GNSS Adjustment of Network

Generated by Doxygen 1.9.1

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 /home/kongkong/gnss_adjustment_network/src/adjustment_impl.c File Reference

implementation of adjustment. firstly, assemble corresponding linear system based on different conditions; then, solve corresponding equation; finally, obtain the solution.

```
#include "../include/adjustment_impl.h"
```

Functions

- void LSEUpdateSolution (Config *var_conf, double *lse_sol, double *lse_b, int lse_size_row, int lse_size_column, Solution *var_solution)

  write solution to solution struct and computing norm of least-square equation
- double lse_norm_2 (double *a, int dimension)

  computing L2 norm of vector a
- void ImplNetworkAdjustment (Config *var_conf, AdjGraph *var_data, Solution *var_solution)

  solve corresponding least-squares equation with configure conditions, mainly divides into three categories, equal weight, diagonal weight and full weight. ATTENTION: FULL WEIGHT CANNOT WORK CURRENTLY!
- void ImplNetworkAdjustment_0 (Config *var_conf, AdjGraph *var_data, Solution *var_solution)

  gnss adjustment of network with equal weight
- void ImplNetworkAdjustment_1 (Config *var_conf, AdjGraph *var_data, Solution *var_solution)

  gnss adjustment of network with diagonal weight
- void ImplNetworkAdjustment_2 (Config *var_conf, AdjGraph *var_data, Solution *var_solution)

  gnss adjustment of network with full weight, ATTENTION: CANNOT WORK CURRENTLY!
- void InitLSE (double *sol, double *rhs, double *residual, double **mat, int row, int column)

  initialize least-squares equation, initialize coefficient matrix and right-hand side vector to 0

2.1.1 Detailed Description

implementation of adjustment. firstly, assemble corresponding linear system based on different conditions; then, solve corresponding equation; finally, obtain the solution.

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.1.2 Function Documentation

2.1.2.1 ImplNetworkAdjustment()

```
void ImplNetworkAdjustment (
            Config * var_conf,
            AdjGraph * var_data,
            Solution * var_solution )
```

solve corresponding least-squares equation with configure conditions, mainly divides into three categories, equal weight, diagonal weight and full weight. ATTENTION: FULL WEIGHT CANNOT WORK CURRENTLY!

Parameters

in	*var_conf*	configure data of gnss adjustment of network
in	*var_data*	graph data
in,out	*var_solution*	solution of least-squares equation

2.1.2.2 ImplNetworkAdjustment_0()

```
void ImplNetworkAdjustment_0 (
            Config * var_conf,
```

```
            AdjGraph * var_data,
            Solution * var_solution )
```

gnss adjustment of network with equal weight

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*var_data*	graph data for gnss adjustment of network
in,out	*var_solution*	solution of gnss adjustment of network

2.1.2.3 ImplNetworkAdjustment_1()

```
void ImplNetworkAdjustment_1 (
            Config * var_conf,
            AdjGraph * var_data,
            Solution * var_solution )
```

gnss adjustment of network with diagonal weight

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*var_data*	graph data for gnss adjustment of network
in,out	*var_solution*	solution of gnss adjustment of network

2.1.2.4 ImplNetworkAdjustment_2()

```
void ImplNetworkAdjustment_2 (
            Config * var_conf,
            AdjGraph * var_data,
            Solution * var_solution )
```

gnss adjustment of network with full weight, ATTENTION: CANNOT WORK CURRENTLY!

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*var_data*	graph data for gnss adjustment of network
in,out	*var_solution*	solution of gnss adjustment of network

2.1.2.5 InitLSE()

```
void InitLSE (
            double * sol,
            double * rhs,
            double * residual,
            double ** mat,
            int row,
            int column )
```

initialize least-squares equation, initialize coefficient matrix and right-hand side vector to 0

Parameters

in,out	*sol*	solution vector
in,out	*rhs*	right-hand side vector
in,out	*residual*	residual vector
in,out	*mat*	coefficient matrix
in	*row*	row size of coefficient matrix
in	*column*	column size of coefficient matrix

2.1.2.6 lse_norm_2()

```
double lse_norm_2 (
            double * a,
            int dimension )
```

computing L2 norm of vector a

Parameters

in	*a*	vector
in	*dimension*	dimension of vector

Returns

double norm of vector a

2.1.2.7 LSEUpdateSolution()

```
void LSEUpdateSolution (
            Config * var_conf,
            double * lse_sol,
            double * lse_b,
            int lse_size_row,
            int lse_size_column,
            Solution * var_solution )
```

write solution to solution struct and computing norm of least-square equation

Parameters

in	*var_conf*	configure data of gnss adjustment of network
in	*lse_sol*	solution of least-squares equation
in	*lse_b*	right-hand side of linear system
in	*lse_size_row*	row size of linear system
in	*lse_size_column*	column size of linear system
in,out	*var_solution*	solution struct

2.2 /home/kongkong/gnss_adjustment_network/src/graph_impl.c File Reference

graph data structure with adjacency list

`#include "../include/graph_impl.h"`

Functions

- void InitializeLinkedList (AdjList *pList)

  initialize linked list, the head pointer of linked list to NULL
- AdjListNode * AdjListAddNode (int dest, double *weight, int weight_size)

  information of added edge, destinated node of edge and weight data of edge, assigning values to the struct
- AdjGraph * GraphGeneration (int n)

  graph generation with n vertices and n linked lists
- void GraphInsert (AdjGraph *graph, int *location, double *weight, int weight_size)

  add edge to graph, the numbering of the two vertices of an edge and weight data of the edge. source node is the first element of location array, destinated node is the second element of the edge. check if the head pointer of the lined list is NULL, if NULL, add edge to new adjacency list, else, add edge to tail of current adjacency list
- void GraphDisplay (AdjGraph *graph)

  display the graph with adjacency list, graph traversal by vertex
- void GraphDestroy (AdjGraph *graph)

  free memory of graph data structure

2.2.1 Detailed Description

graph data structure with adjacency list

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

2.2.2 Function Documentation

2.2.2.1 AdjListAddNode()

```
AdjListNode* AdjListAddNode (
            int dest,
            double * weight,
            int weight_size )
```

information of added edge, destinated node of edge and weight data of edge, assigning values to the struct

Parameters

in	*dest*	destinated node of edge
in	*weight*	weight of edge
in	*weight_size*	size of weight data in edge

Returns

AdjListNode∗ added edge

2.2.2.2 GraphDestroy()

```
void GraphDestroy (
            AdjGraph * graph )
```

free memory of graph data structure

Parameters

in,out	*graph*	grph data structure

2.2.2.3 GraphDisplay()

```
void GraphDisplay (
            AdjGraph * graph )
```

display the graph with adjacency list, graph traversal by vertex

Parameters

in	*graph*	graph structure

2.2.2.4 GraphGeneration()

```
AdjGraph* GraphGeneration (
            int n )
```

graph generation with n vertices and n linked lists

Parameters

in	*n*	vertices of a graph

Returns

AdjGraph∗ graph data structure

2.2.2.5 GraphInsert()

```
void GraphInsert (
            AdjGraph * graph,
            int * location,
            double * weight,
            int weight_size )
```

add edge to graph, the numbering of the two vertices of an edge and weight data of the edge. source node is the first element of location array, destinated node is the second element of the edge. check if the head pointer of the lined list is NULL, if NULL, add edge to new adjacency list, else, add edge to tail of current adjacency list

Parameters

in,out	*graph*	graph data structure
in	*location*	source vertex and destinated vertes
in	*weight*	weight data of edge
in	*weight_size*	size of weight data

2.2.2.6 InitializeLinkedList()

```
void InitializeLinkedList (
            AdjList * pList )
```

initialize linked list, the head pointer of linked list to NULL

Parameters

in,out	*pList*	linked list

2.3 /home/kongkong/gnss_adjustment_network/src/lse_solver.c File Reference

linear algebra solver, linear system assembling, perform QR decompostioin with givens rotation, solving upper triangular linear system with Gaussian elimination

```
#include "../include/lse_solver.h"
```

Functions

- void update_linsys_full_weight (double **mat_Cov, int row_start, int column_start, double **lse_A, double *lse_b)

  updating linear system with full weight
- void assemble_mat_Cov (double **data_Baseline, int row_start, double **mat_Cov)

  assembling variance-covariance matrix with baseline source data
- void LSEGauss (double **mat, double *rhs, int column, double *sol)

  solving upper triangular linear system with gaussian elimination
- void givens_rotation_impl (double val_a, double val_b, int index_i, int index_j, int row, int column, double **mat, double *rhs)

  givens rotation, which can make selective non-zero element become zero element
- void LSEGivens (double **mat, double *rhs, int row, int column)

  solving least-squares equation with QR decomposition, performing QR decomposition with givens rotation
- int IsBaseStation (int code, int *code_BaseStation, int cnt_BaseStation)

  check if variant code is base station code, if true, function returns 1; else, function returns 0
- void LSEAssemble_2 (int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, int *code_BaseStation, double **coo_BaseStation, int cnt_BaseStation, int coo_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column)

  assemble corresponding least-squares equation with full weight
- void LSEAssemble_1 (int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, int *code_BaseStation, double **coo_BaseStation, int cnt_BaseStation, int coo_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column)

  assemble corresponding least-squares equation with diagonal weight
- void LSEAssemble_0 (int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, int *code_BaseStation, double **coo_BaseStation, int cnt_BaseStation, int coo_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column)

  assemble corresponding least-squares equation with equal weight
- void LSESolver_0 (Config *var_conf, int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column, double *lse_sol, double *lse_residual)

  equal weight least-squares equation solver
- void LSESolver_1 (Config *var_conf, int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column, double *lse_sol, double *lse_residual)

  diagonal weight least-squares equation solver
- void LSESolver_2 (Config *var_conf, int **vertex_enum, double **data_BaseLine, int data_row, int data_column, int vertex_column, double **lse_A, double *lse_b, int lse_size_row, int lse_size_column, double *lse_sol, double *lse_residual)

  full weight least-squares equation solver

2.3.1 Detailed Description

linear algebra solver, linear system assembling, perform QR decompostioin with givens rotation, solving upper triangular linear system with Gaussian elimination

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

2.3.2 Function Documentation

2.3.2.1 assemble_mat_Cov()

```
void assemble_mat_Cov (
            double ** data_Baseline,
            int row_start,
            double ** mat_Cov )
```

assembling variance-covariance matrix with baseline source data

Parameters

in	*data_Baseline*	baseline source data
in	*row_start*	row index of block variance-covariance matrix
in,out	*mat_Cov*	variance-covariance matrix

2.3.2.2 givens_rotation_impl()

```
void givens_rotation_impl (
            double val_a,
            double val_b,
```

```
            int index_i,
            int index_j,
            int row,
            int column,
            double ** mat,
            double * rhs )
```

givens rotation, which can make selective non-zero element become zero element

Parameters

in	*val_a*	selective element of matrix
in	*val_b*	selective element of matrix
in	*index_i*	row index of val_a
in	*index_j*	row index of val_b
in	*row*	row size of matrix
in	*column*	column size of matrix
in,out	*mat*	matrix and transformed matrix
in,out	*rhs*	right-hand side vector and transformed right-hand side vector

2.3.2.3 IsBaseStation()

```
int IsBaseStation (
            int code,
            int * code_BaseStation,
            int cnt_BaseStation )
```

check if variant code is base station code, if true, function returns 1; else, function returns 0

Parameters

in	*code*	judged code of station
in	*code_BaseStation*	array of base station
in	*cnt_BaseStation*	size of array of base station

Returns

int(1) is base station, int(0) is not base station

2.3.2.4 LSEAssemble_0()

```
void LSEAssemble_0 (
            int ** vertex_enum,
            double ** data_BaseLine,
```

```
            int data_row,
            int data_column,
            int vertex_column,
            int * code_BaseStation,
            double ** coo_BaseStation,
            int cnt_BaseStation,
            int coo_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column )
```

assemble corresponding least-squares equation with equal weight

Parameters

in	*vertex_enum*	vertex enumeration of graph
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data
in	*vertex_column*	column size of edge vertex, equals to 2
in	*code_BaseStation*	base station code
in	*coo_BaseStation*	coordinate of base station
in	*cnt_BaseStation*	count of base station
in	*coo_column*	column size of coordinate
in,out	*lse_A*	least-squares equation coefficient matrix
in,out	*lse_b*	least-squares right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation

2.3.2.5 LSEAssemble_1()

```
void LSEAssemble_1 (
            int ** vertex_enum,
            double ** data_BaseLine,
            int data_row,
            int data_column,
            int vertex_column,
            int * code_BaseStation,
            double ** coo_BaseStation,
            int cnt_BaseStation,
            int coo_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column )
```

assemble corresponding least-squares equation with diagonal weight

Parameters

in	*vertex_enum*	vertex enumeration of graph
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data
in	*vertex_column*	column size of edge vertex, equals to 2
in	*code_BaseStation*	base station code
in	*coo_BaseStation*	coordinate of base station
in	*cnt_BaseStation*	count of base station
in	*coo_column*	column size of coordinate
in,out	*lse_A*	least-squares equation coefficient matrix
in,out	*lse_b*	least-squares right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation

2.3.2.6 LSEAssemble_2()

```
void LSEAssemble_2 (
            int ** vertex_enum,
            double ** data_BaseLine,
            int data_row,
            int data_column,
            int vertex_column,
            int * code_BaseStation,
            double ** coo_BaseStation,
            int cnt_BaseStation,
            int coo_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column )
```

assemble corresponding least-squares equation with full weight

Parameters

in	*vertex_enum*	vertex enumeration of graph
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data
in	*vertex_column*	column size of edge vertex, equals to 2
in	*code_BaseStation*	base station code
in	*coo_BaseStation*	coordinate of base station
in	*cnt_BaseStation*	count of base station
in	*coo_column*	column size of coordinate
in,out	*lse_A*	least-squares equation coefficient matrix
in,out	*lse_b*	least-squares right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation

2.3.2.7 LSEGauss()

```
void LSEGauss (
            double ** mat,
            double * rhs,
            int column,
            double * sol )
```

solving upper triangular linear system with gaussian elimination

Parameters

in	*mat*	coefficient matrix of linear system
in	*rhs*	right-hand side vector of linear system
in	*column*	dimension of linear system
in,out	*sol*	solution to linear system

2.3.2.8 LSEGivens()

```
void LSEGivens (
            double ** mat,
            double * rhs,
            int row,
            int column )
```

solving least-squares equation with QR decomposition, performing QR decomposition with givens rotation

Parameters

in,out	*mat*	coefficient matrix of least-squares equation
in,out	*rhs*	right-hand side vector of least-squares equation
in	*row*	row size of least-squares equation
out	*column*	column size of least-squares equation

2.3.2.9 LSESolver_0()

```
void LSESolver_0 (
            Config * var_conf,
            int ** vertex_enum,
            double ** data_BaseLine,
            int data_row,
            int data_column,
```

```
            int vertex_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column,
            double * lse_sol,
            double * lse_residual )
```

equal weight least-squares equation solver

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*vertex_enum*	vertex enumeration
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data
in	*vertex_column*	column size of edge vertex, equals to 2
in	*lse_A*	coefficient matrix of least-squares equation
in	*lse_b*	right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation
in,out	*lse_sol*	solution to least-squares equation
in,out	*lse_residual*	residual of least-squares equation

2.3.2.10 LSESolver_1()

```
void LSESolver_1 (
            Config * var_conf,
            int ** vertex_enum,
            double ** data_BaseLine,
            int data_row,
            int data_column,
            int vertex_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column,
            double * lse_sol,
            double * lse_residual )
```

diagonal weight least-squares equation solver

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*vertex_enum*	vertex enumeration
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data

Parameters

in	*vertex_column*	column size of edge vertex, equals to 2
in	*lse_A*	coefficient matrix of least-squares equation
in	*lse_b*	right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation
in,out	*lse_sol*	solution to least-squares equation
in,out	*lse_residual*	residual of least-squares equation

2.3.2.11 LSESolver_2()

```
void LSESolver_2 (
            Config * var_conf,
            int ** vertex_enum,
            double ** data_BaseLine,
            int data_row,
            int data_column,
            int vertex_column,
            double ** lse_A,
            double * lse_b,
            int lse_size_row,
            int lse_size_column,
            double * lse_sol,
            double * lse_residual )
```

full weight least-squares equation solver

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*vertex_enum*	vertex enumeration
in	*data_BaseLine*	baseline source data
in	*data_row*	row size of baseline data
in	*data_column*	column size of baseline data
in	*vertex_column*	column size of edge vertex, equals to 2
in	*lse_A*	coefficient matrix of least-squares equation
in	*lse_b*	right-hand side vector
in	*lse_size_row*	row size of least-squares equation
in	*lse_size_column*	column size of least-squares equation
in,out	*lse_sol*	solution to least-squares equation
in,out	*lse_residual*	residual of least-squares equation

2.3.2.12 update_linsys_full_weight()

```
void update_linsys_full_weight (
```

```
double ** mat_Cov,
int row_start,
int column_start,
double ** lse_A,
double * lse_b )
```

updating linear system with full weight

Parameters

in	*mat_Cov*	variance-covariance matrix
in	*row_start*	row index of block matrix
in	*column_start*	column index of block matrix
in,out	*lse_A*	linear system coefficient matrix
in,out	*lse_b*	linear system right-hand side vector

2.4 /home/kongkong/gnss_adjustment_network/src/main.c File Reference

main function of adjustment of GNSS network

```
#include "../include/main.h"
```

Functions

- int main (int argc, char **argv)
  command line parameters, contain path of configure file and path of source data file

2.4.1 Detailed Description

main function of adjustment of GNSS network

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

2.4.2 Function Documentation

2.4.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

command line parameters, contain path of configure file and path of source data file

Parameters

in	*argc*	command line parameter
in	*argv*	path of file

< path of configure file

< path of data file

2.5 /home/kongkong/gnss_adjustment_network/src/main_function.c File Reference

function in main function, mainly involves configure file process and source data process

```
#include "../include/main.h"
```

Functions

- void ProcessSourceData (FILE *fp_data, char *file_data, Config *var_conf, AdjGraph *var_data)

  source data process, assigning values to graph data structure with source data file
- void EncodeStationId (Config *var_conf, int len_node, int *node)

  encoding base station and rover station sequentially start from 0
- void free_conf (Config *adjust_conf)

  free memory of configure struct
- void ProcessConfigureFile_sort (FILE *fp_conf, char *file_conf, Config *var_conf)

  process sorted configure file, this type of file contains only data information, such as, count of stations, base station coordinate and etc. assigning values to configure struct with configure file
- void ProcessConfigureFile (FILE *fp_conf, char *file_conf, Config *var_conf)

  process configure file, assigning values to configure struct with configure file

2.5.1 Detailed Description

function in main function, mainly involves configure file process and source data process

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

2.5.2 Function Documentation

2.5.2.1 EncodeStationId()

```
void EncodeStationId (
            Config * var_conf,
            int len_node,
            int * node )
```

encoding base station and rover station sequentially start from 0

Parameters

in	*var_conf*	configure data for gnss adjustment of network
in	*len_node*	count of stations
in,out	*node*	array of station code

2.5.2.2 free_conf()

```
void free_conf (
            Config * adjust_conf )
```

free memory of configure struct

Parameters

in,out	*adjust_conf*	configure data for gnss adjustment of network

2.5.2.3 ProcessConfigureFile()

```
void ProcessConfigureFile (
            FILE * fp_conf,
            char * file_conf,
            Config * var_conf )
```

process configure file, assigning values to configure struct with configure file

Parameters

in	*fp_conf*	FILE pointer
in	*file_conf*	path of configure file
in,out	*var_conf*	configure struct

2.5.2.4 ProcessConfigureFile_sort()

```
void ProcessConfigureFile_sort (
            FILE * fp_conf,
            char * file_conf,
            Config * var_conf )
```

process sorted configure file, this type of file contains only data information, such as, count of stations, base station coordinate and etc. assigning values to configure struct with configure file

Parameters

in	*fp_conf*	FILE pointer
in	*file_conf*	path of configure file
in,out	*var_conf*	configure struct

2.5.2.5 ProcessSourceData()

```
void ProcessSourceData (
            FILE * fp_data,
            char * file_data,
            Config * var_conf,
            AdjGraph * var_data )
```

source data process, assigning values to graph data structure with source data file

Parameters

in	*fp_data*	FILE pointer
in	*file_data*	path to source data file
in	*var_conf*	configure data for gnss adjustment of network
in,out	*var_data*	graph data structure

Index