Electromagnet Calibration

Generated by Doxygen 1.8.6

Wed Jun 22 2016 22:23:06

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 YAML::convert< Eigen::Matrix3d > Struct Template Reference

Static Public Member Functions

- static Node **encode** (const Eigen::Matrix3d &rhs)
- static bool **decode** (const Node &node, Eigen::Matrix3d &rhs)

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

- EigenToYAML.h

2.2 YAML::convert< Eigen::MatrixXd > Struct Template Reference

Static Public Member Functions

- static Node **encode** (const Eigen::MatrixXd &rhs)
- static bool **decode** (const Node &node, Eigen::MatrixXd &rhs)

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

- EigenToYAML.h

2.3 YAML::convert< Eigen::Vector3d > Struct Template Reference

Static Public Member Functions

- static Node **encode** (const Eigen::Vector3d &rhs)
- static bool **decode** (const Node &node, Eigen::Vector3d &rhs)

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

- EigenToYAML.h

2.4 YAML::convert< Eigen::VectorXd > Struct Template Reference

Static Public Member Functions

- static Node **encode** (const Eigen::VectorXd &rhs)
- static bool **decode** (const Node &node, Eigen::VectorXd &rhs)

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

- EigenToYAML.h

2.5 YAML::convert< std::vector< double > > Struct Template Reference

Static Public Member Functions

- static Node **encode** (const std::vector< double > &rhs)
- static bool **decode** (const Node &node, std::vector< double > &rhs)

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

- EigenToYAML.h

2.6 ElectromagnetCalibration Class Reference

a class describing the calibration of an arbitrary magnetic system

`#include <electromagnet_calibration.h>`

Classes

- struct MagneticWorkSpace

Public Types

- enum calibration_constraints { UNIT_HEADING_ONLY, HEADING_AND_POSITION, HEADING_THEN_P-OSITION }

  The calibration_constraints enum defines what constraints are active during calibration and how they are applied.

Public Member Functions

- ElectromagnetCalibration (std::string calibrationFileName)

  constructor that builds a calibrated electromagnet system based on a prexisting yaml encoded calibration file.
- ElectromagnetCalibration (std::string systemName, const MagneticWorkSpace &workSpace_, const std::vector< ScalorPotential > &coilList, const ScalorPotential &dc_field_offset=ScalorPotential())

  constructor that builds a calibrated electromanget system based on a list of scalor potentials and an optional offset.
- Eigen::Vector3d fieldAtPoint (const Eigen::VectorXd &currentVector, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the field at a point
- Eigen::Matrix3d gradientAtPoint (const Eigen::VectorXd &currentVector, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

returns the 3x3 symetric gradient matrix at a desired location

- Vector8d fieldAndGradientAtPoint (const Eigen::VectorXd &currentVector, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 8x1 stacked field over gradient vector
- Vector8d offsetFieldAndGradientAtPoint (const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 8x1 stacked field over gradient vector due to the zero-current field offset
- Eigen::MatrixXd fieldCurrentJacobian (const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 3xN matrix mapping field at a point to the current in each of the N sources
- Eigen::MatrixXd gradientCurrentJacobian (const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 5xN matrix mapping the field radient at a point to the current in each of the N sources
- Eigen::MatrixXd fieldAndGradientCurrentJacobian (const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 8xN matrix mapping of the stacked field over field gradient at a point to the current in each of the N sources
- Eigen::Matrix< double, 5, 3 > gradientPositionJacobian (const Eigen::VectorXd &currentVector, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  returns the 5x3 field gradient jacobian at a point given the currents in each of the N sources
- void fullMagneticState (Eigen::Vector3d &fieldAtPoint, Eigen::Matrix< double, 8, 3 > &fieldGradientPositionJacobian, Eigen::MatrixXd &fieldGradientCurrentJacobian, const Eigen::VectorXd &current, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  calculates the field, field and field gradient jacobians, and the current jacobians at a point given the currents in each of the N sources this faunction is more efficient than calling each of the applicable functions seporately.
- MagneticState fullMagneticState (const Eigen::VectorXd &currentVector, const Eigen::Vector3d &position=Eigen::Vector3d::Zero()) const

  calculates the field, field and field gradient jacobians, and the current jacobians at a point given the currents in each of the N sources this faunction is more efficient than calling each of the applicable functions seporately.
- bool loadCalibration (std::string fileName)

  loads a new calibration file
- bool writeCalibration (std::string fileName) const

  writes a new calibration file
- int getNumberOfCoils () const

  returns the number of coils in the calibration
- int getNumberOfSources (unsigned int coilNum) const

  returns the number of sources for the given coil
- int getNumberOfCoeffients (unsigned int coilNum, unsigned int srcNum) const

  returns the number of coefficients for the given source
- bool hasOffset () const

  returns if the calibration has a DC offset
- std::string getName () const

  returns the calibration name
- bool pointInWorkspace (const Eigen::Vector3d &position) const

  checks to see if a point lies within the calibrated workspace
- MagneticWorkSpace getWorkSpace () const

  getWorkSpace returns the rectanglar extent of the calibrated workspace
- void setWorkSpace (const MagneticWorkSpace &ws)

  sets the magnetic workspace to the desired specification
- void useOffset (bool offsetOn)

  enables or disables the use of the offset, if one exists.
- void useOffset (const ScalorPotential &newOffset)

  adds an offset to the system and enables it.
- bool useOffset () const

  returns if the offset is enabled or disabled.

- void calibrate (std::string calibrationName, const std::vector< MagneticMeasurement > &dataList, bool printProgress=true, bool printStats=true, calibration_constraints constraint=HEADING_THEN_POSITION, double minimumSourceToCenterDistance=-1, double maximumSourceToCenterDistance=-1, double converganceTolerance=1e-12, int maxIterations=10000, int numberOfConvergedIterations=1)

    calibrate Performs a system calibration based on gathered data and a current guess as to the scalor potential structure.
- void printStats (const std::vector< MagneticMeasurement > &dataList) const

    Prints to the terminal (cout) statistics describing how well the system reproduces the magnetic measurements provided.

Static Public Member Functions

- static Eigen::Matrix3d remapGradientVector (const Vector5d &gradVector)

    converts a 5x1 gradient vector into a symetric zero trace 3x3 matrix
- static Vector5d remapGradientMatrix (const Eigen::Matrix3d &gradMatrix)

    converts a 3x3 gradient marix into the 5x1 gradient vector
- static Eigen::MatrixXd packForceMatrix (const Eigen::Vector3d &moment)

    converts a 3x1 moment vector into a 3x5 force matrix

Protected Member Functions

- ElectromagnetCalibration ()
- bool **checkSourcePositions** (bool printWarning=false) const

Protected Attributes

- MagneticWorkSpace **workSpace**
- std::vector< ScalorPotential > **coilList**
- ScalorPotential **offset**
- bool **use_offset**
- std::string **name**

2.6.1 Detailed Description

a class describing the calibration of an arbitrary magnetic system

This class assumes the system can be described by spherical harmonic scalor potentials at multiple locations with weightings proportional to the currents appied.

2.6.2 Member Enumeration Documentation

2.6.2.1 enum ElectromagnetCalibration::calibration_constraints

The calibration_constraints enum defines what constraints are active during calibration and how they are applied.

Enumerator

UNIT_HEADING_ONLY Constrains the azimuth of the potentials to be unit length

HEADING_AND_POSITION Constrains the azimuth of potentials to be unit length and enforces that the positions lie in a spherical anulous outside of the measured data and prevents them from going to infinity.

HEADING_THEN_POSITION First solves with the heading constraints, then it resolves pushing any sources that lie inside the measurement data region out of a bounding circle of the data.

2.6.3 Constructor & Destructor Documentation

2.6.3.1 ElectromagnetCalibration::ElectromagnetCalibration (std::string *calibrationFileName*)

constructor that builds a calibrated electromagnet system based on a prexisting yaml encoded calibration file.

Parameters

calibrationFile-Name	is the file location for the calibration

2.6.3.2 ElectromagnetCalibration::ElectromagnetCalibration (std::string *systemName*, const MagneticWorkSpace & *workSpace_*, const std::vector< ScalorPotential > & *coilList*, const ScalorPotential & *dc_field_offset* = ScalorPotential())

constructor that builds a calibrated electromanget system based on a list of scalor potentials and an optional offset.

Parameters

*workspace_*	The workspace for which this calibration applies.
coilList	The list of coils and their respective sources.
dc_field_offset	The dc offset field, if any.

2.6.3.3 ElectromagnetCalibration::ElectromagnetCalibration ( ) [protected]

Default Constructor only available to inheriting classes

2.6.4 Member Function Documentation

2.6.4.1 void ElectromagnetCalibration::calibrate (std::string *calibrationName*, const std::vector< MagneticMeasurement > & *dataList*, bool *printProgress* = true, bool *printStats* = true, calibration_constraints *constraint* = HEADING_THEN_POSITION, double *minimumSourceToCenterDistance* = -1, double *maximumSourceToCenterDistance* = -1, double *converganceTolerance* = 1e-12, int *maxIterations* = 10000, int *numberOfConvergedIterations* = 1)

calibrate Performs a system calibration based on gathered data and a current guess as to the scalor potential structure.

Parameters

calibrationName	The name of the calibration.
dataList	The list of magnetic measurements.
printProgress	Boolean to idenify if the convergance progress should be printed with cout.
printStats	Boolean to identify if the convergence statistics shoudl be printed with cout once completed.
constraint	Identifies what kind of constraits should be applied to the positions and headings during convergance.
minimumSource-ToCenter-Distance	Identifies the minimum acceptable distance between the center of the workspace and any scalor potential source. Default is just outside workspace volume.
maximum-SourceToCenter-Distance	Identifies the maximum acceptable distance between the center of the workspace and any scalor potential source. Default is 10 times the initial guess distances.

converganceTolerance	Specifies the convergance tolerance. Default is 1e-12.
maxIterations	Specifies the maximum number of interations to used to prvent the loop from infinite cycles.
numberOfConvergedIterations	Specifies the number of sequential converged passes to prevent a false positive in convergance.

=0

2.6.4.2 Vector8d ElectromagnetCalibration::fieldAndGradientAtPoint (const Eigen::VectorXd & *currentVector,* const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the 8x1 stacked field over gradient vector

Parameters

currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

The gradient matrix has been repacked, since it is symetric an has zero trace, into a five element vector. The element order is: [dBx/dx, dBx/dy, dBx/dz, dBy/dy, dBy/dz]^T This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.3 Eigen::MatrixXd ElectromagnetCalibration::fieldAndGradientCurrentJacobian (const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the 8xN matrix mapping of the stacked field over field gradient at a point to the current in each of the N sources

Parameters

position	is the position in the workspace the field is desired

The gradient matrix has been repacked, since it is symetric an has zero trace, into a five element vector. The element order is: [dBx/dx, dBx/dy, dBx/dz, dBy/dy, dBy/dz]^T This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.4 Eigen::Vector3d ElectromagnetCalibration::fieldAtPoint (const Eigen::VectorXd & *currentVector,* const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the field at a point

Parameters

currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.5 Eigen::MatrixXd ElectromagnetCalibration::fieldCurrentJacobian (const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the 3xN matrix mapping field at a point to the current in each of the N sources

Parameters

position	is the position in the workspace the field is desired

This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.6 void ElectromagnetCalibration::fullMagneticState (Eigen::Vector3d & *fieldAtPoint,* Eigen::Matrix< double, 8, 3 > & *fieldGradientPositionJacobian,* Eigen::MatrixXd & *fieldGradientCurrentJacobian,* const Eigen::VectorXd & *current,* const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

calculates the field, field and field gradient jacobians, and the current jacobians at a point given the currents in each of the N sources this faunction is more efficient than calling each of the applicable functions seporately.

Parameters

fieldAtPoint	a pass by reference Vector to return the calculated field
fieldGradient-PositionJacobian	a pass by reference matrix to return the field jacobian stacked over the field gradient jacobian
fieldGradient-CurrentJacobian	a pass by reference matrix to return the field and gradient current jacobian
currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

The 3x3 gradient and 3x3x3 field gradient jacobian tensor has been repacked, since the 3x3 field gradient is symetric and has zero trace, into a 8x3 element vector. The element order is:

$$\begin{bmatrix} \frac{\partial B_x}{\partial x} & \frac{\partial B_x}{\partial y} & \frac{\partial B_x}{\partial z} \\ \frac{\partial B_x}{\partial y} & \frac{\partial B_y}{\partial y} & \frac{\partial B_y}{\partial z} \\ \frac{\partial B_x}{\partial z} & \frac{\partial B_y}{\partial z} & -\left(\frac{\partial B_x}{\partial x}+\frac{\partial B_y}{\partial y}\right) \\ \frac{\partial^2 B_x}{\partial x \partial x} & \frac{\partial^2 B_x}{\partial x \partial y} & \frac{\partial^2 B_x}{\partial x \partial z} \\ \frac{\partial^2 B_x}{\partial y \partial x} & \frac{\partial^2 B_x}{\partial y \partial y} & \frac{\partial^2 B_x}{\partial y \partial z} \\ \frac{\partial^2 B_x}{\partial z \partial x} & \frac{\partial^2 B_x}{\partial z \partial y} & \frac{\partial^2 B_x}{\partial z \partial z} \\ \frac{\partial^2 B_y}{\partial y \partial x} & \frac{\partial^2 B_y}{\partial y \partial y} & \frac{\partial^2 B_y}{\partial y \partial z} \\ \frac{\partial^2 B_y}{\partial z \partial x} & \frac{\partial^2 B_y}{\partial z \partial y} & \frac{\partial^2 B_y}{\partial z \partial z} \end{bmatrix}$$

This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.7 MagneticState ElectromagnetCalibration::fullMagneticState (const Eigen::VectorXd & *currentVector,* const Eigen::Vector3d & *position* = `Eigen::Vector3d::Zero()`) const

calculates the field, field and field gradient jacobians, and the current jacobians at a point given the currents in each of the N sources this faunction is more efficient than calling each of the applicable functions seporately.

Parameters

currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

Returns

the information on the magnetic field at a point in the form of a MagneticState object.

2.6.4.8 Eigen::Matrix3d ElectromagnetCalibration::gradientAtPoint (const Eigen::VectorXd & *currentVector,* const Eigen::Vector3d & *position* = `Eigen::Vector3d::Zero()`) const

returns the 3x3 symetric gradient matrix at a desired location

Parameters

currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.9 Eigen::MatrixXd ElectromagnetCalibration::gradientCurrentJacobian (const Eigen::Vector3d & *position* = `Eigen::Vector3d::Zero()`) const

returns the 5xN matrix mapping the field radient at a point to the current in each of the N sources

Parameters

position	is the position in the workspace the field is desired

The gradient matrix has been repacked, since it is symetric an has zero trace, into a five element vector. The element order is: [dBx/dx, dBx/dy, dBx/dz, dBy/dy, dBy/dz]^T This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.10 Eigen::Matrix< double, 5, 3 > ElectromagnetCalibration::gradientPositionJacobian (const Eigen::VectorXd & *currentVector,* const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the 5x3 field gradient jacobian at a point given the currents in each of the N sources

Parameters

currentVector	is an orderd list of currents in each coil
position	is the position in the workspace the field is desired

The 3x3x3 tensor has been repacked, since the 3x3 field gradient is symetric an has zero trace, into a 5x3 element vector. The element order is: [dBx/dxdx, dBx/dxdy, dBx/dxdz dBx/dydx, dBx/dydy, dBx/dydz dBx/dzdx, dBx/dzdy, dBx/dzdz dBy/dydx, dBy/dydy, dBy/dydz dBy/dzdx, dBy/dzdy, dBy/dzdz] This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.11 bool ElectromagnetCalibration::loadCalibration (std::string *fileName*)

loads a new calibration file

Parameters

a	string pointing to the location of the yaml formated calbiration file

2.6.4.12 Vector8d ElectromagnetCalibration::offsetFieldAndGradientAtPoint (const Eigen::Vector3d & *position =* `Eigen::Vector3d::Zero()`) const

returns the 8x1 stacked field over gradient vector due to the zero-current field offset

Parameters

position	is the position in the workspace the field is desired

The gradient matrix has been repacked, since it is symetric an has zero trace, into a five element vector. The element order is: [dBx/dx, dBx/dy, dBx/dz, dBy/dy, dBy/dz]^T This function does not check to see if the point is actually in the calibrated workspace.

2.6.4.13 Eigen::MatrixXd ElectromagnetCalibration::packForceMatrix (const Eigen::Vector3d & *moment*) `[static]`

converts a 3x1 moment vector into a 3x5 force matrix

Parameters

moment	is the magnetic moment that is packed into the force matrix

converts a 3x1 moment vector into a 3x5 force matrix to allow easy calculation of the magnetic force by multiplication with the gradient vector

2.6.4.14 bool ElectromagnetCalibration::pointInWorkspace (const Eigen::Vector3d & *position*) const

checks to see if a point lies within the calibrated workspace

Parameters

position	the desired position to check

2.6.4.15 void ElectromagnetCalibration::printStats (const std::vector< MagneticMeasurement > & *dataList*) const

Prints to the terminal (cout) statistics describing how well the system reproduces the magnetic measurements provided.

Parameters

dataList	A set of magnetic measurements to compair the calibration to.

The output of this function provides the R^2 value, the mean error, and the square root of covariance (standard deviation) of the error in both a millitesla and normalized percentage basis.

2.6.4.16 Vector5d ElectromagnetCalibration::remapGradientMatrix (const Eigen::Matrix3d & *gradMatrix*) [static]

converts a 3x3 gradient marix into the 5x1 gradient vector

Parameters

gradMatrix	is the desired matrix to be remaped

This function does not check to verify the matrix is indeed symetric and zero trace

2.6.4.17 Eigen::Matrix3d ElectromagnetCalibration::remapGradientVector (const Vector5d & *gradVector*) [static]

converts a 5x1 gradient vector into a symetric zero trace 3x3 matrix

Parameters

gradVector	is the desired vector to be remaped

2.6.4.18 bool ElectromagnetCalibration::writeCalibration (std::string *fileName*) const

writes a new calibration file

Parameters

a	string pointing to the location for the yaml formated calbiration file

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

- electromagnet_calibration.h
- electromagnet_calibration.cpp

2.7 MagneticMeasurement Struct Reference

The MagneticMeasurementData struct provides the format calibration data must be supplied for the calibration.

`#include <electromagnet_calibration.h>`

Public Member Functions

- MagneticMeasurement ()
- MagneticMeasurement (const Eigen::Vector3d &Field, const Eigen::Vector3d &Pos, const Eigen::VectorXd &CurrentVec)

calibrationDataPoint

Public Attributes

- Eigen::Vector3d Field
- Eigen::Vector3d Position
- Eigen::VectorXd AppliedCurrentVector

2.7.1 Detailed Description

The MagneticMeasurementData struct provides the format calibration data must be supplied for the calibration.

2.7.2 Constructor & Destructor Documentation

2.7.2.1 MagneticMeasurement::MagneticMeasurement ( )

Initializes all parameters to zero.

2.7.2.2 MagneticMeasurement::MagneticMeasurement (const Eigen::Vector3d & *Field*, const Eigen::Vector3d & *Pos*, const Eigen::VectorXd & *CurrentVec*)

calibrationDataPoint

Parameters

Field	the field value in Tesla
Pos	the position of the measurement in meters
CurrentVec	the applied current vector in Amps

2.7.3 Member Data Documentation

2.7.3.1 Eigen::VectorXd MagneticMeasurement::AppliedCurrentVector

The applied current vector in Amps. A Nx1 Vector

2.7.3.2 Eigen::Vector3d MagneticMeasurement::Field

The measured Field in Tesla. A 3x1 Vector

2.7.3.3 Eigen::Vector3d MagneticMeasurement::Position

The position of the measurement in meters. A 3x1 Vector

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

- electromagnet_calibration.h
- electromagnet_calibration.cpp

2.8 MagneticState Struct Reference

The MagneticState struct contains informtion necessary to quantify the field at a position for control.

```
#include <electromagnet_calibration.h>
```

Public Attributes

- Eigen::Vector3d Field

  Field The magnetic field at the position in Tesla. This is a 3x1 Matrix.
- Eigen::Matrix3d Gradient

  Gradient The Magnetic gradient at the position in Tesla/Meter. This is a 3x3 Matrix.
- Eigen::Matrix< double, 5, 3 > GradientPositionJacobian

  GradientPositionJacobian The rate of change of the 5 unique gradient terms with respect to position in Tesla/Meter^2. This is a 5x3 Matrix, to get the 3x3x3 Gradeint derivative tensor, the function ElectromagnetCalibration::remapGradientVector can be used on each column to get the gradient matrix change in x,y and z in that order.
- Eigen::MatrixXd FieldGradientActuationMatrix

  FieldGradientActuationMatrix The field and gradient (packed into a 5x1 vector) current jacobian. This is a 8xN matrix, where N is the number of current sources. The first 3 rows are for field the last five rows are for gradient ElectromagnetCalibration::remapGradientVector.

2.8.1 Detailed Description

The MagneticState struct contains informtion necessary to quantify the field at a position for control.

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

- electromagnet_calibration.h

2.9 ElectromagnetCalibration::MagneticWorkSpace Struct Reference

Public Member Functions

- **MagneticWorkSpace** (double size)
- **MagneticWorkSpace** (double xMin, double xMax, double yMin, double yMax, double zMin, double zMax)

Public Attributes

- double xMin
- double xMax
- double yMin
- double yMax
- double zMin
- double zMax

2.9.1 Member Data Documentation

2.9.1.1 double ElectromagnetCalibration::MagneticWorkSpace::xMax

Maximum X position in Meters

2.9.1.2 double ElectromagnetCalibration::MagneticWorkSpace::xMin

Minimum X Position in Meters

2.9.1.3 double ElectromagnetCalibration::MagneticWorkSpace::yMax

Maximum Y Position in Meters

2.9.1.4 double ElectromagnetCalibration::MagneticWorkSpace::yMin

Minimum Y Position in Meters

2.9.1.5 double ElectromagnetCalibration::MagneticWorkSpace::zMax

Maximum Z Position in Meters

2.9.1.6 double ElectromagnetCalibration::MagneticWorkSpace::zMin

Minimum Z Position in Meters

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

- electromagnet_calibration.h
- electromagnet_calibration.cpp

2.10 ScalorPotential Class Reference

a class describing the field of a collection of scalor potentials

```
#include <scalorPotential.h>
```

Classes

- struct srcCoeff
- class srcStruct

  The srcStruct struct describes the scalor potential for an individual source.

Public Member Functions

- ScalorPotential ()

  ScalorPotential initializes an empty scalor potential, one with no sources.
- ScalorPotential (const std::vector< srcStruct > &srcList)

  the constructor from a coil list.
- ScalorPotentialState getState (const Eigen::Vector3d &position, int sourceNumber=-1) const

  getState returns the state at the position requested
- double getValue (const Eigen::Vector3d &position, int sourceNumber=-1) const

  getValue Returns the scalor potential magnitude at the position requested
- Eigen::Vector3d getGradient (const Eigen::Vector3d &position, int sourceNumber=-1) const

  getGradient Returns the spacial gradient of the potential at the position requested
- unsigned int getNumberOfSources () const

  returns the number of sources for the given coil
- srcStruct getSourceStruct (unsigned int sourceNumber) const

  getSourceStruct returns a copy ofthe source structure
- void setSourceStruct (unsigned int sourceNumber, const srcStruct &newSrc)

  setSourceStruct Sets a new source to replace an existing source. Good to use with
- void removeSourceStruct (unsigned int sourceNumber)

  removeSourceStruct Removes a source from the list of sources.

Static Public Member Functions

- static Eigen::Matrix3d remapSecondDerivativeVec (const Vector5d &gradVector)

    converts a 5x1 gradient vector into a symetric zero trace 3x3 matrix
- static Vector5d remapSecondDerivativeMat (const Eigen::Matrix3d &gradMatrix)

    converts a 3x3 gradient marix into the 5x1 gradient vector

Protected Member Functions

- ScalorPotentialCalibrationJacobians **srcCalibrationInformation** (const Eigen::Vector3d &position, unsigned int srcNum) const
- template<typename Derived >
  void packCalibrationState (Eigen::MatrixBase< Derived > const &stateVector_) const
- template<typename Derived >
  void **unpackCalibrationState** (Eigen::MatrixBase< Derived > const &stateVector_)
- template<typename Derived , typename Derived2 >
  void **packJacobians** (int srcNum, double current, const Eigen::Vector3d &pos, const Eigen::Vector3d &fieldError, Eigen::MatrixBase< Derived > const &firstTransposed_, Eigen::MatrixBase< Derived2 > const &hessian_)
- int **getNumCalibrationParameters** (int srcNum=-1) const

Static Protected Member Functions

- static double LegandrePolynomial (double x, int order, int der=0)

    Calculates the Legandre Polynomial of a given order deritivative X <= (-1,1)
- static void **srcFieldGradient** (const Eigen::Vector3d &position, const srcStruct &src, ScalorPotentialState &currentState)

Protected Attributes

- std::vector< srcStruct > **srcList**
- int **numCalParameters**

Friends

- class **ElectromagnetCalibration**

2.10.1 Detailed Description

a class describing the field of a collection of scalor potentials

This class assumes the system can be described by spherical harmonic scalor potentials at multiple locations. See. "PUBLICATION HERE"

2.10.2 Constructor & Destructor Documentation

2.10.2.1 ScalorPotential::ScalorPotential (const std::vector< srcStruct > & *srcList_*)

the constructor from a coil list.

Parameters

coilList	The list of coils and their respective sources.
dc_field_offset	The dc offset field, if any.

2.10.3 Member Function Documentation

2.10.3.1 Eigen::Vector3d ScalorPotential::getGradient (const Eigen::Vector3d & *position*, int *sourceNumber* = `-1`) const

getGradient Returns the spacial gradient of the potential at the position requested

Parameters

position	3D position of interest
sourceNumber	The contribution limited to a specific source. Default is -1, which is interperated as the total effect of all sources

Returns

The value of the scalor potential spatial gradeint

2.10.3.2 ScalorPotential::srcStruct ScalorPotential::getSourceStruct (unsigned int *sourceNumber*) const

getSourceStruct returns a copy ofthe source structure

Parameters

sourceNumber	The ordered number of the source to be copied.

Returns

See Also

srcStruct. If the number is out of bounds it will return an empty source.

2.10.3.3 ScalorPotentialState ScalorPotential::getState (const Eigen::Vector3d & *position*, int *sourceNumber* = `-1`) const

getState returns the state at the position requested

Parameters

position	3D position of interest
sourceNumber	The contribion limited to a specific source. Default is -1, which is interperated as the total effect all sources

Returns

The Scalor Potential State

See Also

ScalorPotentialState

2.10.3.4 double ScalorPotential::getValue (const Eigen::Vector3d & *position*, int *sourceNumber* = `-1`) const

getValue Returns the scalor potential magnitude at the position requested

Parameters

position	3D position of interest
sourceNumber	The contribution limited to a specific source. Default is -1, which is interperated as the total effect of all sources

Returns

The value of the scalor potential state

2.10.3.5 template< typename Derived > void ScalorPotential::packCalibrationState (Eigen::MatrixBase< Derived > const & *stateVector_*) const [inline], [protected]

< Returns the vector packing of the node state

2.10.3.6 Vector5d ScalorPotential::remapSecondDerivativeMat (const Eigen::Matrix3d & *gradMatrix*) [static]

converts a 3x3 gradient marix into the 5x1 gradient vector

Parameters

gradMatrix	is the desired matrix to be remaped

This function does not check to verify the matrix is indeed symetric and zero trace

2.10.3.7 Eigen::Matrix3d ScalorPotential::remapSecondDerivativeVec (const Vector5d & *gradVector*) [static]

converts a 5x1 gradient vector into a symetric zero trace 3x3 matrix

Parameters

gradVector	is the desired vector to be remaped

2.10.3.8 void ScalorPotential::removeSourceStruct (unsigned int *sourceNumber*)

removeSourceStruct Removes a source from the list of sources.

Parameters

sourceNumber	The ordered number of the source to replace.

2.10.3.9 void ScalorPotential::setSourceStruct (unsigned int *sourceNumber*, const srcStruct & *newSrc*)

setSourceStruct Sets a new source to replace an existing source. Good to use with

See Also

getSourceStruct for modifying a source configuration.

Parameters

sourceNumber	The ordered number of the source to replace.

newSrc	The new source definition.

This function will add empty sources until sourceNumber is reached if sourceNumber is greator than the current number of sources.

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

- scalorPotential.h
- scalorPotential.cpp

2.11 ScalorPotentialCalibrationJacobians Struct Reference

Public Member Functions

- ScalorPotentialCalibrationJacobians ()

Public Attributes

- Eigen::Vector3d firstSpatialDerivative

  Field, the gradient of the potential.
- Eigen::MatrixXd firstSpatialDerivative_A_CoeffDerivative

  How the field changes with the A coefficients.
- Eigen::MatrixXd firstSpatialDerivative_B_CoeffDerivative

  How the field changes with the B coefficients.
- Eigen::MatrixXd firstSpatialDerivative_dA_dHeading

  The second deritive of the field with respect to the A coefficients and with Heading.
- Eigen::MatrixXd firstSpatialDerivative_dB_dHeading

  The second deritive of the field with respect to the B coefficients and with Heading.
- Eigen::MatrixXd firstSpatialDerivative_dA_dPosition

  The second deritive of the field with respect to the A coefficients and with Position.
- Eigen::MatrixXd firstSpatialDerivative_dB_dPosition

  The second deritive of the field with respect to the B coefficients and with Position.
- Eigen::Matrix3d firstSpatialDerivative_SourcePositionDerivative

  Field spatial gradient.
- Eigen::Matrix3d firstSpatialDerivative_SourceHeadingDerivative

  How the field changes with the source heading.
- Eigen::Matrix< double, 5, 3 > secondSpatialDerivative_SourcePositionDerivative

  5x3 matrix describing how the field spatial gradient changes with the source position
- Eigen::Matrix< double, 5, 3 > secondSpatialDerivative_SourceHeadingDerivative

  5x3 matrix describing how the field spatial gradient changes with the source heading
- Eigen::Matrix< double, 9, 3 > firstSpatialDerivative_secondSourceHeadingDerivative

  A list of 9x3 matricies describing how the field changes with the source heading [d(B∗X)/dz; d(B∗Y)/dz; d(B∗Z)/dz].

2.11.1 Constructor & Destructor Documentation

2.11.1.1 ScalorPotentialCalibrationJacobians::ScalorPotentialCalibrationJacobians ( )

< Field, the gradient of the potential.

< How the field changes with the A coefficients

< How the field changes with the B coefficients

< Field spatial gradient

< How the field changes with the source heading

< the second deritive of field with heading (d(BX)dz; d(BY/dz; d(BZ)/dz)

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

- scalorPotential.h
- scalorPotential.cpp

2.12 ScalorPotentialState Struct Reference

Public Attributes

- double value

  The value of the scalor potential.
- Eigen::Vector3d firstSpatialDerivative

  Field, the gradient of the potential.
- Eigen::Matrix< double, 3, 3 > secondSpatialDerivative

  Field spatial gradient.
- Eigen::Matrix< double, 5, 3 > thirdSpatialDerivative

  How the field spatial gradient changes with position, assumes potentials of order > 1. (no sources or sinks)
- std::vector< Eigen::MatrixXd > firstSpatialDerivative_SourceHeadingDerivative

  A list of 3x3 matricies describing how the field changes with the source heading.
- std::vector< Eigen::MatrixXd > firstSpatialDerivative_SourcePositionDerivative

  A list of 3x3 matricies describing how the field changes with the source position.
- std::vector< Eigen::MatrixXd > secondSpatialDerivative_SourcePositionDerivative

  A list of 5x3 matricies describing how the field spatial gradient changes with the source position.
- std::vector< Eigen::MatrixXd > secondSpatialDerivative_SourceHeadingDerivative

  A list of 5x3 matricies describing how the field spatial gradient changes with the source heading.

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

- scalorPotential.h
- scalorPotential.cpp

2.13 ScalorPotential::srcCoeff Struct Reference

Public Member Functions

- **srcCoeff** (double value, unsigned int order)

Public Attributes

- unsigned int **order**
- double **coeff**

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

- scalorPotential.h
- scalorPotential.cpp

2.14 ScalorPotential::srcStruct Class Reference

The srcStruct struct describes the scalor potential for an individual source.

```
#include <scalorPotential.h>
```

Public Member Functions

- unsigned int **getMaxOrder_A_Coeff** () const
- unsigned int **getMaxOrder_B_Coeff** () const

Public Attributes

- std::vector< srcCoeff > A_Coeff
- std::vector< srcCoeff > B_Coeff
- Eigen::Vector3d srcPosition
- Eigen::Vector3d srcDirection

2.14.1 Detailed Description

The srcStruct struct describes the scalor potential for an individual source.

The field given by a sources is the negative gradient of a scalor potential PHI according to the equation: PHI = sum((A_coeff(n)∗r^n + B_coeff(n)∗r^(-n-1))∗P_n(cos(theta) , n= 1..inf) where r is the distance from the point of interest to the source position. cos(theta) is the angle between the r vector and the source direction. P_n is a legandre polynomial of order n. A_coeff and B_Coeff define the contributions of each field shape. Here we only keep the first terms in the summation. Note: the length of A_Coeff and B_Coeff need not be the same.

2.14.2 Member Data Documentation

2.14.2.1 std::vector<**srcCoeff**> ScalorPotential::srcStruct::A_Coeff

Ordered coefficients each associated with distances to an increasing positive power.

2.14.2.2 std::vector<**srcCoeff**> ScalorPotential::srcStruct::B_Coeff

Ordered coefficients each associated with distances to an increasing negative power.

2.14.2.3 Eigen::Vector3d ScalorPotential::srcStruct::srcDirection

Source Heading. Should be a unit vector

2.14.2.4 Eigen::Vector3d ScalorPotential::srcStruct::srcPosition

Source Position in meters

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

- scalorPotential.h
- scalorPotential.cpp

Index