liegen.

`#include <distellpackmatrixgpu.hpp>`Klassendiagramm für Icarus::DistEllpackMatrixGpu< Scalar >::



Öffentliche Typen

- typedef MatrixTraits< DistEllpackMatrixGpu< Scalar > >::VectorType VectorType

  Zugeordneter (d.h. bezüglich der Operatoren vertäglicher) Vektor-Typ.

Öffentliche Methoden

- DistEllpackMatrixGpu (size_t dim_global, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- **DistEllpackMatrixGpu** (DistEllpackMatrixGpu &&other)
- **DistEllpackMatrixGpu** (const DistEllpackMatrixGpu &other)
- DistEllpackMatrixGpu & **operator=** (DistEllpackMatrixGpu &&other)
- DistEllpackMatrixGpu & **operator=** (const DistEllpackMatrixGpu &other)
- MPI_Comm get_comm () const

  Gibt den Kommunikator in die Prozessgruppe der Matrix zurück.

- size_t get_dim_local () const

  Gibt die lokale Dimension der Matrix, d.h. die Anzahl der auf der aufrufenden Node gespeicherten Zeilen zurück.

- size_t get_dim_local_nopad () const

  Gibt die Anzahl der Zeilen, wie sie auf einer der ersten N-1 Nodes liegen, zurück.

- size_t get_dim_global () const

  Gibt die globale Dimension, d.h. die Anzahl der Zeilen der Matrix, zurück.

- void prepare_sequential_fill (size_t max_row_length)

Bereitet den zeilenweisen Füllvorgang der Matrix vor.

- void sequential_fill (size_t colind, const Scalar &val)

  Fülle die Matrix zeilenweise.

- void end_of_row ()

  Beende eine Zeile beim zeilenweisen Füllen der Matrix.

- bool is_filled () const

  Prüft, ob die Matrix korrekt gefüllt wurde.

- size_t first_row_on_node () const

  Gibt den globalen Index der ersten auf der Node liegenden Zeile zurück.

- DistEllpackMatrixGpu precond_equi () const

  Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

- DistEllpackMatrixGpu precond_jacobi () const

  Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

- void print_local_data (std::ostream &os) const

  Schreibe den lokalen Inhalt des Block in den Stream out.

Öffentliche, statische Methoden

- static DistEllpackMatrixGpu import_csr_file (const std::string &filename, MPI_Comm new_comm=MPI_COMM_WORLD)

  Lese eine DistEllpackMatrixGpu aus einem CSR-artigen Dateiformat ein.

Freundbeziehungen

- class **Matrix< DistEllpackMatrixGpu< Scalar > >**

4.4.1 Ausführliche Beschreibung

template<typename Scalar> class Icarus::DistEllpackMatrixGpu< Scalar >

Dünnbesetzte, quadratische Matrix, deren Zeilen gleichverteilt auf einer Menge von Nodes liegen. Die Zeilen dieser Matrix liegen (annähernd) gleichverteilt auf einer Menge von Nodes der zugeordneten Prozessgruppe. Lokal auf der Node werden die Zeilen im Ellpack-Format gespeichert für maximale Effizienz der MV-Multiplikation in CUDA.

Dieser Matrixtyp kann nur sequentiell zeilenweise gefüllt werden, wobei die maximale Zeilenlänge (node-weise) bekannt sein muss. Siehe dazu auch die Dokumtation der Funktion sequential_fill.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 31 der Datei distellpackmatrixgpu.hpp.

4.4.2 Beschreibung der Konstruktoren und Destruktoren

4.4.2.1 template<typename Scalar > Icarus::DistEllpackMatrixGpu< Scalar >::DistEllpackMatrixGpu (size_t *dim_global*, MPI_Comm *my_comm* = MPI_COMM_WORLD) [inline]

Standardkonstruktor. Erzeugt einen Vektor der Dimension dim, der komplett auf jeder Node der Prozessgruppe my_comm liegt.

Parameter:

dim_globaö Dimension der Matrix.

my_comm Kommunikator in die Prozessgruppe der Matrix.

Definiert in Zeile 26 der Datei distellpackmatrixgpu.tpp.

4.4.3 Dokumentation der Elementfunktionen

4.4.3.1 template<typename Scalar > void Icarus::DistEllpackMatrixGpu< Scalar >::end_of_row () [inline]

Beende eine Zeile beim zeilenweisen Füllen der Matrix. Diese Funktion beendet die aktuelle Zeile beim Füllvorgang und setzt den Füllcursor auf die nächste Zeile, oder beendet den Füllvorgang, falls die letzte auf der Node vorhandene Zeile beendet wurde.

Vor dem erste Aufruf dieser Funktion muss mit prepare_sequential_fill die maximal auf der füllenden Node auftretende Zeilenlänge gesetzt werden.

Diese Funktion muss während eines Füllvorgangs auf der Node genau dim_local mal aufgerufen werden.

Definiert in Zeile 200 der Datei distellpackmatrixgpu.tpp.

4.4.3.2 template<typename Scalar > DistEllpackMatrixGpu< Scalar > Icarus::DistEllpackMatrixGpu< Scalar >::import_csr_file (const std::string & *filename*, MPI_Comm *new_comm* = MPI_COMM_WORLD) [inline, static]

Lese eine DistEllpackMatrixGpu aus einem CSR-artigen Dateiformat ein. Das benötigte Dateiformat wird von dem MATLAB-Skript /util/csrwrite.m erzeugt. Die Infor-

mationen werden in drei Dateien gespeichert, deren Namen aus einem gemeinsamen Präfix und verschiedenen Endungen bestehen. Dieser Funktion wird (wie auch csrwrite.m) dieser Präfix übergeben.

Parameter:

filename Präfix des Dateinamens des Dateitripels, das eingelesen werden soll.

new_comm Kommunikator in die Prozessgruppe, der die neu erzeugte Matrix gehören soll.

Rückgabe:

Gibt die aus dem Dateitripel erzeugte DistEllpackMatrixGpu zurück.

Definiert in Zeile 233 der Datei distellpackmatrixgpu.tpp.

4.4.3.3 template<typename Scalar> bool Icarus::DistEllpackMatrixGpu< Scalar >::is_filled () const [inline]

Prüft, ob die Matrix korrekt gefüllt wurde. Ein positiver Rückgabewert dieser Funktion ist einerseits ein Indikator für einen erfolgreich abgeschlossenen Füllvorgang und andererseits die Voraussetzung für sämtliche algebraische Operationen mit der Matrix.

Rückgabe:

Gibt zurück, ob die Matrix korrekt gefüllt wurde.

Definiert in Zeile 151 der Datei distellpackmatrixgpu.hpp.

4.4.3.4 template<typename Scalar > DistEllpackMatrixGpu< Scalar > Icarus::DistEllpackMatrixGpu< Scalar >::precond_equi () const [inline]

Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer.

Rückgabe:

Der Vorkonditionierer hat denselben Typ wie das Objekt, auf das die Funktion aufgerufen wird.

Definiert in Zeile 319 der Datei distellpackmatrixgpu.tpp.

4.4.3.5 template<typename Scalar > DistEllpackMatrixGpu< Scalar > Icarus::DistEllpackMatrixGpu< Scalar >::precond_jacobi () const [inline]

Erstellt einen zu der Matrix passenden Äquilibrierungsvorkonditionierer. Wenn in einer Zeile eine Null auf der Diagonalen steht, wird diese Zeile durch die Vorkonditionierung nicht verändert.

Rückgabe:

Der Vorkonditionierer hat denselben Typ wie das Objekt, auf das die Funktion aufgerufen wird.

Definiert in Zeile 339 der Datei distellpackmatrixgpu.tpp.

4.4.3.6 template<typename Scalar > void Icarus::DistEllpackMatrixGpu< Scalar >::prepare_sequential_fill (size_t *max_row_length*) [inline]

Bereitet den zeilenweisen Füllvorgang der Matrix vor. Diese Funktion muss von jeder Node genau einmal zu Beginn des Füllvorgangs aufgerufen werden. Anschließend können die Zeilen mit sequential_fill und end_of_row gefüllt werden, beginnend bei der lokal ersten Zeile. Die maximale auf dieser Node auftretende Zeilenlänge muss vorher bekannt sein.

Parameter:

max_row_length Maximal auf dieser Node auftretende Zeilenlänge.

Definiert in Zeile 171 der Datei distellpackmatrixgpu.tpp.

4.4.3.7 template<typename Scalar > void Icarus::DistEllpackMatrixGpu< Scalar >::print_local_data (std::ostream & *os*) const [inline]

Schreibe den lokalen Inhalt des Block in den Stream out. Für die Verwendung dieser Funktion muss eine entsprechende Überladung des Operators std::ostream::operator<<(Scalar) existieren.

Parameter:

out Stream, in den die Ausgabe geschrieben werden soll.

Definiert in Zeile 366 der Datei distellpackmatrixgpu.tpp.

4.4.3.8 template<typename Scalar > void Icarus::DistEllpackMatrixGpu< Scalar >::sequential_fill (size_t *colind*, const Scalar & *val*) [inline]

Fülle die Matrix zeilenweise. Diese Funktion fügt der aktuellen Zeile den Wert val mit Spaltenindex col hinzu. Vor dem erste Aufruf dieser Funktion muss mit prepare_sequential_fill die maximal auf der füllenden Node auftretende Zeilenlänge gesetzt werden. Nachdem der letzte Eintrag einer Zeile gesetzt wurde, wird mit end_of_row die Zeile beendet.

Parameter:

colind Spaltenindex des einzutregenden Werts. Es muss colind < dim_global gelten.

val Wert, der an die Position colind geschrieben werden soll.

Definiert in Zeile 188 der Datei distellpackmatrixgpu.tpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrixgpu.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrixgpu.tpp

4.5 Icarus::Face Klassenreferenz

Eine Flaeche im Raum, aufgespannt von beliebig vielen Punkten. Der Benutzer muss selbst darauf achten, dass die Eckpunkte alle in einer Ebene liegen.

`#include <discretizer.hpp>`

Öffentliche Methoden

- Face (int num_vertices, std::vector< int > &id_vertices, std::vector< Vertex > &vertices, Vertex normal)

  Konstruktor.

- bool pointInsideYz (Vertex point)

  Ueberprueft, ob ein Punkt in der Projektion der Flaeche auf die Y-Z-Ebene liegt.

- Vertex get_vertex (int local_id)

  Gibt einen Eckpunkt der Fleache zureuck.

- Vertex get_normal ()

  Gibt Normalenvektor zurueck.

4.5.1 Ausführliche Beschreibung

Eine Flaeche im Raum, aufgespannt von beliebig vielen Punkten. Der Benutzer muss selbst darauf achten, dass die Eckpunkte alle in einer Ebene liegen.

Definiert in Zeile 23 der Datei discretizer.hpp.

4.5.2 Beschreibung der Konstruktoren und Destruktoren

4.5.2.1 Icarus::Face::Face (int *num_vertices*, std::vector< int > & *id_vertices*, std::vector< Vertex > & *vertices*, Vertex *normal*)

Konstruktor.

Parameter:

num_vertices Anzahl Eckpunkte, die die Flaeche aufspannen

id_vertices ID's geben an welche Punkte von 'vertices' zu der Flaeche gehoeren.

vertices Liste von Punkten, die mindestens die Eckpunkte der Flaeche enthaelt.

normal Normalenvektor zur Flaeche.

Definiert in Zeile 11 der Datei discretizer.cpp.

4.5.3 Dokumentation der Elementfunktionen

4.5.3.1 Vertex Icarus::Face::get_normal () `[inline]`

Gibt Normalenvektor zurueck.

Rückgabe:

Normalenvektor der Flaeche.

Definiert in Zeile 46 der Datei discretizer.hpp.

4.5.3.2 Vertex Icarus::Face::get_vertex (int *local_id*) `[inline]`

Gibt einen Eckpunkt der Fleache zureuck.

Parameter:

local_id ID des gewuenschten Punktes bzgl der Nummerierung der Punkte der Flaeche.

Rückgabe:

ID-ter Eckpunkt der Flaeche.

Definiert in Zeile 42 der Datei discretizer.hpp.

4.5.3.3 bool Icarus::Face::pointInsideYz (Vertex *point*)

Ueberprueft, ob ein Punkt in der Projektion der Flaeche auf die Y-Z-Ebene liegt.

Parameter:

point Raumpunkt der ueberprueft werden soll.

Rückgabe:

Gibt zurueck, ob der Punkt in der Projektion der Flaeche auf die Y-Z-Ebene liegt oder nicht.

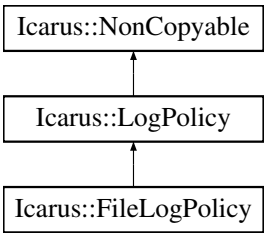
Definiert in Zeile 26 der Datei discretizer.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/discretizer.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/discretizer.cpp

4.6 Icarus::FileLogPolicy Klassenreferenz

Klassendiagramm für Icarus::FileLogPolicy::



Öffentliche Methoden

- virtual void **openLogStream** (const std::string &name="")
- virtual void **closeLogStream** ()
- virtual void **write** (const std::string &msg)
- virtual void **write_err** (const std::string &msg)

4.6.1 Ausführliche Beschreibung

Definiert in Zeile 74 der Datei logger.hpp.

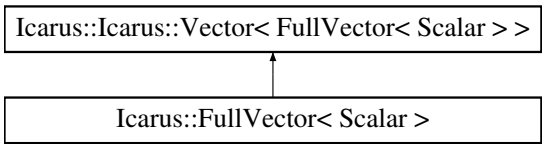
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/logger.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/logger.cpp

4.7 Icarus::FullVector< Scalar > Template-Klassenreferenz

Vektor, dessen Inhalt komplett auf jeder Node liegt.

`#include <fullvector.hpp>`Klassendiagramm für Icarus::FullVector< Scalar >::



Öffentliche Typen

- typedef Scalar **ScalarType**
- typedef ScalarTraits< Scalar >::RealType **RealType**

Öffentliche Methoden

- FullVector (size_t dim, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- FullVector (const SlicedVector< Scalar > &vec)

  Konvertierkonstruktor für einen SlicedVector.

- **FullVector** (const FullVector &other)
- **FullVector** (FullVector &&other)
- FullVector & **operator=** (const FullVector &other)
- FullVector & **operator=** (FullVector &&other)
- Scalar & operator[ ] (size_t index)

  Operator für den elementweisen Zugriff.

- const Scalar & operator[ ] (size_t index) const

  Operator für den elementweisen Zugriff, konstante Variante.

Freundbeziehungen

- class **Vector**< **FullVector**< **Scalar** > >

4.7.1 Ausführliche Beschreibung

template<typename Scalar> class Icarus::FullVector< Scalar >

Vektor, dessen Inhalt komplett auf jeder Node liegt. Alle Elemente dieses Vektors liegen auf jeder Node der zugeordneten Prozessgruppe.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 34 der Datei fullvector.hpp.

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 template<typename Scalar > Icarus::FullVector< Scalar >::FullVector (size_t *dim*, MPI_Comm *my_comm* = `MPI_COMM_WORLD`) `[inline, explicit]`

Standardkonstruktor. Erzeugt einen Vektor der Dimension dim, der komplett auf jeder Node der Prozessgruppe my_comm liegt.

Parameter:

dim Dimension des Vektors.

my_comm Kommunikator in die Prozessgruppe des Vektors.

Definiert in Zeile 25 der Datei fullvector.tpp.

4.7.2.2 template<typename Scalar> Icarus::FullVector< Scalar >::FullVector (const SlicedVector< Scalar > & *vec*) `[inline, explicit]`

Konvertierkonstruktor für einen SlicedVector. Erzeugt einen Vektor mit der globalen Dimension von vec, der alle Teile von vec lokal enthält. Alle Prozesse der Gruppe, die vec verwalten, erhalten eine vollständige Kopie des FullVectors.

Parameter:

vec SlicedVector, der vollständig verteilt werden soll.

Definiert in Zeile 44 der Datei fullvector.tpp.

4.7.3 Dokumentation der Elementfunktionen

4.7.3.1 template<typename Scalar> const Scalar& Icarus::FullVector< Scalar >::operator[ ] (size_t *index*) const `[inline]`

Operator für den elementweisen Zugriff, konstante Variante. Dieser Operator ermöglicht das elementweise Auslesen des FullVector, analog zu C-Arrays und STL-

Containern.

Parameter:

index Index des Elements, das gelesen werden soll.

Definiert in Zeile 99 der Datei fullvector.hpp.

4.7.3.2 template<typename Scalar> Scalar& Icarus::FullVector< Scalar >::operator[ ] (size_t *index*) `[inline]`

Operator für den elementweisen Zugriff. Dieser Operator ermöglicht die elementweise Manipulation des FullVector, analog zu C-Arrays und STL-Containern.

Parameter:

index Index des Elements, auf das zugegriffen werden soll.

Definiert in Zeile 89 der Datei fullvector.hpp.

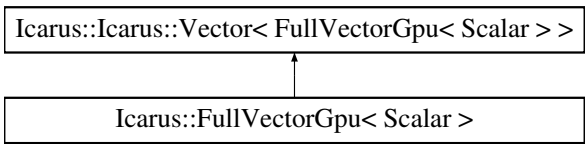
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvector.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvector.tpp

4.8 Icarus::FullVectorGpu< Scalar > Template-Klassenreferenz

Vektor, dessen Inhalt komplett auf jeder Node liegt.

`#include <fullvectorgpu.hpp>`Klassendiagramm für Icarus::FullVectorGpu< Scalar >::



Öffentliche Typen

- typedef Scalar **ScalarType**
- typedef ScalarTraits< Scalar >::RealType **RealType**

Öffentliche Methoden

- FullVectorGpu (size_t dim, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- FullVectorGpu (const SlicedVectorGpu< Scalar > &vec)

  Konvertierkonstruktor für einen SlicedVectorGpu.

- **FullVectorGpu** (const FullVectorGpu &other)
- **FullVectorGpu** (FullVectorGpu &&other)
- FullVectorGpu & **operator=** (const FullVectorGpu &other)
- FullVectorGpu & **operator=** (FullVectorGpu &&other)
- Scalar & operator[ ] (size_t index)

  Operator für den elementweisen Zugriff.

- const Scalar & operator[ ] (size_t index) const

  Operator für den elementweisen Zugriff, konstante Variante.

- Scalar * **getDataPointer** ()
- RealType **l2norm2_impl** () const
- RealType **maxnorm_impl** () const
- void **clear_impl** ()
- void **fill_const_impl** (const Scalar &s)
- Scalar **scal_prod_impl** (const FullVectorGpu &other) const
- void **axpy_impl** (const Scalar &alpha, const FullVectorGpu &y)
- void **scal_impl** (const Scalar &alpha)

- void **swap_impl** (FullVectorGpu &other)
- void **copy_impl** (const FullVectorGpu &other)
- size_t **get_dim_impl** () const

Freundbeziehungen

- class **Vector**< **FullVectorGpu**< **Scalar** > >

4.8.1 Ausführliche Beschreibung

template<typename Scalar> class Icarus::FullVectorGpu< Scalar >

Vektor, dessen Inhalt komplett auf jeder Node liegt. Alle Elemente dieses Vektors liegen auf jeder Node der zugeordneten Prozessgruppe.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 35 der Datei fullvectorgpu.hpp.

4.8.2 Beschreibung der Konstruktoren und Destruktoren

4.8.2.1 template<typename Scalar > Icarus::FullVectorGpu< Scalar >::FullVectorGpu (size_t *dim*, MPI_Comm *my_comm* = MPI_COMM_WORLD) [inline, explicit]

Standardkonstruktor. Erzeugt einen Vektor der Dimension dim, der komplett auf jeder Node der Prozessgruppe my_comm liegt.

Parameter:

dim Dimension des Vektors.

my_comm Kommunikator in die Prozessgruppe des Vektors.

Definiert in Zeile 38 der Datei fullvectorgpu.tpp.

4.8.2.2 template<typename Scalar> Icarus::FullVectorGpu< Scalar >::FullVectorGpu (const SlicedVectorGpu< Scalar > & *vec*) [inline, explicit]

Konvertierkonstruktor für einen SlicedVectorGpu. Erzeugt einen Vektor mit der globalen Dimension von vec, der alle Teile von vec lokal enthält. Alle Prozesse der Gruppe, die vec verwalten, erhalten eine vollständige Kopie des FullVectorGpus.

Parameter:

vec SlicedVectorGpu, der vollständig verteilt werden soll.

Definiert in Zeile 57 der Datei fullvectorgpu.tpp.

4.8.3 Dokumentation der Elementfunktionen

4.8.3.1 template<typename Scalar> const Scalar& Icarus::FullVectorGpu< Scalar >::operator[ ] (size_t *index*) const [inline]

Operator für den elementweisen Zugriff, konstante Variante. Dieser Operator ermöglicht das elementweise Auslesen des FullVectorGpu, analog zu C-Arrays und STL-Containern.

Parameter:

index Index des Elements, das gelesen werden soll.

Definiert in Zeile 102 der Datei fullvectorgpu.hpp.

4.8.3.2 template<typename Scalar> Scalar& Icarus::FullVectorGpu< Scalar >::operator[ ] (size_t *index*) [inline]

Operator für den elementweisen Zugriff. Dieser Operator ermöglicht die elementweise Manipulation des FullVectorGpu, analog zu C-Arrays und STL-Containern.

Parameter:

index Index des Elements, auf das zugegriffen werden soll.

Definiert in Zeile 92 der Datei fullvectorgpu.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvectorgpu.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvectorgpu.tpp

4.9 Icarus::Interface Klassenreferenz

4.9.1 Ausführliche Beschreibung

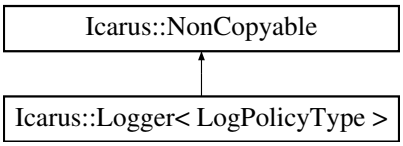
Definiert in Zeile 26 der Datei proto.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/proto.hpp

4.10 Icarus::Logger< LogPolicyType > Template-Klassenreferenz

Klassendiagramm für Icarus::Logger< LogPolicyType >::



Öffentliche Methoden

- **Logger** (const std::string &name="")
- template<SeverityType sev, typename... Args>
  void **print** (unsigned line, std::string file, Args...args)

Geschützte Methoden

- std::string **getLogInfo** ()

Statische geschützte Attribute

- static const unsigned **FIELD_WIDTH** = 7

4.10.1 Ausführliche Beschreibung

template<typename LogPolicyType> class Icarus::Logger< LogPolicyType >

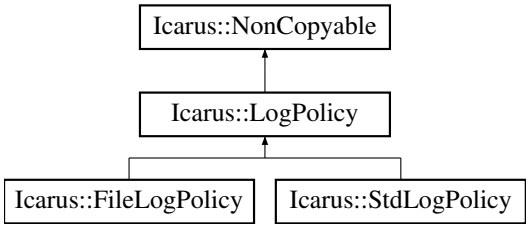
Definiert in Zeile 99 der Datei logger.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/logger.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/logger.tpp

4.11 Icarus::LogPolicy Klassenreferenz

Klassendiagramm für Icarus::LogPolicy::



Öffentliche Methoden

- virtual void **openLogStream** (const std::string &name="")=0
- virtual void **closeLogStream** ()=0
- virtual void **write** (const std::string &msg)=0
- virtual void **write_err** (const std::string &msg)=0

4.11.1 Ausführliche Beschreibung

Definiert in Zeile 48 der Datei logger.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/logger.hpp

4.12 Icarus::mathfunction Klassenreferenz

Öffentliche Methoden

- **mathfunction** (int type)
- double **eval** (double x, double y, double z)

4.12.1 Ausführliche Beschreibung

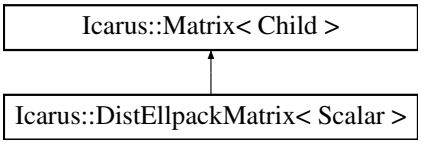
Definiert in Zeile 7 der Datei mathfunction.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/mathfunction.hpp

4.13 Icarus::Matrix< Child > Template-Klassenreferenz

Klassendiagramm für Icarus::Matrix< Child >::



Öffentliche Typen

- typedef MatrixTraits< Child >::RealType **RealType**
- typedef MatrixTraits< Child >::ScalarType **ScalarType**
- typedef MatrixTraits< Child >::VectorType **VectorType**

Öffentliche Methoden

- Child & **leaf** ()
- const Child & **leaf** () const
- void **mult_vec** (const Vector< VectorType > &x, Vector< VectorType > &res) const
- size_t **get_dim** () const

4.13.1 Ausführliche Beschreibung

template<class Child> class Icarus::Matrix< Child >

Definiert in Zeile 16 der Datei matrix.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/matrix.hpp

4.14 Icarus::MatrixTraits< DistEllpackMatrix< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**
- typedef SlicedVector< Scalar > **VectorType**

4.14.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::MatrixTraits< DistEllpackMatrix< Scalar > >

Definiert in Zeile 389 der Datei distellpackmatrix.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrix.tpp

4.15 Icarus::MatrixTraits< DistEllpackMatrixGpu< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**
- typedef SlicedVectorGpu< Scalar > **VectorType**

4.15.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::MatrixTraits< DistEllpackMatrixGpu< Scalar > >

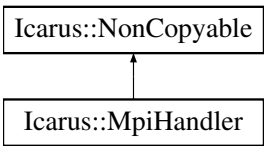
Definiert in Zeile 384 der Datei distellpackmatrixgpu.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/distellpackmatrixgpu.tpp

4.16 Icarus::MpiHandler Klassenreferenz

Klassendiagramm für Icarus::MpiHandler::



Öffentliche Methoden

- int **get_n_procs** () const
- bool **is_first** () const
- bool **is_last** () const
- int **get_my_rank** () const
- void **MpiSafeCall** (int line, std::string file, int error) const

4.16.1 Ausführliche Beschreibung

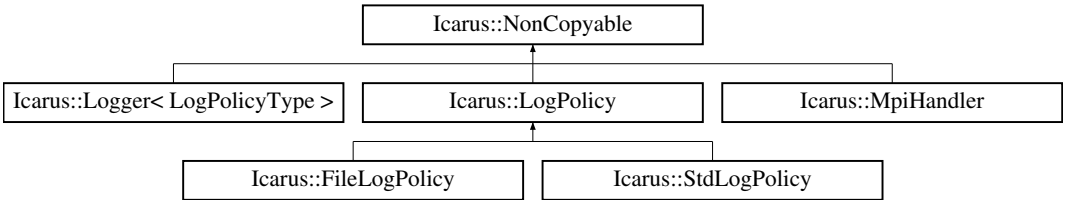
Definiert in Zeile 36 der Datei mpihandler.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/mpihandler.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/mpihandler.cpp

4.17 Icarus::NonCopyable Klassenreferenz

Klassendiagramm für Icarus::NonCopyable::



4.17.1 Ausführliche Beschreibung

Definiert in Zeile 17 der Datei proto.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/proto.hpp

4.18 Icarus::Object Klassenreferenz

Ein Objekt im Raum. Wird durch Flaechen aufgespannt. Ermoeglicht die Ueberpruefung, ob ein Raumpunkt im Inneren, auf dem Rand oder ausserhalb des Objektes liegt.

`#include <discretizer.hpp>`

Öffentliche Methoden

- Object (std::string name="")

  Konstruktor.

- void set_vertex (float x, float y, float z)

  Fuegt einen Raumpunkt hinzu.

- void set_normal (float x, float y, float z)

  Fuegt einen Normalenvektor hinzu, der spaeter einer Flaeche zugewiesen werden kann.

- void set_face (int num_vertices, std::vector< int > &id_vertices, int local_id_normal)

  Fuegt eine Flaeche hinzu. Benutzt werden dafuer bereits gespeicherte Punkte und Normalenvektoren. Es wird nicht ueberprueft, ob alle Punkte in einer Ebene liegen.

- char pointInside (Vertex point)

  Ueberprueft, ob Punkt in, auf oder ausserhalb vom Objekt liegt.

4.18.1 Ausführliche Beschreibung

Ein Objekt im Raum. Wird durch Flaechen aufgespannt. Ermoeglicht die Ueberpruefung, ob ein Raumpunkt im Inneren, auf dem Rand oder ausserhalb des Objektes liegt.

Definiert in Zeile 59 der Datei discretizer.hpp.

4.18.2 Beschreibung der Konstruktoren und Destruktoren

4.18.2.1 Icarus::Object::Object (std::string *name* = "")

Konstruktor.

Parameter:

name Optionaler Name des Objektes um daraus zB Material o.ae. abzuleiten.

Definiert in Zeile 54 der Datei discretizer.cpp.

4.18.3 Dokumentation der Elementfunktionen

4.18.3.1 char Icarus::Object::pointInside (Vertex *point*)

Ueberprueft, ob Punkt in, auf oder ausserhalb vom Objekt liegt.

Parameter:

point Zu ueberpruefender Punkt.

Rückgabe:

Gibt 'o' zurck, wenn sich der Punkt im Objekt befindet, sonst 'a'.

Definiert in Zeile 78 der Datei discretizer.cpp.

4.18.3.2 void Icarus::Object::set_face (int *num_vertices*, std::vector< int > & *id_vertices*, int *local_id_normal*)

Fuegt eine Flaeche hinzu. Benutzt werden dafuer bereits gespeicherte Punkte und Normalenvektoren. Es wird nicht ueberprueft, ob alle Punkte in einer Ebene liegen.

Parameter:

num_vertices Anzahl an Eckpunkten der Flaeche.

id_vertices Liste von ID's bzgl aller Punkte des Objektes, die die Flaeche aufspannen.

local_id_normal ID bzgl aller Normalenvektoren des Objektes, der der Flaeche zugeordnet werden soll.

Definiert in Zeile 72 der Datei discretizer.cpp.

4.18.3.3 void Icarus::Object::set_normal (float *x*, float *y*, float *z*)

Fuegt einen Normalenvektor hinzu, der spaeter einer Flaeche zugewiesen werden kann.

Parameter:

x x-Koordinate

y y-Koordinate

z z-Koordinate

Definiert in Zeile 67 der Datei discretizer.cpp.

4.18.3.4 void Icarus::Object::set_vertex (float *x*, float *y*, float *z*)

Fuegt einen Raumpunkt hinzu.

Parameter:

- ***x*** x-Koordinate
- ***y*** y-Koordinate
- ***z*** z-Koordinate

Definiert in Zeile 61 der Datei discretizer.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/discretizer.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/discretizer.cpp

4.19 Icarus::ScalarTraits< double > Template-Strukturreferenz

Öffentliche Typen

- typedef double **RealType**

Öffentliche, statische Methoden

- static double **abs2** (double d)
- static double **abs** (double d)
- static double **smult** (double d1, double d2)

Statische öffentliche Attribute

- static constexpr MPI_Datatype **mpi_type** = MPI_DOUBLE

4.19.1 Ausführliche Beschreibung

template<> struct Icarus::ScalarTraits< double >

Definiert in Zeile 23 der Datei scalartraits.hpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/scalartraits.hpp

4.20 Icarus::ScalarTraits< float > Template-Strukturreferenz

Öffentliche Typen

- typedef float **RealType**

Öffentliche, statische Methoden

- static float **abs2** (float f)
- static float **abs** (float f)
- static float **smult** (float f1, float f2)

Statische öffentliche Attribute

- static constexpr MPI_Datatype **mpi_type** = MPI_FLOAT

4.20.1 Ausführliche Beschreibung

template<> struct Icarus::ScalarTraits< float >

Definiert in Zeile 33 der Datei scalartraits.hpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/scalartraits.hpp

4.21 Icarus::ScalarTraits< std::complex< double > > Template-Strukturreferenz

Öffentliche Typen

- typedef double **RealType**

Öffentliche, statische Methoden

- static double **abs2** (const std::complex< double > &c)
- static double **abs** (const std::complex< float > &c)
- static std::complex< double > **smult** (const std::complex< double > &c1, const std::complex< double > &c2)

Statische öffentliche Attribute

- static constexpr MPI_Datatype **mpi_type** = MPI_DOUBLE_COMPLEX

4.21.1 Ausführliche Beschreibung

template<> struct Icarus::ScalarTraits< std::complex< double > >

Definiert in Zeile 56 der Datei scalartraits.hpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/scalartraits.hpp

4.22 Icarus::ScalarTraits< std::complex< float > > Template-Strukturreferenz

Öffentliche Typen

- typedef float **RealType**

Öffentliche, statische Methoden

- static float **abs2** (const std::complex< float > &c)
- static double **abs** (const std::complex< float > &c)
- static std::complex< float > **smult** (const std::complex< float > &c1, const std::complex< float > &c2)

Statische öffentliche Attribute

- static constexpr MPI_Datatype **mpi_type** = MPI_COMPLEX

4.22.1 Ausführliche Beschreibung

template<> struct Icarus::ScalarTraits< std::complex< float > >

Definiert in Zeile 43 der Datei scalartraits.hpp.

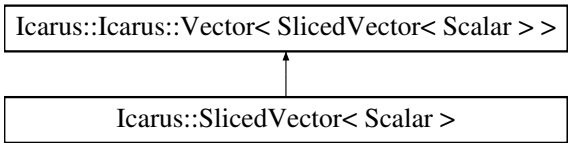
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/scalartraits.hpp

4.23 Icarus::SlicedVector< Scalar > Template-Klassenreferenz

Vektor, dessen Inhalt gleichverteilt auf einer Menge von Nodes liegt.

`#include <slicedvector.hpp>`Klassendiagramm für Icarus::SlicedVector< Scalar >::



Öffentliche Typen

- typedef Scalar **ScalarType**
- typedef ScalarTraits< Scalar >::RealType **RealType**

Öffentliche Methoden

- SlicedVector (size_t dim_global, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- **SlicedVector** (const SlicedVector &other)
- **SlicedVector** (SlicedVector &&other)
- SlicedVector & **operator=** (const SlicedVector &other)
- SlicedVector & **operator=** (SlicedVector &&other)
- void set_global (size_t pos, const Scalar &val)

  Setze den Wert val an die globale Position pos.

- Scalar get_global (size_t pos) const

  Hole den Eintrag an der globalen Position pos.

- size_t get_dim_global () const

  Gibt die globale Dimension des Vektors zurück.

- size_t get_dim_local () const

  Gibt die lokale Dimension des Vektors, d.h. die Größe des auf der aufrufenden Node gespeicherten Blocks zurück.

- size_t get_dim_local_nopad () const

  Gibt die Größe eines Blocks, wie er auf den ersten N-1 Nodes liegt, zurück.

- size_t get_dim_local_last () const

Gibt die Größe des Blocks, der auf der letzten Node liegt, zurück.

- MPI_Comm get_comm () const
  Gibt den MPI-Kommunikator in doe Prozessgrupe, der der Vektor gehört, zurück.
- void print_local_data (std::ostream &out) const
  Schreibe den lokalen Inhalt des Block in den Stream out.
- void set_local (size_t pos, const Scalar &val)
  Setze den Wert val an die lokale Position pos.
- Scalar get_local (size_t pos) const
  Hole den Eintrag an der lokalen Position pos.

Freundbeziehungen

- class **Vector< SlicedVector< Scalar > >**

4.23.1 Ausführliche Beschreibung

template<typename Scalar> class Icarus::SlicedVector< Scalar >

Vektor, dessen Inhalt gleichverteilt auf einer Menge von Nodes liegt. Die Elemente dieses Vektors liegen (annähernd) gleichverteilt auf einer Menge von Nodes der zugeordneten Prozessgruppe.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 39 der Datei slicedvector.hpp.

4.23.2 Beschreibung der Konstruktoren und Destruktoren

4.23.2.1 template<typename Scalar > Icarus::SlicedVector< Scalar >::SlicedVector (size_t *dim_global*, MPI_Comm *my_comm* = MPI_COMM_WORLD) [inline, explicit]

Standardkonstruktor. Konstuiert einen Vektor, dessen Elemente auf den Nodes der Prozessgruppe my_comm verteilt werden. Die ersten N-1 Nodes enthalten die gleiche Anzahl an Elementen, die letzte Node enthält den (eventuell etwas kleineren) Rest.

Parameter:

dim_global Globale Dimension des Vektors, also Summe der Größen der auf die Nodes verteilten Blöcke.

Definiert in Zeile 26 der Datei slicedvector.tpp.

4.23.3 Dokumentation der Elementfunktionen

4.23.3.1 template<typename Scalar > Scalar Icarus::SlicedVector< Scalar >::get_global (size_t *pos*) const [inline]

Hole den Eintrag an der globalen Position pos. Hole den Wert an der globale Position pos. Diese Operation erfordert MPI-Kommunikation, wenn die zu lesende Position nicht auf der Node liegt, die den Befehl ausführt, und ist daher hinsichtlich Effizienz mit Vorsicht zu benutzen.

Parameter:

pos Globale Position des Elements, das gelesen werden soll.

Rückgabe:

Eintrag an der globalen Position pos.

Definiert in Zeile 155 der Datei slicedvector.tpp.

4.23.3.2 template<typename Scalar> Scalar Icarus::SlicedVector< Scalar >::get_local (size_t *pos*) const [inline]

Hole den Eintrag an der lokalen Position pos. Hole den Wert an der lokalen Position pos. Diese Operation erfordert keine MPI-Kommunikation.

Parameter:

pos Lokale Position des Elements, das gelesen werden soll.

Rückgabe:

Eintrag an der lokalen Position pos.

Definiert in Zeile 167 der Datei slicedvector.hpp.

4.23.3.3 template<typename Scalar > void Icarus::SlicedVector< Scalar >::print_local_data (std::ostream & *out*) const [inline]

Schreibe den lokalen Inhalt des Block in den Stream out. Für die Verwendung dieser Funktion muss eine entsprechende Überladung des Operators std::ostream::operator<<(Scalar) existieren.

Parameter:

out Stream, in den die Ausgabe geschrieben werden soll.

Definiert in Zeile 169 der Datei slicedvector.tpp.

4.23.3.4 template<typename Scalar> void Icarus::SlicedVector< Scalar >::set_global (size_t *pos*, const Scalar & *val*) [inline]

Setze den Wert val an die globale Position pos. Setze den Wert val an die globale Position pos. Diese Operation erfordert MPI-Kommunikation, wenn die zu setzende Position nicht auf der Node liegt, die den Befehl ausführt, und ist daher hinsichtlich Effizienz mit Vorsicht zu benutzen.

Parameter:

pos Globale Position des Elements, das gesetzt werden soll.

val Wert, der an die Stelle pos kopiert werden soll.

Definiert in Zeile 146 der Datei slicedvector.tpp.

4.23.3.5 template<typename Scalar> void Icarus::SlicedVector< Scalar >::set_local (size_t *pos*, const Scalar & *val*) [inline]

Setze den Wert val an die lokale Position pos. Setze den Wert val an die lokale Position pos. Diese Operation erfordert keine MPI-Kommunikation.

Parameter:

pos Lokale Position des Elements, das gesetzt werden soll.

val Wert, der an die Stelle pos kopiert werden soll.

Definiert in Zeile 152 der Datei slicedvector.hpp.

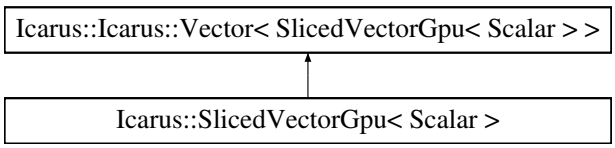
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvector.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvector.tpp

4.24 Icarus::SlicedVectorGpu< Scalar > Template-Klassenreferenz

Vektor, dessen Inhalt gleichverteilt auf einer Menge von Nodes liegt.

`#include <slicedvectorgpu.hpp>`Klassendiagramm für Icarus::SlicedVectorGpu< Scalar >::



Öffentliche Typen

- typedef Scalar **ScalarType**
- typedef ScalarTraits< Scalar >::RealType **RealType**

Öffentliche Methoden

- SlicedVectorGpu (size_t dim_global, MPI_Comm my_comm=MPI_COMM_WORLD)

  Standardkonstruktor.

- **SlicedVectorGpu** (const SlicedVectorGpu &other)
- **SlicedVectorGpu** (SlicedVectorGpu &&other)
- SlicedVectorGpu & **operator=** (const SlicedVectorGpu &other)
- SlicedVectorGpu & **operator=** (SlicedVectorGpu &&other)
- void set_global (size_t pos, const Scalar &val)

  Setze den Wert val an die globale Position pos.

- Scalar get_global (size_t pos) const

  Hole den Eintrag an der globalen Position pos.

- size_t get_dim_global () const

  Gibt die globale Dimension des Vektors zurück.

- size_t get_dim_local () const

  Gibt die lokale Dimension des Vektors, d.h. die Größe des auf der aufrufenden Node gespeicherten Blocks zurück.

- size_t get_dim_local_nopad () const

  Gibt die Größe eines Blocks, wie er auf den ersten N-1 Nodes liegt, zurück.

- size_t get_dim_local_last () const

  Gibt die Größe des Blocks, der auf der letzten Node liegt, zurück.

- MPI_Comm get_comm () const

  Gibt den MPI-Kommunikator in doe Prozessgrupe, der der Vektor gehört, zurück.

- void print_local_data (std::ostream &out) const

  Schreibe den lokalen Inhalt des Block in den Stream out.

- void set_local (size_t pos, const Scalar &val)

  Setze den Wert val an die lokale Position pos.

- Scalar get_local (size_t pos) const

  Hole den Eintrag an der lokalen Position pos.

- Scalar * **getDataPointer** ()

Freundbeziehungen

- class **Vector**< **SlicedVectorGpu**< **Scalar** > >

4.24.1 Ausführliche Beschreibung

template<**typename Scalar**> **class Icarus::SlicedVectorGpu**< **Scalar** >

Vektor, dessen Inhalt gleichverteilt auf einer Menge von Nodes liegt. Die Elemente dieses Vektors liegen (annähernd) gleichverteilt auf einer Menge von Nodes der zugeordneten Prozessgruppe.

Template Parameters:

Scalar Skalarer Typ der Einträge.

Definiert in Zeile 40 der Datei slicedvectorgpu.hpp.

4.24.2 Beschreibung der Konstruktoren und Destruktoren

4.24.2.1 template<typename Scalar > Icarus::SlicedVectorGpu< Scalar >::SlicedVectorGpu (size_t *dim_global*, MPI_Comm *my_comm* = MPI_COMM_WORLD) [inline, explicit]

Standardkonstruktor. Konstuiert einen Vektor, dessen Elemente auf den Nodes der Prozessgruppe my_comm verteilt werden. Die ersten N-1 Nodes enthalten die gleiche Anzahl an Elementen, die letzte Node enthält den (eventuell etwas kleineren) Rest.

Parameter:

dim_global Globale Dimension des Vektors, also Summe der Größen der auf die Nodes verteilten Blöcke.

Definiert in Zeile 39 der Datei slicedvectorgpu.tpp.

4.24.3 Dokumentation der Elementfunktionen

4.24.3.1 template<typename Scalar > Scalar Icarus::SlicedVectorGpu< Scalar >::get_global (size_t *pos*) const [inline]

Hole den Eintrag an der globalen Position pos. Hole den Wert an der globale Position pos. Diese Operation erfordert MPI-Kommunikation, wenn die zu lesende Position nicht auf der Node liegt, die den Befehl ausführt, und ist daher hinsichtlich Effizienz mit Vorsicht zu benutzen.

Parameter:

pos Globale Position des Elements, das gelesen werden soll.

Rückgabe:

Eintrag an der globalen Position pos.

Definiert in Zeile 168 der Datei slicedvectorgpu.tpp.

4.24.3.2 template<typename Scalar> Scalar Icarus::SlicedVectorGpu< Scalar >::get_local (size_t *pos*) const [inline]

Hole den Eintrag an der lokalen Position pos. Hole den Wert an der lokalen Position pos. Diese Operation erfordert keine MPI-Kommunikation.

Parameter:

pos Lokale Position des Elements, das gelesen werden soll.

Rückgabe:

Eintrag an der lokalen Position pos.

Definiert in Zeile 170 der Datei slicedvectorgpu.hpp.

4.24.3.3 template<typename Scalar > void Icarus::SlicedVectorGpu< Scalar >::print_local_data (std::ostream & *out*) const [inline]

Schreibe den lokalen Inhalt des Block in den Stream out. Für die Verwendung dieser Funktion muss eine entsprechende Überladung des Operators std::ostream::operator<<(Scalar) existieren.

Parameter:

out Stream, in den die Ausgabe geschrieben werden soll.

Definiert in Zeile 182 der Datei slicedvectorgpu.tpp.

4.24.3.4 template<typename Scalar> void Icarus::SlicedVectorGpu< Scalar >::set_global (size_t *pos*, const Scalar & *val*) [inline]

Setze den Wert val an die globale Position pos. Setze den Wert val an die globale Position pos. Diese Operation erfordert MPI-Kommunikation, wenn die zu setzende Position nicht auf der Node liegt, die den Befehl ausführt, und ist daher hinsichtlich Effizienz mit Vorsicht zu benutzen.

Parameter:

pos Globale Position des Elements, das gesetzt werden soll.

val Wert, der an die Stelle pos kopiert werden soll.

Definiert in Zeile 159 der Datei slicedvectorgpu.tpp.

4.24.3.5 template<typename Scalar> void Icarus::SlicedVectorGpu< Scalar >::set_local (size_t *pos*, const Scalar & *val*) [inline]

Setze den Wert val an die lokale Position pos. Setze den Wert val an die lokale Position pos. Diese Operation erfordert keine MPI-Kommunikation.

Parameter:

pos Lokale Position des Elements, das gesetzt werden soll.

val Wert, der an die Stelle pos kopiert werden soll.

Definiert in Zeile 155 der Datei slicedvectorgpu.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvectorgpu.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvectorgpu.tpp

4.25 Icarus::Solver< Child > Template-Klassenreferenz

Öffentliche Typen

- typedef SolverTraits< Child >::VectorType **VectorType**

Öffentliche Methoden

- Child & **leaf** ()
- const Child & **leaf** () const
- void **solve** (Vector< VectorType > &dest)

4.25.1 Ausführliche Beschreibung

template<class Child> class Icarus::Solver< Child >

Definiert in Zeile 148 der Datei solver.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/solver.hpp

4.26 Icarus::SolverTraits< BiCgStabSolver< MatrixT > > Template-Strukturreferenz

Öffentliche Typen

- typedef MatrixT::VectorType **VectorType**

4.26.1 Ausführliche Beschreibung

template<typename MatrixT> struct Icarus::SolverTraits< BiCgStabSolver< MatrixT > >

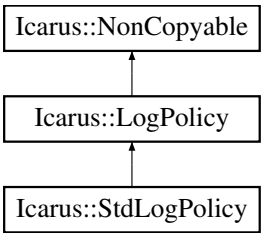
Definiert in Zeile 143 der Datei bicgstabsolver.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/bicgstabsolver.tpp

4.27 Icarus::StdLogPolicy Klassenreferenz

Klassendiagramm für Icarus::StdLogPolicy::



Öffentliche Methoden

- virtual void **openLogStream** (const std::string &name="")
- virtual void **closeLogStream** ()
- virtual void **write** (const std::string &msg)
- virtual void **write_err** (const std::string &msg)

4.27.1 Ausführliche Beschreibung

Definiert in Zeile 61 der Datei logger.hpp.

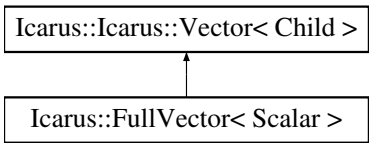
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/logger.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/logger.cpp

4.28 Icarus::Icarus::Vector< Child > Template-Klassenreferenz

Basisklasse (Interface) für alle Vektortypen.

`#include <solver.hpp>`Klassendiagramm für Icarus::Icarus::Vector< Child >::



Öffentliche Typen

- typedef VectorTraits< Child >::RealType **RealType**
- typedef VectorTraits< Child >::ScalarType **ScalarType**

Öffentliche Methoden

- Child & leaf ()

  Zugriff auf den abgeleiteten Typ.

- const Child & leaf () const

  Zugriff auf den abgeleiteten Typ, konstante Variante.

- RealType l2norm2 () const

  Berechnet das Quadrat der L2-Norm des Vektors.

- RealType l2norm () const

  Berechnet die L2-Norm des Vektors.

- RealType maxnorm () const

  Berechnet die Maximum-Norm des Vektors.

- void clear ()

  Füllt den gesamten Vektor mit Scalar(0).

- void fill_const (const ScalarType &s)

  Füllt den gesamten Vektor mit Kopien von s.

- ScalarType scal_prod (const Vector< Child > &other)

  Berechne das Skalarprodukt mit einem zweiten Vektor desselben Typs.

- void axpy (const ScalarType &alpha, const Vector< Child > &y)

  Berechne die BLAS-Operation x <- x + alpha y.

- void scal (const ScalarType &alpha)

  Skaliere den Vektor um einen konstanten Skalar.

- size_t get_dim () const

  Gibt die Dimension des Vektors zurück.

- void copy (const Vector< Child > &other)

  Explizite Kopieroperation.

- void swap (Vector< Child > &other)

  Explizite Tauschoperation.

4.28.1 Ausführliche Beschreibung

template<class Child> class Icarus::Icarus::Vector< Child >

Basisklasse (Interface) für alle Vektortypen. Diese Klasse definiert das Interface Vector, das alle Operationen vorstellt, die ein Vektor implementieren muss. Die entsprechenden Implementierungen werden als private Methoden der abgeleiteten Klassen mit dem Suffix "_impl" bereitgestellt.

Template Parameters:

Child Der Typ der abgeleiteten Klasse (siehe CRTP).

Definiert in Zeile 40 der Datei solver.hpp.

4.28.2 Dokumentation der Elementfunktionen

4.28.2.1 template<class Child> void Icarus::Icarus::Vector< Child >::axpy (const ScalarType & *alpha*, const Vector< Child > & *y*) [inline]

Berechne die BLAS-Operation x <- x + alpha y.

Parameter:

alpha Der Skalar, mit dem der zweite Vektor multipliziert wird.

y Der zweite Vektor.

Definiert in Zeile 104 der Datei solver.hpp.

4.28.2.2 template<class Child> void Icarus::Icarus::Vector< Child >::copy (const Vector< Child > & *other*) [inline]

Explizite Kopieroperation. Kopiert den Inhalt eines zweiten Vektors in den Vektor.

Parameter:

other Vektor, dessen Inhalt in diesen Vektor kopiert wird.

Definiert in Zeile 127 der Datei solver.hpp.

4.28.2.3 template<class Child> void Icarus::Icarus::Vector< Child >::fill_const (const ScalarType & *s*) [inline]

Füllt den gesamten Vektor mit Kopien von s.

Parameter:

s Objekt, das an alle Positionen des Vektors geschrieben werden soll.

Definiert in Zeile 87 der Datei solver.hpp.

4.28.2.4 template<class Child> void Icarus::Icarus::Vector< Child >::scal (const ScalarType & *alpha*) [inline]

Skaliere den Vektor um einen konstanten Skalar. Multipliziere alle Komponenten des Vektors mit alpha.

Parameter:

alpha Der Skalar, mit dem der Vektor multipliziert werden.

Definiert in Zeile 113 der Datei solver.hpp.

4.28.2.5 template<class Child> ScalarType Icarus::Icarus::Vector< Child >::scal_prod (const Vector< Child > & *other*) [inline]

Berechne das Skalarprodukt mit einem zweiten Vektor desselben Typs. Bei komplexen Datentypen wird der zweite Faktor automatisch konjugiert.

Parameter:

other Der zweite Vektor in dem Skalarprodukt.

Definiert in Zeile 96 der Datei solver.hpp.

4.28.2.6 template<class Child> void Icarus::Icarus::Vector< Child >::swap (Vector< Child > & *other*) `[inline]`

Explizite Tauschoperation. Tauscht den Inhalt eines zweiten Vektors mit diesem Vektor.

Parameter:

other Vektor, dessen Inhalt mit diesem Vektor vertauscht wird.

Definiert in Zeile 136 der Datei solver.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/solver.hpp

4.29 Icarus::Vector< Child > Template-Klassenreferenz

Basisklasse (Interface) für alle Vektortypen.

`#include <vector.hpp>`

Öffentliche Typen

- typedef VectorTraits< Child >::RealType **RealType**
- typedef VectorTraits< Child >::ScalarType **ScalarType**

Öffentliche Methoden

- Child & leaf ()

  Zugriff auf den abgeleiteten Typ.

- const Child & leaf () const

  Zugriff auf den abgeleiteten Typ, konstante Variante.

- RealType l2norm2 () const

  Berechnet das Quadrat der L2-Norm des Vektors.

- RealType l2norm () const

  Berechnet die L2-Norm des Vektors.

- RealType maxnorm () const

  Berechnet die Maximum-Norm des Vektors.

- void clear ()

  Füllt den gesamten Vektor mit Scalar(0).

- void fill_const (const ScalarType &s)

  Füllt den gesamten Vektor mit Kopien von s.

- ScalarType scal_prod (const Vector< Child > &other)

  Berechne das Skalarprodukt mit einem zweiten Vektor desselben Typs.

- void axpy (const ScalarType &alpha, const Vector< Child > &y)

  Berechne die BLAS-Operation x <- x + alpha y.

- void scal (const ScalarType &alpha)

  Skaliere den Vektor um einen konstanten Skalar.

- size_t get_dim () const

Gibt die Dimension des Vektors zurück.

- void copy (const Vector< Child > &other)
  Explizite Kopieroperation.
- void swap (Vector< Child > &other)
  Explizite Tauschoperation.

4.29.1 Ausführliche Beschreibung

template<class Child> class Icarus::Vector< Child >

Basisklasse (Interface) für alle Vektortypen. Diese Klasse definiert das Interface Vector, das alle Operationen vorstellt, die ein Vektor implementieren muss. Die entsprechenden Implementierungen werden als private Methoden der abgeleiteten Klassen mit dem Suffix "_impl" bereitgestellt.

Template Parameters:

Child Der Typ der abgeleiteten Klasse (siehe CRTP).

Definiert in Zeile 34 der Datei vector.hpp.

4.29.2 Dokumentation der Elementfunktionen

4.29.2.1 template<class Child> void Icarus::Vector< Child >::axpy (const ScalarType & *alpha*, const Vector< Child > & *y*) [inline]

Berechne die BLAS-Operation x <- x + alpha y.

Parameter:

alpha Der Skalar, mit dem der zweite Vektor multipliziert wird.

y Der zweite Vektor.

Definiert in Zeile 98 der Datei vector.hpp.

4.29.2.2 template<class Child> void Icarus::Vector< Child >::copy (const Vector< Child > & *other*) [inline]

Explizite Kopieroperation. Kopiert den Inhalt eines zweiten Vektors in den Vektor.

Parameter:

other Vektor, dessen Inhalt in diesen Vektor kopiert wird.

Definiert in Zeile 121 der Datei vector.hpp.

4.29.2.3 template<class Child> void Icarus::Vector< Child >::fill_const (const ScalarType & *s*) [inline]

Füllt den gesamten Vektor mit Kopien von s.

Parameter:

s Objekt, das an alle Positionen des Vektors geschrieben werden soll.

Definiert in Zeile 81 der Datei vector.hpp.

4.29.2.4 template<class Child> void Icarus::Vector< Child >::scal (const ScalarType & *alpha*) [inline]

Skaliere den Vektor um einen konstanten Skalar. Multipliziere alle Komponenten des Vektors mit alpha.

Parameter:

alpha Der Skalar, mit dem der Vektor multipliziert werden.

Definiert in Zeile 107 der Datei vector.hpp.

4.29.2.5 template<class Child> ScalarType Icarus::Vector< Child >::scal_prod (const Vector< Child > & *other*) [inline]

Berechne das Skalarprodukt mit einem zweiten Vektor desselben Typs. Bei komplexen Datentypen wird der zweite Faktor automatisch konjugiert.

Parameter:

other Der zweite Vektor in dem Skalarprodukt.

Definiert in Zeile 90 der Datei vector.hpp.

4.29.2.6 template<class Child> void Icarus::Vector< Child >::swap (Vector< Child > & *other*) [inline]

Explizite Tauschoperation. Tauscht den Inhalt eines zweiten Vektors mit diesem Vektor.

Parameter:

other Vektor, dessen Inhalt mit diesem Vektor vertauscht wird.

Definiert in Zeile 130 der Datei vector.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/vector.hpp

4.30 Icarus::VectorTraits< FullVector< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**

4.30.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::VectorTraits< FullVector< Scalar > >

Definiert in Zeile 225 der Datei fullvector.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvector.tpp

4.31 Icarus::VectorTraits< FullVectorGpu< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**

4.31.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::VectorTraits< FullVectorGpu< Scalar > >

Definiert in Zeile 238 der Datei fullvectorgpu.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/fullvectorgpu.tpp

4.32 Icarus::VectorTraits< SlicedVector< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**

4.32.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::VectorTraits< SlicedVector< Scalar > >

Definiert in Zeile 260 der Datei slicedvector.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvector.tpp

4.33 Icarus::VectorTraits< SlicedVectorGpu< Scalar > > Template-Strukturreferenz

Öffentliche Typen

- typedef ScalarTraits< Scalar >::RealType **RealType**
- typedef Scalar **ScalarType**

4.33.1 Ausführliche Beschreibung

template<typename Scalar> struct Icarus::VectorTraits< SlicedVectorGpu< Scalar > >

Definiert in Zeile 273 der Datei slicedvectorgpu.tpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/slicedvectorgpu.tpp

4.34 Icarus::Vertex Strukturreferenz

Simple Struktur um einen Raumpunkt kompakt zu speichern. Kann ebenso dazu benutzt werden einen Vektor zu speichern.

`#include <discretizer.hpp>`

Datenfelder

- float **x**
- float **y**
- float **z**

4.34.1 Ausführliche Beschreibung

Simple Struktur um einen Raumpunkt kompakt zu speichern. Kann ebenso dazu benutzt werden einen Vektor zu speichern.

Definiert in Zeile 17 der Datei discretizer.hpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/discretizer.hpp

4.35 Icarus::vtkWriter Klassenreferenz

Öffentliche Methoden

- vtkWriter (std::string filename, std::string title, size_t xdim, size_t ydim, size_t zdim, size_t timesteps)

  Konstruktor.

- vtkWriter (std::string filename, std::string title, size_t xdim, size_t ydim, size_t zdim, double h, size_t timesteps)

  Konstruktor.

- ~vtkWriter ()

  Destruktor.

- template<typename type >
  void addPointDataToTimestep (const type data[ ], const size_t length, const size_t timestep, std::string name)

  Fuegt skalarwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointDataToTimestep (const FullVector< type > &data, const size_t timestep, const std::string name)

  Fuegt skalarwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellDataToTimestep (const type data[ ], const size_t length, const size_t timestep, const std::string name)

  Fuegt skalarwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellDataToTimestep (const FullVector< type > &data, const size_t timestep, const std::string name)

  Fuegt skalarwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointDataToAll (const type data[ ], size_t length, std::string name)

  Fuegt skalarwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointDataToAll (const FullVector< type > &data, const std::string name)

Fuegt skalarwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellDataToAll (const type data[ ], size_t length, std::string name)

  Fuegt skalarwertige Zelldaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellDataToAll (const FullVector< type > &data, const std::string name)

  Fuegt skalarwertige Zelldaten zu allen Zeitschritten hinzu Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointVecToTimestep (const type datax[ ], const type datay[ ], const type dataz[ ], const size_t length, const size_t timestep, std::string name)

  Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointVecToTimestep (const FullVector< type > &datax, const FullVector< type > &datay, const FullVector< type > &dataz, const size_t timestep, const std::string name)

  Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellVecToTimestep (const type datax[ ], const type datay[ ], const type dataz[ ], const size_t length, const size_t timestep, std::string name)

  Fuegt vektorwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellVecToTimestep (const FullVector< type > &datax, const FullVector< type > &datay, const FullVector< type > &dataz, const size_t timestep, const std::string name)

  Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointVecToAll (const type datax[ ], const type datay[ ], const type dataz[ ], size_t length, std::string name)

  Fuegt vektorwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addPointVecToAll (const FullVector< type > &datax, const FullVector< type > &datay, const FullVector< type > &dataz, const std::string name)

Fuegt vektorwertige Punktdaten zu allen Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellVecToAll (const type datax[ ], const type datay[ ], const type dataz[ ], size_t length, std::string name)

  Fuegt vektorwertige Zelldaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

- template<typename type >
  void addCellVecToAll (const FullVector< type > &datax, const FullVector< type > &datay, const FullVector< type > &dataz, const std::string name)

  Fuegt vektorwertige Zelldaten zu allen Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

4.35.1 Ausführliche Beschreibung

Definiert in Zeile 24 der Datei vtkwriter.hpp.

4.35.2 Beschreibung der Konstruktoren und Destruktoren

4.35.2.1 Icarus::vtkWriter::vtkWriter (std::string *filename*, std::string *title*, size_t *xdim*, size_t *ydim*, size_t *zdim*, size_t *timesteps*)

Konstruktor.

Parameter:

filename string mit dem Dateinamen OHNE DATEIENDUNG

title Titel der Datei

xdim Anzahl der Punkte in x Richtung

ydim Anzahl der Punkte in y Richtung

zdim Anzahl der Punkte in z Richtung

timesteps Anzahl der Zeitschritte

Definiert in Zeile 6 der Datei vtkwriter.cpp.

4.35.2.2 Icarus::vtkWriter::vtkWriter (std::string *filename*, std::string *title*, size_t *xdim*, size_t *ydim*, size_t *zdim*, double *h*, size_t *timesteps*)

Konstruktor.

Parameter:

filename string mit dem Dateinamen OHNE DATEIENDUNG

title Titel der Datei

xdim Anzahl der Punkte in x Richtung

ydim Anzahl der Punkte in y Richtung

zdim Anzahl der Punkte in z Richtung

h Ortschrittweite in alle Richtungen

timesteps Anzahl der Zeitschritte

Definiert in Zeile 41 der Datei vtkwriter.cpp.

4.35.3 Dokumentation der Elementfunktionen

4.35.3.1 template<typename type > void Icarus::vtkWriter::addCellDataToAll (const FullVector< type > & *data*, const std::string *name*) [inline]

Fuegt skalarwertige Zelldaten zu allen Zeitschritten hinzu Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data Vektor mit den zu schreibenden Daten

name Name der Daten

Definiert in Zeile 366 der Datei vtkwriter.tpp.

4.35.3.2 template<typename type > void Icarus::vtkWriter::addCellDataToAll (const type *data*[ ], size_t *length*, std::string *name*) [inline]

Fuegt skalarwertige Zelldaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data[ ] Array der Quelldaten

length Länge des Arrays

name Name der Daten

Definiert in Zeile 354 der Datei vtkwriter.tpp.

4.35.3.3 template<typename type > void Icarus::vtkWriter::addCellDataToTimestep (const FullVector< type > & *data*, const size_t *timestep*, const std::string *name*) [inline]

Fuegt skalarwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data Vektor mit den zu schreibenden Daten

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 165 der Datei vtkwriter.tpp.

4.35.3.4 template<typename type > void Icarus::vtkWriter::addCellDataToTimestep (const type *data*[ ], const size_t *length*, const size_t *timestep*, const std::string *name*) [inline]

Fuegt skalarwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data[ ] Array der Quelldaten

length Länge des Arrays

timestep Zeitschritt, dem die Daten hinzugefügt werden

name Name der Daten

Definiert in Zeile 205 der Datei vtkwriter.tpp.

4.35.3.5 template<typename type > void Icarus::vtkWriter::addCellVecToAll (const FullVector< type > & *datax*, const FullVector< type > & *datay*, const FullVector< type > & *dataz*, const std::string *name*) [inline]

Fuegt vektorwertige Zelldaten zu allen Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax Vektor mit den x Komponenten der zu schreibenden Daten

datay Vektor mit den y Komponenten der zu schreibenden Daten

dataz Vektor mit den z Komponenten der zu schreibenden Daten

name Name der Daten

Definiert in Zeile 405 der Datei vtkwriter.tpp.

4.35.3.6 template<typename type > void Icarus::vtkWriter::addCellVecToAll (const type *datax*[ ], const type *datay*[ ], const type *dataz*[ ], size_t *length*, std::string *name*) [inline]

Fuegt vektorwertige Zelldaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax[ ] Array der x Komponenten

datay[ ] Array der y Komponenten

dataz[ ] Array der z Komponenten

length Länge der Arrays

name Name der Daten

Definiert in Zeile 419 der Datei vtkwriter.tpp.

4.35.3.7 template<typename type > void Icarus::vtkWriter::addCellVecToTimestep (const FullVector< type > & *datax*, const FullVector< type > & *datay*, const FullVector< type > & *dataz*, const size_t *timestep*, const std::string *name*) [inline]

Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax Vektor mit den x Komponenten der zu schreibenden Daten

datay Vektor mit den y Komponenten der zu schreibenden Daten

dataz Vektor mit den z Komponenten der zu schreibenden Daten

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 245 der Datei vtkwriter.tpp.

4.35.3.8 template<typename type > void Icarus::vtkWriter::addCellVecToTimestep (const type *datax*[ ], const type *datay*[ ], const type *dataz*[ ], const size_t *length*, const size_t *timestep*, std::string *name*) [inline]

Fuegt vektorwertige Zelldaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax[ ] Array der x Komponenten

datay[ ] Array der y Komponenten

dataz[ ] Array der z Komponenten

length Länge der Arrays

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 288 der Datei vtkwriter.tpp.

4.35.3.9 template<typename type > void Icarus::vtkWriter::addPointDataToAll (const FullVector< type > & *data*, const std::string *name*) [inline]

Fuegt skalarwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data Vektor mit den zu schreibenden Daten

name Name der Daten

Definiert in Zeile 343 der Datei vtkwriter.tpp.

4.35.3.10 template<typename type > void Icarus::vtkWriter::addPointDataToAll (const type *data*[ ], size_t *length*, std::string *name*) [inline]

Fuegt skalarwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data[ ] Array der Quelldaten

length Länge des Arrays

name Name der Daten

Definiert in Zeile 332 der Datei vtkwriter.tpp.

4.35.3.11 template<typename type > void Icarus::vtkWriter::addPointDataToTimestep (const FullVector< type > & *data*, const size_t *timestep*, const std::string *name*) [inline]

Fuegt skalarwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data Vektor mit den zu schreibenden Daten

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 44 der Datei vtkwriter.tpp.

4.35.3.12 template<typename type > void Icarus::vtkWriter::addPointDataToTimestep (const type *data*[ ], const size_t *length*, const size_t *timestep*, std::string *name*) [inline]

Fuegt skalarwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

data[ ] Array der Quelldaten

length Länge des Arrays

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 5 der Datei vtkwriter.tpp.

4.35.3.13 template<typename type > void Icarus::vtkWriter::addPointVecToAll (const FullVector< type > & *datax*, const FullVector< type > & *datay*, const FullVector< type > & *dataz*, const std::string *name*) [inline]

Fuegt vektorwertige Punktdaten zu allen Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax Vektor mit den x Komponenten der zu schreibenden Daten

datay Vektor mit den y Komponenten der zu schreibenden Daten

dataz Vektor mit den z Komponenten der zu schreibenden Daten

name Name der Daten

Definiert in Zeile 377 der Datei vtkwriter.tpp.

4.35.3.14 template<typename type > void Icarus::vtkWriter::addPointVecToAll (const type *datax*[ ], const type *datay*[ ], const type *dataz*[ ], size_t *length*, std::string *name*) [inline]

Fuegt vektorwertige Punktdaten zu allen Zeitschritten hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax[ ] Array der x Komponenten

datay[ ] Array der y Komponenten

dataz[ ] Array der z Komponenten

length Länge der Arrays

name Name der Daten

Definiert in Zeile 391 der Datei vtkwriter.tpp.

4.35.3.15 template<typename type > void Icarus::vtkWriter::addPointVecToTimestep (const FullVector< type > & *datax*, const FullVector< type > & *datay*, const FullVector< type > & *dataz*, const size_t *timestep*, const std::string *name*) [inline]

Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax Vektor mit den x Komponenten der zu schreibenden Daten

datay Vektor mit den y Komponenten der zu schreibenden Daten

dataz Vektor mit den z Komponenten der zu schreibenden Daten

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 83 der Datei vtkwriter.tpp.

4.35.3.16 template<typename type > void Icarus::vtkWriter::addPointVecToTimestep (const type *datax*[ ], const type *datay*[ ], const type *dataz*[ ], const size_t *length*, const size_t *timestep*, std::string *name*) [inline]

Fuegt vektorwertige Punktdaten zu einem Zeitschritt hinzu. Die Daten werden in der Datei als float abgespeichert.

Template Parameters:

type Datentyp der Quelldaten

Parameter:

datax[ ] Array der x Komponenten

datay[ ] Array der y Komponenten

dataz[ ] Array der z Komponenten

length Länge der Arrays

timestep Zeitschritt, dem die Daten hinzugefuegt werden

name Name der Daten

Definiert in Zeile 122 der Datei vtkwriter.tpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/vtkwriter.hpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/vtkwriter.tpp
- /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/vtkwriter.cpp

Kapitel 5

Datei-Dokumentation

5.1 /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/discretizer.hp Dateireferenz

```
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
#include <cassert>
```

Datenstrukturen

- struct Icarus::Vertex

  Simple Struktur um einen Raumpunkt kompakt zu speichern. Kann ebenso dazu benutzt werden einen Vektor zu speichern.

- class Icarus::Face

  Eine Flaeche im Raum, aufgespannt von beliebig vielen Punkten. Der Benutzer muss selbst darauf achten, dass die Eckpunkte alle in einer Ebene liegen.

- class Icarus::Object

  Ein Objekt im Raum. Wird durch Flaechen aufgespannt. Ermoeglicht die Ueberpruefung, ob ein Raumpunkt im Inneren, auf dem Rand oder ausserhalb des Objektes liegt.

Funktionen

- std::vector< char > Icarus::discretizer (std::string inputFile, float h, int nx, int ny, int nz)

  "Diskretisiert" Raum anhand einer obj-Datei. Die Diskretisierung beginnt im Ursprung und geht in positve Koordinatenrichtungen.

- void Icarus::save_discretizer (std::vector< char > discretized_points, std::string outputFile, int nx, int ny, int nz)

  "Diskretiesiert" mit 'discretizer' und schreibt den Rueckgabevektor in eine Datei.

5.1.1 Ausführliche Beschreibung

Definiert in Datei discretizer.hpp.

5.2 /home/warehouse15/choeppke/Documents/Kurse/Semester-05/Studienprojekt-TM/68616564616C7573/Projekt/src/include/mpihandler.h Dateireferenz

MpiHandler. `#include <string>`

`#include "mpi.h"`

`#include "proto.hpp"`

Datenstrukturen

- class Icarus::MpiHandler

Makrodefinitionen

- #define **USE_MPI_ERROR_CHECKING**
- #define **MPI_HANDLER** __mpi_inst
- #define **MPI_SCALL**(X) __mpi_inst.MpiSafeCall(__LINE__,__FILE__,X)

Variablen

- Icarus::MpiHandler **__mpi_inst**

5.2.1 Ausführliche Beschreibung

MpiHandler.

Autor:

David

Definiert in Datei mpihandler.hpp.