GeneralBrokenLinesTest

API Documentation

August 5, 2011

Contents

1 Module gblfit

Track fit with general broken lines.

Created on Jul 27, 2011

Author: kleinwrt

1.1 Variables

Name	Description
__package__	**Value:** `None`

1.2 Class GblPoint

object

gblfit.GblPoint

User supplied point on (initial) trajectory.

Must have jacobians for propagation to previous or next point with offsets (first, last point, points with scatterer). May have:

1. Measurement (1D or 2D)
2. Scatterer (thin, 2D kinks)
3. Additional local parameters (with derivatives). Fitted together with track parameters.
4. Additional global parameters (with labels and derivatives). Not fitted, only passed on to (binary) file for fitting with Millepede-II.

1.2.1 Methods

__init__(*self*)

Create new point.

Overrides: object.__init__

addMeasurement(*self*, *aMeasurement*)

Add a mesurement to a point.

Parameters

`aMeasurement:` measurement (projection residuals, precision)

(type=list(matrix(float)))

hasMeasurement(*self*)

Check point for a measurement.

Return Value

bool

getMeasurement(*self*)

Retrieve measurement of a point.

Return Value

measurement (projection residuals, precision)

(type=list(matrix(float)))

addScatterer(*self*, *aScatterer*)

Add a (thin) scatterer to a point.

Parameters

`aScatterer`: scatterer (kinks, precision)

(type=list(matrix(float)))

hasScatterer(*self*)

Check point for a scatterer.

Return Value

bool

getScatterer(*self*)

Retrieve scatterer of a point.

Return Value

scatterer (kinks, precision)

(type=list(matrix(float)))

addLocals(*self*, *derivatives*)

Add local derivatives.

Parameters

`derivatives`: local derivatives

(type=list(matrix(float)))

addGlobals(*self*, *labels*, *derivatives*)

Add global derivatives.

Parameters

`labels`: global labels

(type=list(matrix(int)))

`derivatives`: global derivatives

(type=list(matrix(float)))

getNumLocals(*self*)

Get number of local derivatives.

Return Value

Number of local derivatives at point

(type=int)

getLocalDerivatives(*self*)

Get local derivatives.

Return Value

local derivatives

(type=matrix(float))

getGlobalLabels(*self*)

Get global labels.

Return Value

lglobal labels

(type=matrix(int))

getGlobalDerivatives(*self*)

Get global derivatives.

Return Value

global derivatives

(type=matrix(float))

setLabel(*self*, *aLabel*)

Define label of a point.

Parameters

`aLabel:` label

(type=int)

getLabel(*self*)

Retrieve label of a point.

Return Value

label

(type=int)

setOffset(*self*, *anOffset*, *aPrev*, *aNext*)

Define offset of a point and references to previous and next point with offset.

Parameters

`anOffset`: offset number (at point (>=0) or at next point with offset (<0))

(type=int)

`aPrev`: label of previous point with offset

(type=int)

`aNext`: label of next point with offset

(type=int)

getOffset(*self*)

Get offset of a point.

Return Value

offset number (at point (>=0) or at next point with offset (<0))

(type=int)

queryJacobians(*self*)

Query point for labels of enclosing offsets.

Return Value

labels of previous and next point with offsets

(type=pair(int))

addJacobians(*self*, *twoJacobians*)

Add jacobians to enclosing offsets.

Parameters

`twoJacobians`: jacobians for propagation to previous and next point with offsets

(type=pair(matrix(float)))

getDerivatives(*self*, *index*)

Get derivatives for locally linearized track model (backward or forward propagation).

Parameters

`index`: 0 (previous) or 1 (next point with offsets)

(type=int)

Return Value

derivatives

(type=list(matrix(float)))

printPoint(*self*)

Print point.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.2.2 Properties

Name	Description
Inherited from object __class__	

1.3 Class GblData

object
gblfit.GblData

Data (block) containing value, precision and derivatives for measurements and kinks.

Created from attributes of GblPoints, used to construct linear equation system for track fit.

1.3.1 Methods

__init__(*self*)

Create new data.

Overrides: object.__init__

addDerivatives(*self*, *aLabel*, *aMeas*, *aPrec*, *parCurv*, *derCurv*, *aDim*, *firstParBand*, *derBand*, *firstParLocal*=`0`, *derLocal*=`[]`, *labGlobal*=`[]`, *derGlobal*=`[]`)

Add derivatives to data (block). Generate lists of labels.

Parameters

`aLabel`: label of corresponding point

(type=int)

`aMeas`: value

(type=float)

`aPrec`: precision

(type=float)

`parCurv`: label for 'curvature' parameter (with (1) or without (0) 'curvature')

(type=int)

`derCurv`: derivative vs 'curvature'

(type=list(float))

`firstParBand`: label of first band parameter (offset)

(type=int)

`derBand`: derivatives vs band parameters (offsets)

(type=list(float))

`firstParLocal`: label of first local parameter

(type=int)

`labGlobal`: labels of global parameters

(type=list(int))

`derGlobal`: derivatives vs global parameters

(type=list(float))

getMatrices(*self*)

Calculate compressed matrix and right hand side from data.

Return Value

indices, compressed right hand side and matrix

(type=list)

setPrediction(*self*, *aVector*)

Calculate prediction for data from fit.

Parameters

`aVector:` values of fit parameters

(type=vector(float))

setDownWeighting(*self*, *aMethod*)

Outlier down weighting with M-estimators.

Parameters

`aMethod:` method (1=Tukey, 2=Huber, 3=Cauchy)

(type=int)

Return Value

weight (0..1)

(type=float)

getChi2(*self*)

Calculate Chi2 (contribution) from data.

Return Value

Chi2

(type=float)

toRecord(*self*)

Get data components (for copying to MP binaty record)

Return Value

data components

(type=list)

fromRecord(*self*, *dataList*)

Set data components (from MP binaty record)

Parameters

`dataList:` data components

(type=list)

analyzeData(*self*, *maxBand*)

Analyze labels of fit parameters to determine number of parameters and border size with given maximal band width.

Parameters

`maxBand:` maximal band width

(type=int)

Return Value

number of parameters and border size (from this data)

(type=pair(int))

printData(*self*)

Print data.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.3.2 Properties

Name	Description
Inherited from object __class__	

1.4 Class GblTrajectory

object ─┐
gblfit.GblTrajectory

(section) General Broken Lines Trajectory.

For a track with an initial trajectory from a prefit of the measurements (internal seed) or an external prediction (external seed) the description of multiple scattering is added by offsets in a local system. Along the initial trajectory points are defined with can describe a measurement or a (thin) scatterer or both. The refit provides corrections to the local track parameters (in the local system) and the corresponding covariance matrix at any of those points.

The broken lines trajectory is defined by (2D) offsets at the first and last point and all points with a scatterer. The prediction for a measurement is obtained by interpolation of the enclosing offsets and for triplets of adjacent offsets kink angles are determined. This requires for all points the jacobians for propagation to the previous and next offset.

Additional local or global parameters can be added and the trajectories can be written to special binary files for calibration and alignment with Millepede-II. (V. Blobel, NIM A, 566 (2006), pp. 5-13).

The conventions for the coordinate systems follow: Derivation of Jacobians for the propagation of covariance matrices of track parameters in homogeneous magnetic fields A. Strandlie, W. Wittek, NIM A, 566 (2006) 687-698.

(section) Calling sequence:

1. Create trajectory:
   ```
   traj = GblTrajectory()
   ```
2. For all points on initial trajectory
   - Create points and add appropiate attributes:
     ```
     point = GblPoint()
     point.addMeasurement(..)
     point.addScatterer(..)
     point.addLocals(..)
     point.addGlobals(..)
     ```
   - Add point (ordered by arc length) to trajectory, get label of point:
     ```
     label = traj.addPoint(point)
     ```
3. Define (points with) offsets:
   ```
   traj.defineOffsets()
   ```
4. Optionally add external seed:
   ```
   traj.addExternalSeed(..)
   ```
5. For all points on initial trajectory
   - Query for required jacobians:
     ```
     traj.queryJacobians(label)
     ```
   - Add requested jacobians:
     ```
     traj.addJacobians(label,..)
     ```
6. Optionally write trajectory to MP binary file:
   ```
   traj.milleOut(..)
   ```
7. Fit trajectory, bet Chi2, Ndf (and weight lost by M-estimators):
   ```
   [..] = traj.fit()
   ```
8. For any point on inital trajectory
   - Get corrections and covariance matrix for track parameters:
     ```
     [..] = traj.getResults(label)
     ```

Alternatively trajectories can by read from MP binary files and fitted. As the points on the initial trajectory are not stored in this files results at points (corrections, covariance matrix) are not available.

1.4.1 Methods

__init__(*self*, *hasCurv*=`True`, *aDim*=`[0, 1]`)

Create new trajectory.

Overrides: object.__init__

addPoint(*self*, *point*)

Add point to trajectory. Points have to be ordered in arc length.

Parameters

`point`: point to be added

(type=GblPoint)

Return Value

label of added point

(type=int)

getNumPoints(*self*)

Get number of points on trajectory.

Return Value

number of points

(type=int)

addExternalSeed(*self*, *aLabel*, *aSeed*)

Add external seed to trajectory.

Parameters

`aLabel`: label of point with external seed

(type=int)

`aSeed`: seed (covariance matrix of track parameters at point)

(type=list(matrix(float)))

queryJacobians(*self*, *aLabel*)

Query point for adjacent scatterers.

Parameters

`aLabel`: label of point

(type=int)

Return Value

labels of previous and next point with offset

(type=pair(int))

addJacobians(*self*, *aLabel*, *twoJacobians*)

Provide point for jacobians to adjacent scatterers.

Parameters

`aLabel`: label of point

(type=int)

`twoJacobians`: jacobians for propagation to previous and next point with offset

(type=pair(matrix(float)))

defineOffsets(*self*)

Define offsets from list of points.

dump(*self*)

Dump trajectory.

milleOut(*self*, *aFile*)

Write (data blocks of) trajectory to MP (binary) file.

Parameters

`aFile`: MP file

(type=file)

milleIn(*self*, *aFile*)

Read (data blocks of) trajectory from MP (binary) file.

Parameters

`aFile`: MP file

(type=file)

getResults(*self*, *aLabel*)

Get results (corrections, covarinace matrix) at point in forward or backward direction.

Parameters

`aLabel`: signed label of point (<0 backward, >0 forward)

(type=int)

Return Value

correction vector, covarinace matrix for track parameters

(type=list)

fit(*self*, *optionList*=`''`)

Perform fit of trajectory.

Parameters

`optionList`: M-estimators to be used (one iteration per character)

(type=string)

Return Value

Chi2, Ndf, loss of weight from fit ([0., -1, 0.] if fit failed)

(type=list)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.4.2 Properties

Name	Description
Inherited from object __class__	

2 Module gblnum

Algebra for linear equation system with bordered band matrix.

Created on Jul 27, 2011

Author: kleinwrt

2.1 Variables

Name	Description
__package__	**Value:** `None`

2.2 Class BorderedBandMatrix

object ─┐
gblnum.BorderedBandMatrix

(Symmetric) Bordered Band Matrix.

Separate storage of border, mixed and band parts. Example for matrix size=8 with border size and band width of two:

```
+-                               -+
|  B11 B12 M13 M14 M15 M16 M17 M18 |
|  B12 B22 M23 M24 M25 M26 M27 M28 |
|  M13 M23 C33 C34 C35  0.  0.  0. |
|  M14 M24 C34 C44 C45 C46  0.  0. |
|  M15 M25 C35 C45 C55 C56 C57  0. |
|  M16 M26  0. C46 C56 C66 C67 C68 |
|  M17 M27  0.  0. C57 C67 C77 C78 |
|  M18 M28  0.  0.  0. C68 C78 C88 |
+-                               -+
```

Is stored as:

```
+-         -+     +-                         -+
|  B11 B12  |     |  M13 M14 M15 M16 M17 M18  |
|  B12 B22  |     |  M23 M24 M25 M26 M27 M28  |
+-         -+     +-                         -+

                  +-                         -+
                  |  C33 C44 C55 C66 C77 C88  |
                  |  C34 C45 C56 C67 C78  0.  |
                  |  C35 C46 C57 C68  0.  0.  |
                  +-                         -+
```

2.2.1 Methods

__init__(*self*, *nSize*, *nBorder*=1, *nBand*=5)

Create new BBmatrix.

Parameters

`nSize:` size of matrix

(type=int)

`nBorder:` size of border (default: 1, 'curvature')

(type=int)

`nBand:` (maximal) band width (5)

(type=int)

Overrides: object.__init__

addBlockMatrix(*self*, *aIndex*, *aMatrix*)

Add (compressed) block to BBmatrix:

```
BBmatrix(aIndex(i),aIndex(j)) += aMatrix(i,j)
```

Parameters

`aIndex:` list of indices

(type=list(int))

`aMatrix:` (compressed) matrix

(type=matrix(float))

getBlockMatrix(*self*, *aIndex*)

Retrieve (compressed) block from BBmatrix:

```
aMatrix(i,j) = BBmatrix(aIndex(i),aIndex(j))
```

Parameters

`aIndex`: list of indices

(type=list(int))

Return Value

(compressed) matrix

(type=matrix(float))

printMatrix(*self*)

Print BBmatrix.

solveAndInvertBorderedBand(*self*, *aRightHandSide*)

Solve linear equation A*x=b system with BBmatrix A, calculate BB part of inverse of A.

Parameters

`aRightHandSide`: right hand side ’b’ of linear equation system

(type=vector(float))

Return Value

solution

(type=vector(float))

Raises

`ZeroDivisionError` Band matrix is not positive definite

Note: BBmatrix is replaced by BB part of it’s inverse

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

2.2.2 Properties

Name	Description
Inherited from object __class__	

3 Module gbltst

Simple Test Program for General Broken Lines.

Created on Jul 27, 2011

Author: kleinwrt

3.1 Functions

example1()

Create points on initial trajectory, create trajectory from points, fit and write trajectory to MP-II binary file, get track parameter corrections and covariance matrix at points.

Equidistant measurement layers and thin scatterers, propagation with simple jacobian (quadratic in arc length differences).

example2()

Read trajectory from MP-II binary file and refit.

3.2 Variables

Name	Description
__package__	**Value:** `None`

4 Module mille

Input/output of MP-II binary records.

Created on Aug 1, 2011

Author: kleinwrt

4.1 Variables

Name	Description
__package__	**Value:** None

4.2 Class MilleRecord

```
object ─┐
        │
      mille.MilleRecord
```

Millepede-II (binary) record.

Containing information for local (track) and global fit.

The data blocks are collected in two arrays, a real array and an integer array, of same length. The content of the arrays:

```
     real array              integer array
 0   0.0                     error count (this record)
 1   RMEAS, measured value   0                            __iMeas   -+
 2   local derivative        index of local derivative               |
 3   local derivative        index of local derivative               |
 4    ...                                                            | block
     SIGMA, error (>0)        0                           __ iErr    |
     global derivative       label of global derivative              |
     global derivative       label of global derivative             -+
     RMEAS, measured value   0                            __position
     local derivative        index of local derivative
     local derivative        index of local derivative
     ...
     SIGMA, error            0
     global derivative       label of global derivative
     global derivative       label of global derivative
     ...
     global derivative       label of global derivative   __recLen
```

4.2.1 Methods

__init__(*self*)

Create MP-II binary record.

Overrides: object.__init__

addData(*self*, *dataList*)

Add data block to (end of) record.

Parameters

`dataList`: list with measurement, error, labels and derivatives

(type=list)

getData(*self*)

Get data block from current position in record.

Return Value

list with measurement, error, labels and derivatives

(type=list)

addSpecial(*self*, *iSpecial*, *gSpecial*, *aTag*=0)

Add special data block to record:

```
real array              integer array
0.0                     0
-float(NSP)-0.1*tag     0   ! indicates special data of length NSP
following NSP floating and NSP integer data
```

Parameters

`iSpecial`: list of labels

(type=list(int))

`gSpecial`: list of values

(type=list(float))

`aTag`: tag (1: external seed (matrix))

(type=int)

getSpecial(*self*)

Get special data block from current position.

Return Value

special data block (list of labels, list of values)

(type=list)

printRecord(*self*)

Print record.

writeRecord(*self*, *aFile*)

Write record to file.

Parameters

`aFile:` (binary) file

(type=file)

readRecord(*self*, *aFile*)

Read record from file.

Parameters

`aFile:` (binary) file

(type=file)

moreData(*self*)

Locate next data block.

Return Value

next block exists

(type=bool)

specialDataTag(*self*)

Get special data tag from block.

Return Value

tag or -1 for ordinary data block

(type=int)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

4.2.2 Properties

Name	Description
Inherited from object __class__	

Index