GNSS Baseline Process

Generated by Doxygen 1.9.8

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

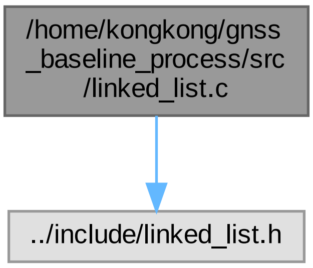
Chapter 2

File Documentation

2.1 /home/kongkong/gnss_baseline_process/src/linked_list.c File Reference

linked list implementation, contains linked list generation, initialize linked list, add item to linked list, destroy linked list, traversal of linked list

`#include "../include/linked_list.h"`
Include dependency graph for linked_list.c:



Functions

- void InitializeLinkedList (List_LinkedList ∗pList)
  initialize linked list
- int IsLinkedListEmpty (const List_LinkedList ∗pList)
  check if linked list is empty, if empty, return 1; else, return 0
- int IsLinkedListFull (const List_LinkedList ∗pList)
  check if linked list if full, if full, return 1; else, return 0
- int AddItemToLinkedList (Item_LinkedList item, List_LinkedList ∗pList)
  add new item to linked list, add to tail of current linked list, if add successfully, return 1, else return 0
- void TraverseLinkedList (const List_LinkedList ∗pList, void(∗pFun)(Item_LinkedList item))
  traversal of linked list, from head of linked list to tail of linked list, with function pointer
- void DestroyLinkedList (List_LinkedList ∗pList)
  free memory of linked list

2.1.1 Detailed Description

linked list implementation, contains linked list generation, initialize linked list, add item to linked list, destroy linked list, traversal of linked list

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.1.2 Function Documentation

2.1.2.1 AddItemToLinkedList()

```
int AddItemToLinkedList (
            Item_LinkedList item,
            List_LinkedList * pList )
```

add new item to linked list, add to tail of current linked list, if add successfully, return 1, else return 0

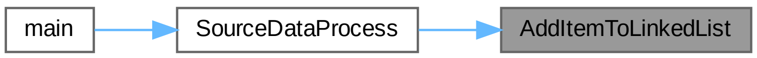
Parameters

in	*item*	struct variant
in,out	*pList*	linked list

Returns

int(1) add successfully, int(0) add unsuccessfully

Here is the caller graph for this function:



2.1.2.2 DestroyLinkedList()

```
void DestroyLinkedList (
            List_LinkedList * pList )
```

free memory of linked list

Parameters

in,out	*pList*	linked list

Here is the caller graph for this function:

main
DestroyLinkedList

2.1.2.3 InitializeLinkedList()

```
void InitializeLinkedList (
            List_LinkedList * pList )
```

initialize linked list

Parameters

in,out	*pList*	linked list

Here is the caller graph for this function:

main
InitializeLinkedList

2.1.2.4 IsLinkedListEmpty()

```
int IsLinkedListEmpty (
            const List_LinkedList * pList )
```

check if linked list is empty, if empty, return 1; else, return 0

Parameters

in	*pList*	linked list

Returns

int(1) empty linked list, int(0) not empty linked list

2.1.2.5 IsLinkedListFull()

```
int IsLinkedListFull (
            const List_LinkedList * pList )
```

check if linked list if full, if full, return 1; else, return 0

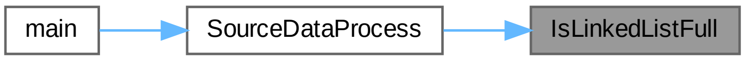
Parameters

in	*pList*	linked list

Returns

int(1) full linked list, int(0) non full linked list

Here is the caller graph for this function:



2.1.2.6 TraverseLinkedList()

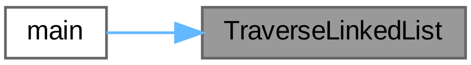
```
void TraverseLinkedList (
            const List_LinkedList * pList,
            void(*)(Item_LinkedList item) pFun )
```

traversal of linked list, from head of linked list to tail of linked list, with function pointer

Parameters

in	*pList*	linked list
in	*pFun*	function pointer

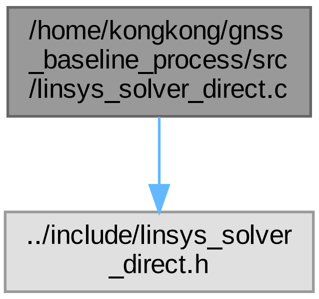
Here is the caller graph for this function:



2.2 /home/kongkong/gnss_baseline_process/src/linsys_solver_direct.c File Reference

```
#include "../include/linsys_solver_direct.h"
```

Include dependency graph for linsys_solver_direct.c:


/home/kongkong/gnss_baseline_process/src/linsys_solver_direct.c
../include/linsys_solver_direct.h

Functions

- void GaussElimination (double ∗∗mat, double ∗rhs, double ∗sol, int n)

  gaussian elimination for solving linear system
- void MatTranspose (double ∗∗mat, double ∗∗trans_mat, int size_row, int size_column)

  computing transpose of matrix
- void MatMatProduct (double ∗∗mat_1, double ∗∗mat_2, double ∗∗mat, int size_row, int size_column, int size_column_2)

  computing matrix by matrix product
- void MatVecProduct (double ∗∗mat, double ∗vec, double ∗sol, int m, int n)

  computing matrix by vector product

2.2.1 Detailed Description

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.2.2 Function Documentation

2.2.2.1 GaussElimination()

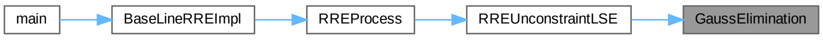
```
void GaussElimination (
             double ** mat,
             double * rhs,
             double * sol,
             int n )
```

gaussian elimination for solving linear system

Parameters

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

Here is the caller graph for this function:



2.2.2.2 MatMatProduct()

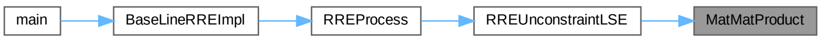
```
void MatMatProduct (
             double ** mat_1,
             double ** mat_2,
             double ** mat,
             int size_row,
             int size_column,
             int size_column_2 )
```

computing matrix by matrix product

Parameters

`in`	*mat_1*	first matrix
`in`	*mat_2*	second matrix
`in,out`	*mat*	mat_1 by mat_2 product
`in`	*size_row*	row size of mat_1
`in`	*size_column*	column size of mat_1
`in`	*size_column↩ _2*	column_size of mat_2

Here is the caller graph for this function:



2.2.2.3 MatTranspose()

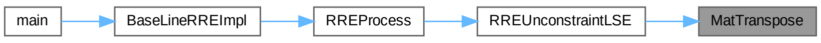
```
void MatTranspose (
              double ** mat,
              double ** trans_mat,
              int size_row,
              int size_column )
```

computing transpose of matrix

Parameters

`in`	*mat*	original matrix
`in,out`	*trans_mat*	transpose of original matrix
`in`	*size_row*	row size of original matrix
`in`	*size_column*	column size of original matrix

Here is the caller graph for this function:



2.2.2.4 MatVecProduct()

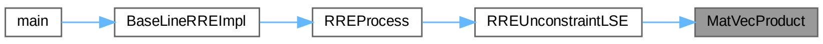
```
void MatVecProduct (
             double ** mat,
             double * vec,
             double * sol,
             int m,
             int n )
```

computing matrix by vector product

Parameters

in	*mat*	original matrix
in	*vec*	original vector
in,out	*sol*	mat by vec product
in	*m*	row size of original matrix
in	*n*	column size of original matrix

Here is the caller graph for this function:

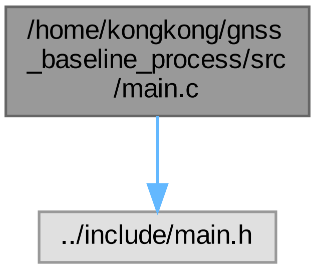


2.3 /home/kongkong/gnss_baseline_process/src/main.c File Reference

main function, baseline data process, fusing valid datas within a time period into a data output

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

Include dependency graph for main.c:



Functions

- void DisplayItem (Item_LinkedList item)
  display struct item
- int main (int argc, char **argv)
  process valid baseline source data

2.3.1 Detailed Description

main function, baseline data process, fusing valid datas within a time period into a data output

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.3.2 Function Documentation

2.3.2.1 DisplayItem()

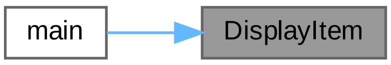
```
void DisplayItem (
             Item_LinkedList item )
```

display struct item

Parameters

in	*item*	struct variable

Here is the caller graph for this function:



2.3.2.2 main()

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

process valid baseline source data

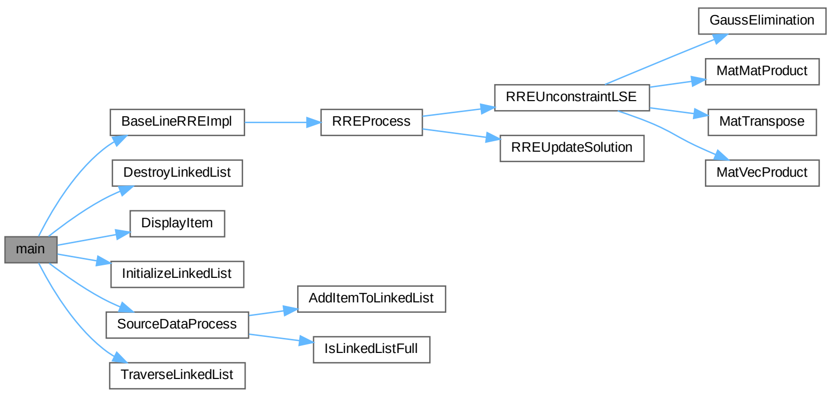
Parameters

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

Returns

int

Here is the call graph for this function:

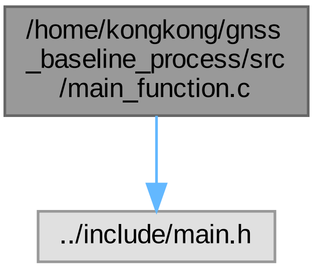


2.4 /home/kongkong/gnss_baseline_process/src/main_function.c File Reference

post position data process

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

Include dependency graph for main_function.c:



Functions

- void SourceDataProcess (char *path_file, double valid_ratio, double *base_station, List_LinkedList *rover↩_station_data)

    source data file process, e.g. assign values to base station coordinate, assign values to linked list

2.4.1 Detailed Description

post position data process

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.4.2 Function Documentation

2.4.2.1 SourceDataProcess()

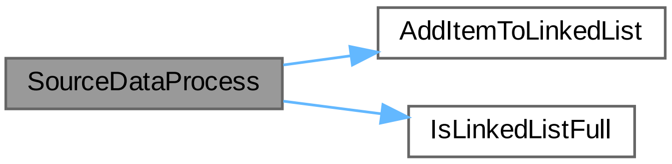
```
void SourceDataProcess (
            char * path_file,
            double valid_ratio,
            double * base_station,
            List_LinkedList * rover_station_data )
```

source data file process, e.g. assign values to base station coordinate, assign values to linked list

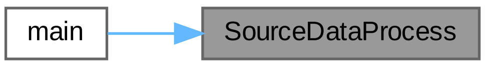
Parameters

`in`	*path_file*	path of source data file
`in`	*valid_ratio*	ratio threshold
`in,out`	*base_station*	coordinate of base station
`in,out`	*rover_station_data*	linked list to store valid data

Here is the call graph for this function:



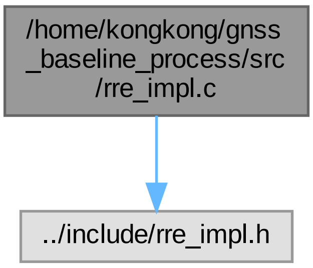
Here is the caller graph for this function:



2.5 /home/kongkong/gnss_baseline_process/src/rre_impl.c File Reference

reduced rank extrapolation implementation functions

`#include "../include/rre_impl.h"`
Include dependency graph for rre_impl.c:



Functions

- void BaseLineRREImpl (const List_LinkedList ∗pList, const double ∗base_station, double ∗solution)
    baseline process RRE main implementation
- void RREProcess (double ∗∗vec_seq, int size_row, int size_column, double ∗trans_vec_seq)
    rre process
- void RREUpdateSolution (double ∗∗mat_1, double ∗∗mat_2, double ∗gamma, double ∗solution, int m, int n)
    fusing valid data with linear combination coefficients to update solution
- void RREUnconstraintLSE (double ∗∗delta_mat_u, double ∗∗mat_u, int size_row, int size_column, double ∗gamma)
    assemble unconstraint least-squares equation

2.5.1 Detailed Description

reduced rank extrapolation implementation functions

Author

Zikang Qin

Version

0.1

Date

2023-06-21

Copyright

Copyright (c) 2023

2.5.2 Function Documentation

2.5.2.1 BaseLineRREImpl()

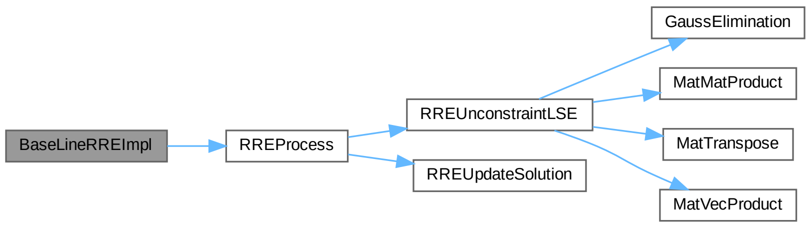
```
void BaseLineRREImpl (
            const List_LinkedList * pList,
            const double * base_station,
            double * solution )
```

baseline process RRE main implementation

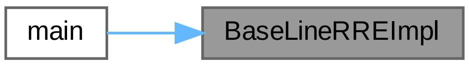
Parameters

in	*pList*	linked list
in	*base_station*	coordinate of base station
in,out	*solution*	solution to rre

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.2.2 RREProcess()

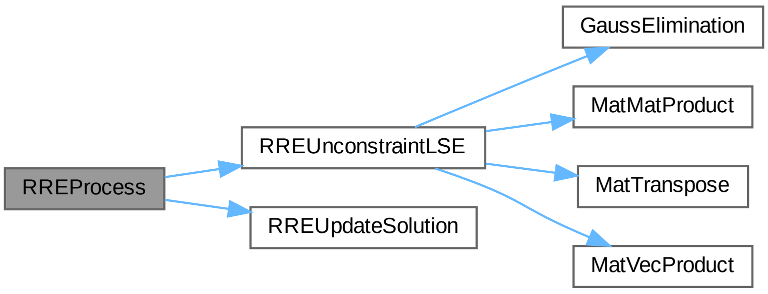
```
void RREProcess (
            double ** vec_seq,
            int size_row,
            int size_column,
            double * trans_vec_seq )
```

rre process

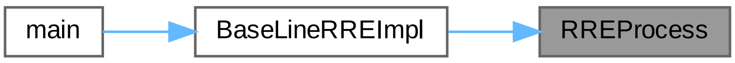
Parameters

`in`	*vec_seq*	original vector sequence
`in`	*size_row*	row size of vector sequence
`in`	*size_column*	column size of vector sequence
`in,out`	*trans_vec_seq*	solution to rre

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.2.3 RREUnconstraintLSE()

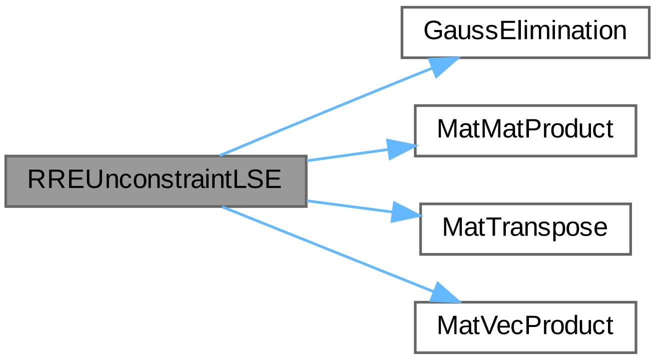
```
void RREUnconstraintLSE (
              double ** delta_mat_u,
              double ** mat_u,
              int size_row,
              int size_column,
              double * gamma )
```

assemble unconstraint least-squares equation

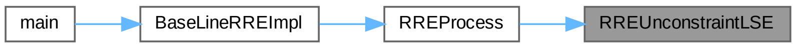
Parameters

`in`	*delta_mat←_u*	difference of mat_u
`in`	*mat_u*	difference of original vector sequence
`in`	*size_row*	row size of delta_mat_u
`in`	*size_column*	column size of delta_mat_u
`in,out`	*gamma*	linear combination coefficients of rre

Here is the call graph for this function:



Here is the caller graph for this function:



2.5.2.4 RREUpdateSolution()

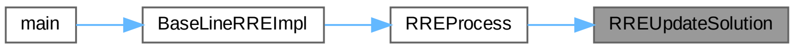
```
void RREUpdateSolution (
             double ** mat_1,
             double ** mat_2,
             double * gamma,
             double * solution,
             int m,
             int n )
```

fusing valid data with linear combination coefficients to update solution

Parameters

in	*mat_1*	original vector sequence
in	*mat_2*	difference of original vector sequence
in	*gamma*	linear combination coefficients
in,out	*solution*	solution to rre
in	*m*	row size of vector sequence
in	*n*	column size of vector sequence

Here is the caller graph for this function:



Index