My Project

2.0

Generated by Doxygen 1.13.2

Chapter 1

Bug List

File Accel.cpp

No known bugs

File Accel.h

No known bugs

File AccelHarmonic.cpp

No known bugs

File AccelHarmonic.h

No known bugs

File AccelPointMass.cpp

No known bugs

File AccelPointMass.h

No known bugs

File AzElPa.cpp

No known bugs

File AzElPa.h

No known bugs

File Cheb3D.cpp

No known bugs

File Cheb3D.h

No known bugs

File DEInteg.cpp

No known bugs

File DEInteg.h

No known bugs

File EccAnom.cpp

No known bugs

File EccAnom.h

No known bugs

File EKF_GEOS3.cpp

No known bugs

File EqnEquinox.cpp

No known bugs

File EqnEquinox.h

No known bugs

File Frac.cpp

No known bugs

File Frac.h

No known bugs

File G_AccelHarmonic.cpp

No known bugs

File G_AccelHarmonic.h

No known bugs

File gast.cpp

No known bugs

File gast.h

No known bugs

File GHAMatrix.cpp

No known bugs

File GHAMatrix.h

No known bugs

File GLOBAL.cpp

No known bugs

File GLOBAL.h

No known bugs

File gmst.cpp

No known bugs

File gmst.h

No known bugs

File IERS.cpp

No known bugs

File IERS.h

No known bugs

File JPL_Eph_DE430.cpp

Noknownbugs

File JPL_Eph_DE430.h

No known bugs

File Legendre.cpp

No known bugs

File Legendre.h

No known bugs

File LTC.cpp

No known bugs

File LTC.h

No known bugs

File matrix.cpp

No known bugs

File matrix.h

No known bugs

File MeanObliquity.cpp

No known bugs

File MeanObliquity.h

No known bugs

File MeasUpdate.cpp

No known bugs

File MeasUpdate.h

No known bugs

File Mjday.cpp

No known bugs

File Mjday.h

No known bugs

File Mjday_TDB.cpp

No known bugs

File Mjday_TDB.h

No known bugs

File NutAngles.cpp

No known bugs

File NutAngles.h

No known bugs

File NutMatrix.cpp

No known bugs

File NutMatrix.h

No known bugs

File PoleMatrix.cpp

No known bugs

File PoleMatrix.h

No known bugs

File Position.cpp

No known bugs

File Position.h

No known bugs

File PrecMatrix.cpp

No known bugs

File PrecMatrix.h

No known bugs

File R_x.cpp

No known bugs

File R_x.h

No known bugs

File R_y.cpp

No known bugs

File R_y.h

No known bugs

File R_z.cpp

No known bugs

File R_z.h

No known bugs

File SAT_Const.h

No known bugs

File sign_.cpp

No known bugs

File sign_.h

No known bugs

File timediff.cpp

No known bugs

File timediff.h

No known bugs

File TimeUpdate.cpp

No known bugs

File TimeUpdate.h

No known bugs

File VarEqn.cpp

No known bugs

File VarEqn.h

No known bugs

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Matrix Class Reference

Public Member Functions

- Matrix ()
- Matrix (const int n)
- Matrix (const int n_row, const int n_column)
- double & operator() (const int n)
- double & operator() (const int row, const int column)
- Matrix & operator+ (Matrix &m)
- Matrix & operator- (Matrix &m)
- Matrix & operator* (Matrix &m)
- Matrix & operator/ (Matrix &m)
- Matrix & operator= (Matrix &m)
- Matrix & operator+ (double d)
- Matrix & operator- (double d)
- Matrix & operator* (double d)
- Matrix & operator/ (double d)

Public Attributes

- int n_row
- int n_column
- double ** data

Friends

- ostream & operator<< (ostream &o, Matrix &m)

4.1.1 Detailed Description

Definition at line 18 of file matrix.h.

4.1.2 Constructor & Destructor Documentation

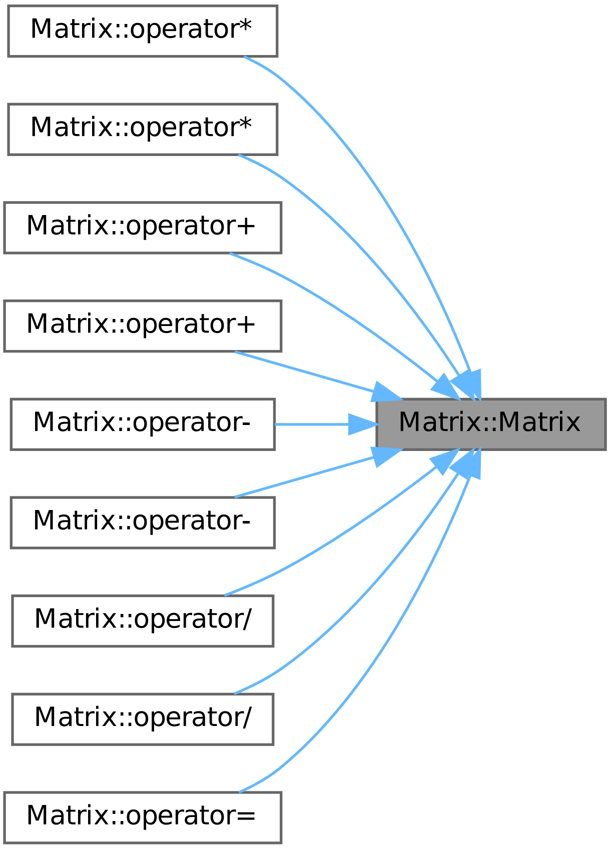
4.1.2.1 Matrix() [1/3]

```
Matrix::Matrix ()
```

Crea un objeto Matrix vacio

Definition at line 10 of file matrix.cpp.

Here is the caller graph for this function:



4.1.2.2 Matrix() [2/3]

```
Matrix::Matrix (
            const int n)
```

Crea una matriz [0][n] que simula un vector

Parameters

n	número de columas del vector

Definition at line 17 of file matrix.cpp.

4.1.2.3 Matrix() [3/3]

```
Matrix::Matrix (
             const int n_row,
             const int n_column)
```

Crea una matriz [n_row][n_column]

Parameters

n_row	número de filas de la matriz
n_column	número de columas de la matriz

Definition at line 35 of file matrix.cpp.

4.1.3 Member Function Documentation

4.1.3.1 operator()() [1/2]

```
double & Matrix::operator() (
             const int n)
```

Obtiene el elemento [(n-1)/n_column][(n-1)n_column]

Parameters

n	elemnto

Definition at line 55 of file matrix.cpp.

4.1.3.2 operator()() [2/2]

```
double & Matrix::operator() (
             const int row,
             const int column)
```

Obtiene el elemento [(row-1)][(column-1)]

Parameters

n_row	fila de la matriz
n_column	columa de la matriz

Definition at line 64 of file matrix.cpp.

4.1.3.3 operator*() [1/2]

```
Matrix & Matrix::operator* (
             double d)
```

Multiplica todas las componentes de this con d y devuelve una nueva Matrix

Parameters

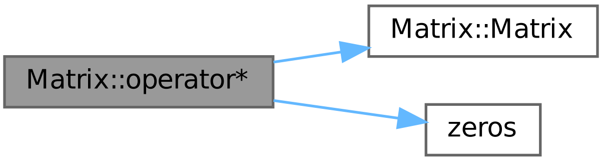
d	valor a ser operado por cada componente de la matriz

Returns

una Matrix donde todas sus componentes se le multiplica d, sin modificar las matrices

Definition at line 192 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.4 operator∗() [2/2]

```
Matrix & Matrix::operator* (
            Matrix & m)
```

Multiplación sobre matrices para 2(this∗m) Matrix y devuelve el valor

Parameters

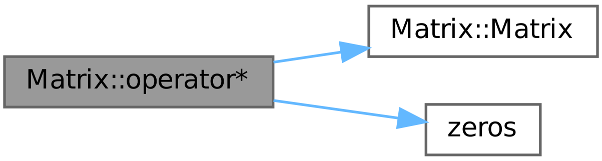
m	Matrix

Returns

devuelve un Matrix = this∗m, sin modificarlos

Definition at line 108 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.5 operator+() [1/2]

```
Matrix & Matrix::operator+ (
             double d)
```

Suma todas las componentes de this con d y devuelve una nueva Matrix

Parameters

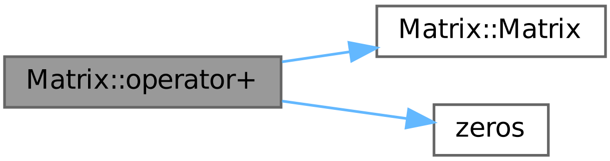
d	valor a ser operado por cada componente de la matriz

Returns

una Matrix donde todas sus componentes se le suma d, sin modificar las matrices

Definition at line 168 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.6 operator+() [2/2]

```
Matrix & Matrix::operator+ (
             Matrix & m)
```

Suma 2(this+m) Matrix y devuelve el valor

Parameters

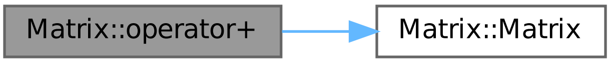
m	Matrix

Returns

devuelve un Matrix suma de this+ m, sin modificarlos

Definition at line 74 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.7 operator-() [1/2]

```
Matrix & Matrix::operator- (
            double d)
```

Resta todas las componentes de this con d y devuelve una nueva Matrix

Parameters

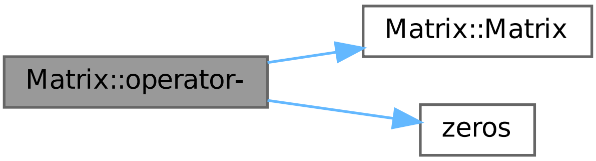
d	valor a ser operado por cada componente de la matriz

Returns

una Matrix donde todas sus componentes se le resta d, sin modificar las matrices

Definition at line 180 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.8 operator-() [2/2]

```
Matrix & Matrix::operator- (
            Matrix & m)
```

Resta 2(this-m) Matrix y devuelve el valor

Parameters

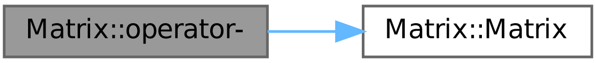
m	Matrix

Returns

devuelve un Matrix resta de this-m, sin modificarlos

Definition at line 91 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.9 operator/() [1/2]

```
Matrix & Matrix::operator/ (
             double d)
```

Divide todas las componentes de this con d y devuelve una nueva Matrix

Parameters

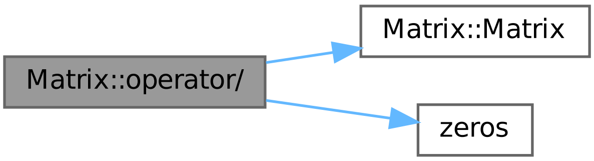
d	valor a ser operado por cada componente de la matriz

Returns

una Matrix donde todas sus componentes se le divide d, sin modificar las matrices

Definition at line 202 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.10 operator/() [2/2]

```
Matrix & Matrix::operator/ (
             Matrix & m)
```

this∗(m^(-1)) y devuelve el valor

Parameters

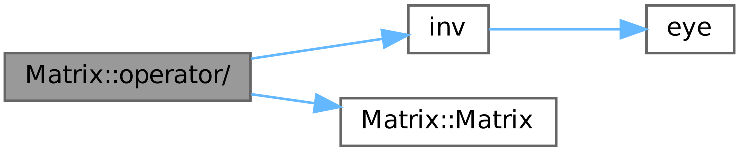
m	Matrix

Returns

devuelve un Matrix =this-∗m^(-1), sin modificarlos

Definition at line 127 of file matrix.cpp.

Here is the call graph for this function:



4.1.3.11 operator=()

```
Matrix & Matrix::operator= (
            Matrix & m)
```

Se crea una Matrix equivalente a m y se le asigna a this

Parameters

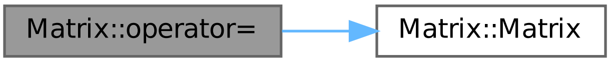
m	Matrix

Returns

una Matrix=m, sin modificar las matrices

Definition at line 138 of file matrix.cpp.

Here is the call graph for this function:



4.1.4 Friends And Related Symbol Documentation

4.1.4.1 operator<<

```
ostream & operator<< (
             ostream & o,
             Matrix & m)  [friend]
```

Definition at line 212 of file matrix.cpp.

4.1.5 Member Data Documentation

4.1.5.1 data

```
double** Matrix::data
```

Definition at line 21 of file matrix.h.

4.1.5.2 n_column

```
int Matrix::n_column
```

Definition at line 20 of file matrix.h.

4.1.5.3 n_row

```
int Matrix::n_row
```

Definition at line 20 of file matrix.h.

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

- matrix.h
- matrix.cpp

4.2 Param Struct Reference

Public Attributes

- double Mjd_UTC
- double Mjd_TT
- int n
- int m
- int sun
- int moon
- int planets

4.2.1 Detailed Description

Definition at line 13 of file GLOBAL.h.

4.2.2 Member Data Documentation

4.2.2.1 m

```
int Param::m
```

Definition at line 15 of file GLOBAL.h.

4.2.2.2 Mjd_TT

```
double Param::Mjd_TT
```

Definition at line 14 of file GLOBAL.h.

4.2.2.3 Mjd_UTC

```
double Param::Mjd_UTC
```

Definition at line 14 of file GLOBAL.h.

4.2.2.4 moon

```
int Param::moon
```

Definition at line 15 of file GLOBAL.h.

4.2.2.5 n

```
int Param::n
```

Definition at line 15 of file GLOBAL.h.

4.2.2.6 planets

```
int Param::planets
```

Definition at line 15 of file GLOBAL.h.

4.2.2.7 sun

```
int Param::sun
```

Definition at line 15 of file GLOBAL.h.

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

- GLOBAL.h

Chapter 5

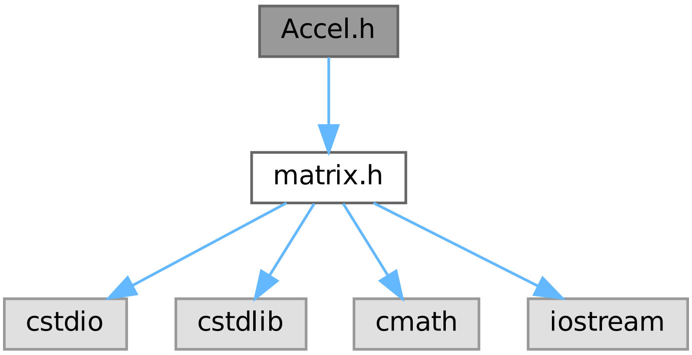
File Documentation

5.1 Accel.h File Reference

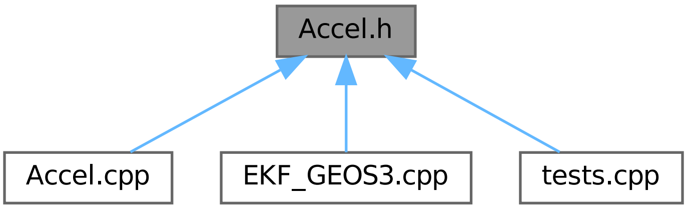
El archivo contiene la funcion Accel.

```
#include "matrix.h"
```

Include dependency graph for Accel.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & Accel (double x, Matrix Y)

5.1.1 Detailed Description

El archivo contiene la funcion Accel.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Accel.h.

5.1.2 Function Documentation

5.1.2.1 Accel()

```
Matrix & Accel (
             double x,
             Matrix Y)
```

Computes the acceleration of an Earth orbiting satellite due to

- the Earth's harmonic gravity field,
- the gravitational perturbations of the Sun and Moon
- the solar radiation pressure and
- the atmospheric drag

Parameters

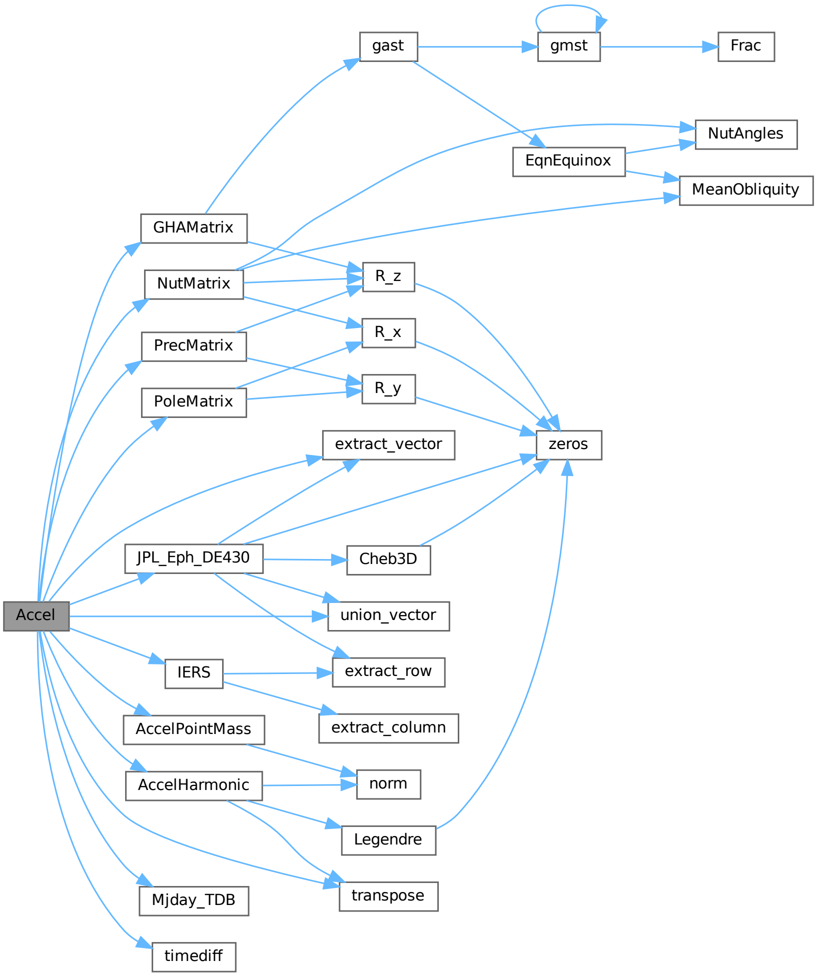
Mjd_TT	Terrestrial Time (Modified Julian Date)
Y	Satellite state vector in the ICRF/EME2000 system

Returns

dY Acceleration (a=d^2r/dt^2) in the ICRF/EME2000 system

Definition at line 20 of file Accel.cpp.

Here is the call graph for this function:



5.2 Accel.h

Go to the documentation of this file.

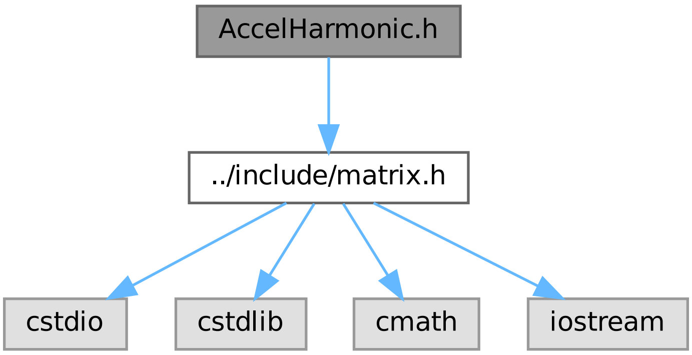
```
00001 #ifndef _Accel_
00002 #define _Accel_
00003 #include "matrix.h"
00004
00005 using namespace std;
00006
00023     Matrix& Accel(double x,Matrix Y);
00024 #endif
00025
00026
```

5.3 AccelHarmonic.h File Reference

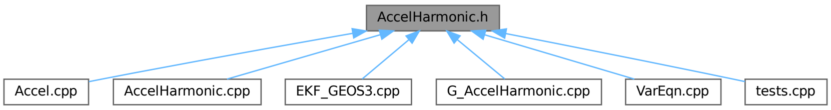
El archivo contiene la funcion AccelHarmonic.

```
#include "../include/matrix.h"
```

Include dependency graph for AccelHarmonic.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & AccelHarmonic (Matrix r, Matrix E, int n_max, int m_max)

5.3.1 Detailed Description

El archivo contiene la funcion AccelHarmonic.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AccelHarmonic.h.

5.3.2 Function Documentation

5.3.2.1 AccelHarmonic()

```
Matrix & AccelHarmonic (
            Matrix r,
            Matrix E,
            int n_max,
            int m_max)
```

Parameters

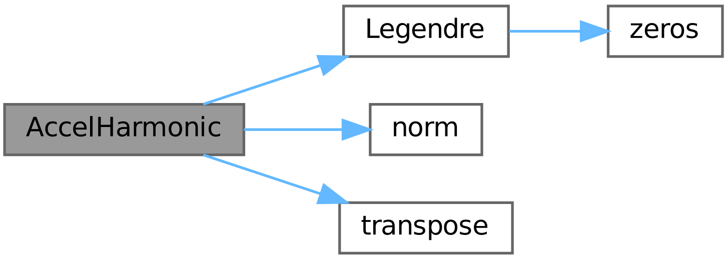
r	Satellite position vector in the inertial system
E	Transformation matrix to body-fixed system
n_max	Maximum degree
m_max	Maximum order (m_max<=n_max; m_max=0 for zonals, only)

Returns

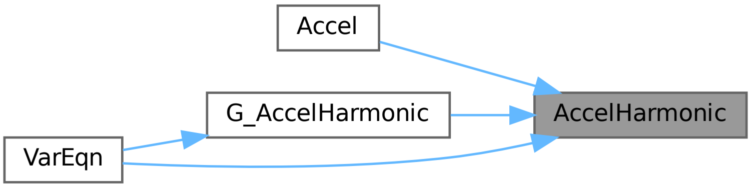
a Acceleration (a=d^2r/dt^2)

Definition at line 12 of file AccelHarmonic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4 AccelHarmonic.h

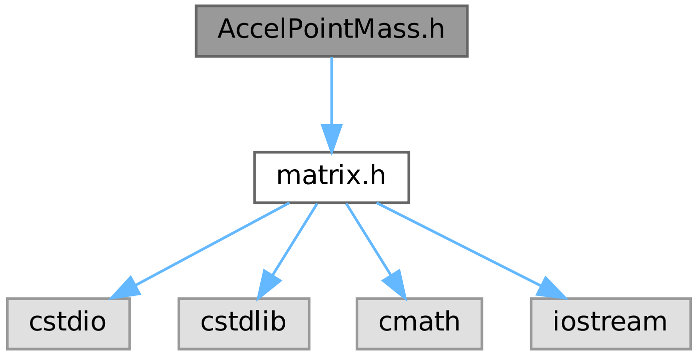
Go to the documentation of this file.

```
#ifndef _AccelHarmonic_
#define _AccelHarmonic_
using namespace std;
#include "../include/matrix.h"



    Matrix& AccelHarmonic(Matrix r,Matrix E,int n_max,int m_max);
#endif


```

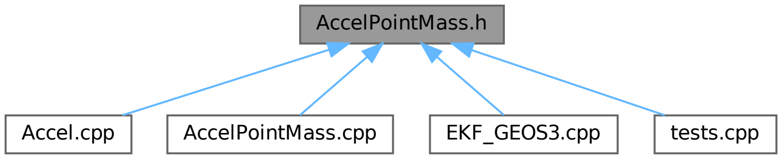
5.5 AccelPointMass.h File Reference

El archivo contiene la funcion AccelPointMass.

`#include "matrix.h"`
Include dependency graph for AccelPointMass.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & AccelPointMass (Matrix &r, Matrix &s, double GM)

5.5.1 Detailed Description

El archivo contiene la funcion AccelPointMass.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AccelPointMass.h.

5.5.2 Function Documentation

5.5.2.1 AccelPointMass()

```
Matrix & AccelPointMass (
             Matrix & r,
             Matrix & s,
             double GM)
```

Parameters

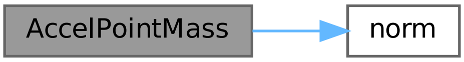
r	Satellite position vector
s	Point mass position vector
GM	Gravitational coefficient of point mass

Returns

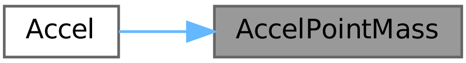
Acceleration (a=d^2r/dt^2)

Definition at line 10 of file AccelPointMass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6 AccelPointMass.h

Go to the documentation of this file.

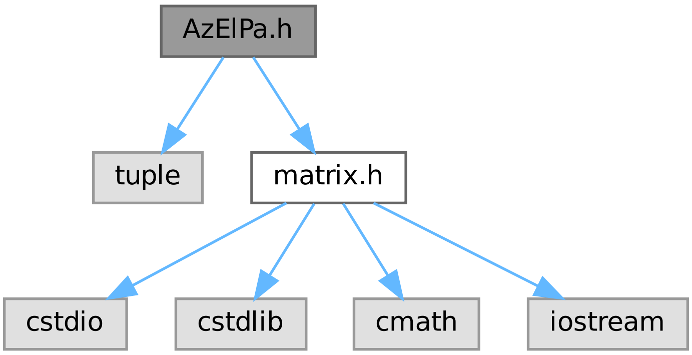
```
#ifndef _AccelPointMass_
#define _AccelPointMass_
#include "matrix.h"

using namespace std;

    Matrix&   AccelPointMass(Matrix& r,Matrix& s,double GM);
#endif


```

5.7 AzElPa.h File Reference

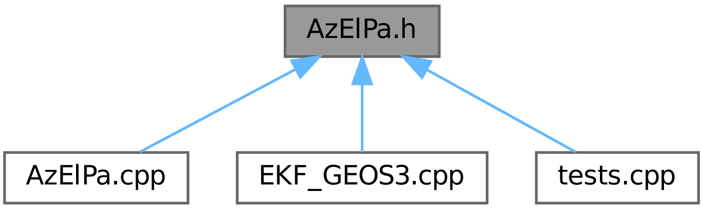
El archivo contiene la funcion AzElPa.

```
#include <tuple>
#include "matrix.h"
```

Include dependency graph for AzElPa.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< double, double, Matrix &, Matrix & > AzElPa (Matrix s)

5.7.1 Detailed Description

El archivo contiene la funcion AzElPa.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AzElPa.h.

5.7.2 Function Documentation

5.7.2.1 AzElPa()

```
tuple< double, double, Matrix &, Matrix & > AzElPa (
            Matrix s)
```

Computes azimuth, elevation and partials from local tangent coordinates s

Parameters

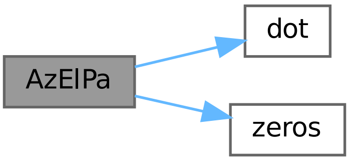
s	Matrix 1x3 Topocentric local tangent coordinates (East-North-Zenith frame)

Returns

tuple<A,E,dAds,dEds> Azimuth [rad],Elevation [rad],Partials of azimuth w.r.t. s,Partials of elevation w.r.t. s

Definition at line 9 of file AzElPa.cpp.

Here is the call graph for this function:



5.8 AzElPa.h

Go to the documentation of this file.

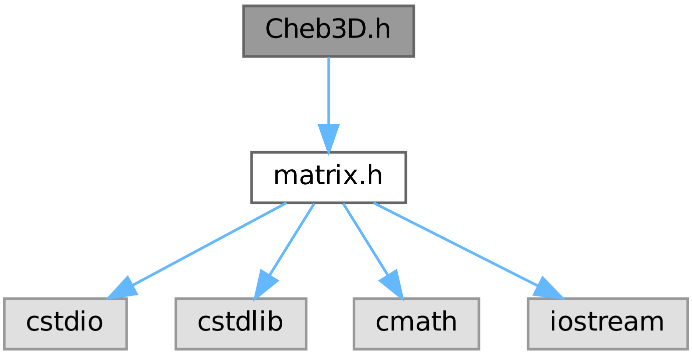
```
#ifndef _AzElPa_
#define _AzElPa_
#include <tuple>
#include "matrix.h"
using namespace std;

    tuple<double,double,Matrix&,Matrix&> AzElPa(Matrix s);
#endif


```

5.9 Cheb3D.h File Reference

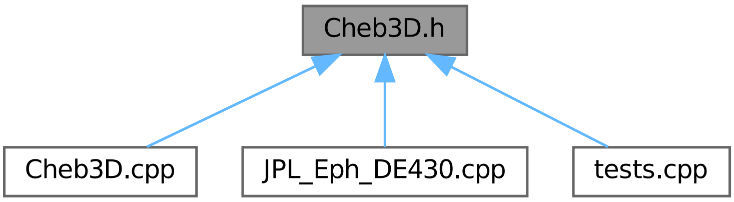
El archivo contiene la funcion Cheb3D.

`#include "matrix.h"`

Include dependency graph for Cheb3D.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & Cheb3D (double t, int N, double Ta, double Tb, Matrix &Cx, Matrix &Cy, Matrix &Cz)

5.9.1 Detailed Description

El archivo contiene la funcion Cheb3D.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Cheb3D.h.

5.9.2 Function Documentation

5.9.2.1 Cheb3D()

```
Matrix & Cheb3D (
            double t,
            int N,
            double Ta,
            double Tb,
            Matrix & Cx,
            Matrix & Cy,
            Matrix & Cz)
```

Parameters

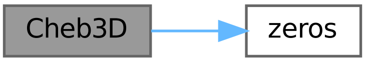
t	time
N	Number of coefficients
Ta	Begin interval
Tb	End interval
Cx	Coefficients of Chebyshev polyomial (x-coordinate)
Cy	Coefficients of Chebyshev polyomial (y-coordinate)
Cz	Coefficients of Chebyshev polyomial (z-coordinate)

Returns

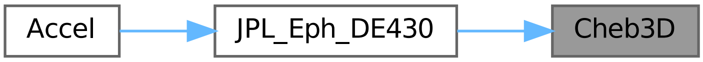
Chebyshev approximation of 3-dimensional vectors

Definition at line 9 of file Cheb3D.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10 Cheb3D.h

Go to the documentation of this file.

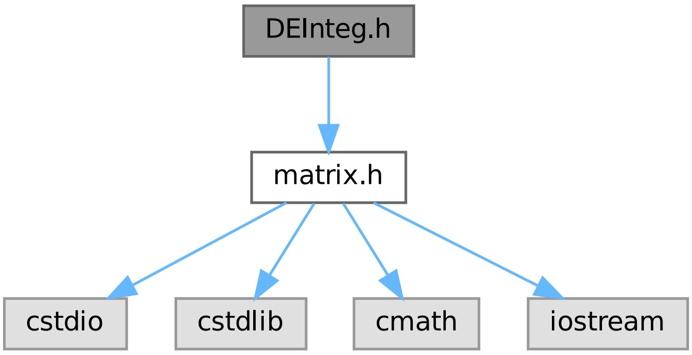
```
#ifndef _Cheb3D_
#define _Cheb3D_
#include "matrix.h"

using namespace std;

    Matrix&   Cheb3D( double t, int N, double Ta, double Tb, Matrix& Cx,Matrix& Cy,Matrix& Cz);
#endif


```

5.11 DEInteg.h File Reference

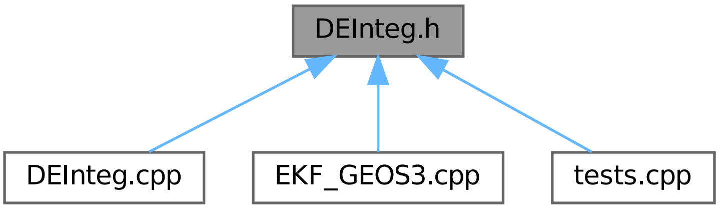
El archivo contiene la funcion DEInteg.

`#include "matrix.h"`

Include dependency graph for DEInteg.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & DEInteg (Matrix &f(double t, Matrix y), double t, double tout, double relerr, double abserr, int n_eqn, Matrix &y)

5.11.1 Detailed Description

El archivo contiene la funcion DEInteg.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file DEInteg.h.

5.11.2 Function Documentation

5.11.2.1 DEInteg()

```
Matrix & DEInteg (
            Matrix & fdouble t, Matrix y,
            double t,
            double tout,
            double relerr,
            double abserr,
            int n_eqn,
            Matrix & y)
```

Numerical integration methods for ordinaray differential equations This module provides implemenation of the variable order variable stepsize multistep method of Shampine & Gordon.

Parameters

f	funcion pasas double y Matrix devuelve Matrix
t	double
tout	double
relerr	double
abserr	double
n_eqn	int
y	Matrix

Returns

Matrix resultado

5.12 DEInteg.h

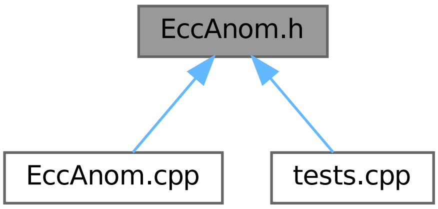
Go to the documentation of this file.

```
#ifndef _DEInteg_
#define _DEInteg_
#include "matrix.h"

using namespace std;

     Matrix& DEInteg(Matrix& f(double t,Matrix y),double t, double tout,double relerr,double abserr,int n_eqn,Matrix &y);
#endif


```

5.13 EccAnom.h File Reference

El archivo contiene la funcion EccAnom.

This graph shows which files directly or indirectly include this file:



Functions

- double EccAnom (double M, double e)

5.13.1 Detailed Description

El archivo contiene la funcion EccAnom.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file EccAnom.h.

5.13.2 Function Documentation

5.13.2.1 EccAnom()

```
double EccAnom (
             double M,
             double e)
```

Computes the eccentric anomaly for elliptic orbits

Parameters

M	Mean anomaly in [rad]
e	Eccentricity of the orbit [0,1]

Returns

double resultado

Definition at line 12 of file EccAnom.cpp.

5.14 EccAnom.h

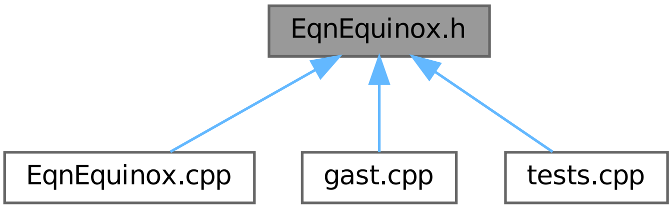
Go to the documentation of this file.

```
#ifndef _EccAnom_
#define _EccAnom_

using namespace std;

    double EccAnom (double M,double e);
#endif


```

5.15 EqnEquinox.h File Reference

El archivo contiene la funcion EqnEquinox.

This graph shows which files directly or indirectly include this file:



Functions

- double EqnEquinox (double Mjd_TT)

5.15.1 Detailed Description

El archivo contiene la funcion EqnEquinox.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file EqnEquinox.h.

5.15.2 Function Documentation

5.15.2.1 EqnEquinox()

```
double EqnEquinox (
              double Mjd_TT)
```

Computation of the equation of the equinoxes

Parameters

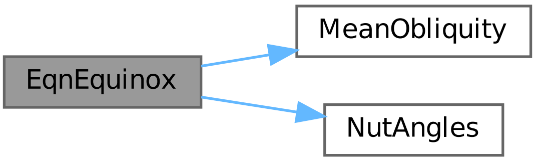
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

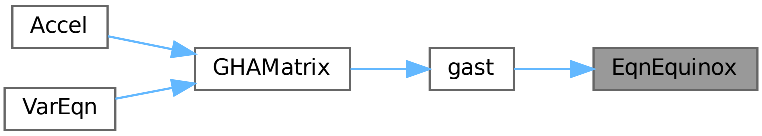
double Equation of the equinoxes

Definition at line 13 of file EqnEquinox.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.16 EqnEquinox.h

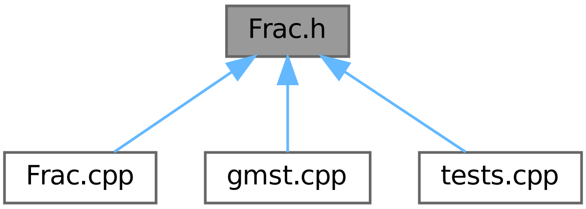
Go to the documentation of this file.

```
00001 #ifndef _EqnEquinox_
00002 #define _EqnEquinox_
00003 using namespace std;
00004
00016     double EqnEquinox (double Mjd_TT);
00017 #endif
00018
00019
```

5.17 Frac.h File Reference

El archivo contiene la funcion Frac.

This graph shows which files directly or indirectly include this file:



Functions

- double Frac (double x)

5.17.1 Detailed Description

El archivo contiene la funcion Frac.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Frac.h.

5.17.2 Function Documentation

5.17.2.1 Frac()

```
double Frac (
              double x)
```

Parameters

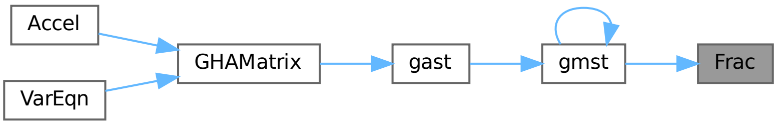
x	double

Returns

double parte fraccion de x

Definition at line 10 of file Frac.cpp.

Here is the caller graph for this function:



5.18 Frac.h

Go to the documentation of this file.

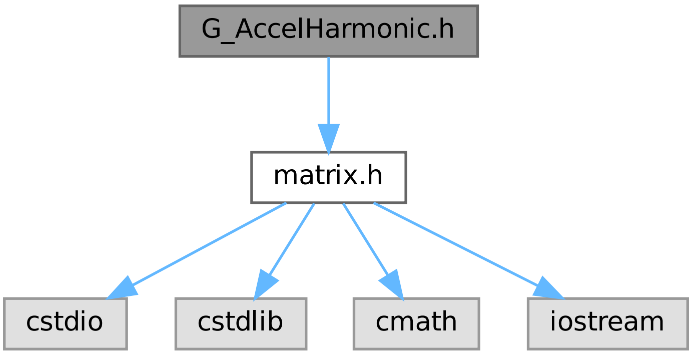
```
00001 #ifndef _Frac_
00002 #define _Frac_
00003
00004 using namespace std;
00005
00016     double Frac(double x);
00017 #endif
00018
00019
```

5.19 G_AccelHarmonic.h File Reference

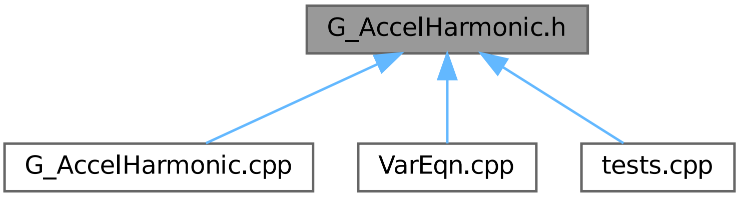
El archivo contiene la funcion G_AccelHarmonic.

```
#include "matrix.h"
```

Include dependency graph for G_AccelHarmonic.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & G_AccelHarmonic (Matrix r, Matrix U, int n_max, int m_max)

5.19.1 Detailed Description

El archivo contiene la funcion G_AccelHarmonic.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file G_AccelHarmonic.h.

5.19.2 Function Documentation

5.19.2.1 G_AccelHarmonic()

```
Matrix & G_AccelHarmonic (
             Matrix r,
             Matrix U,
             int n_max,
             int m_max)
```

Parameters

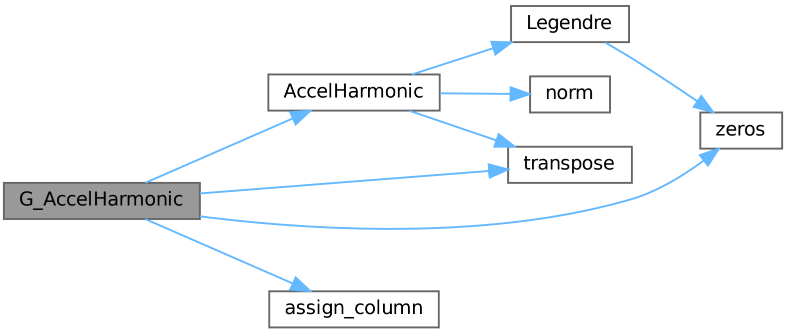
r	Satellite position vector in the true-of-date system
U	Transformation matrix to body-fixed syste
n	Gravity model degree
m	Gravity model order

Returns

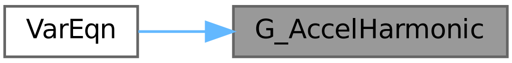
G Gradient (G=da/dr) in the true-of-date system

Definition at line 10 of file G_AccelHarmonic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20 G_AccelHarmonic.h

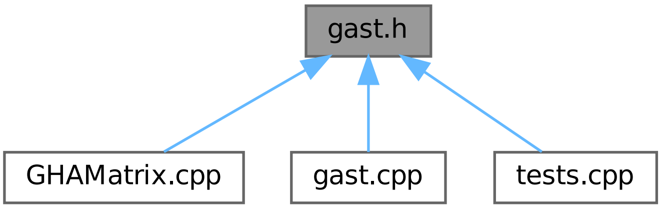
Go to the documentation of this file.

```
00001 #ifndef _G_AccelHarmonic_
00002 #define _G_AccelHarmonic_
00003 #include "matrix.h"
00004
00005 using namespace std;
00006
00020     Matrix& G_AccelHarmonic( Matrix r,Matrix U, int n_max, int m_max );
00021 #endif
00022
00023
```

5.21 gast.h File Reference

El archivo contiene la funcion gast.

This graph shows which files directly or indirectly include this file:



Functions

- double gast (double Mjd_UT1)

5.21.1 Detailed Description

El archivo contiene la funcion gast.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file gast.h.

5.21.2 Function Documentation

5.21.2.1 gast()

```
double gast (
            double Mjd_UT1)
```

Greenwich Apparent Sidereal Time

Parameters

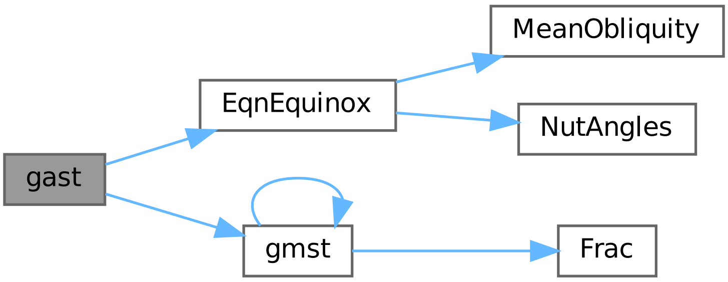
Mjd_UT1	Modified Julian Date UT1

Returns

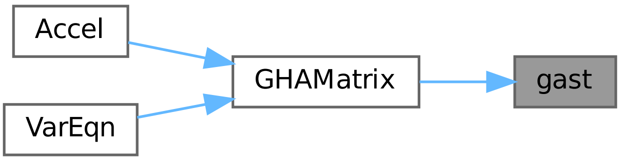
gstime GAST in [rad]

Definition at line 14 of file gast.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.22 gast.h

Go to the documentation of this file.

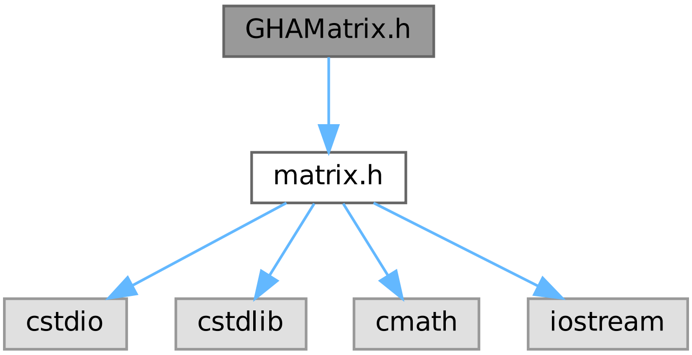
```
#ifndef _gast_
#define _gast_

using namespace std;

    double gast(double Mjd_UT1);
#endif


```

5.23 GHAMatrix.h File Reference

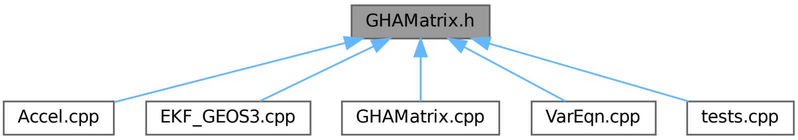
El archivo contiene la funcion GHAMatrix.

```
#include "matrix.h"
```

Include dependency graph for GHAMatrix.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & GHAMatrix (double Mjd_UT1)

5.23.1 Detailed Description

El archivo contiene la funcion GHAMatrix.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file GHAMatrix.h.

5.23.2 Function Documentation

5.23.2.1 GHAMatrix()

```
Matrix & GHAMatrix (
            double Mjd_UT1)
```

Transformation from true equator and equinox to Earth equator and Greenwich meridian system

Parameters

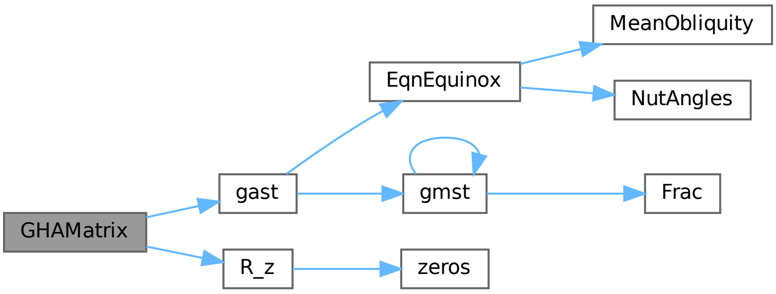
Mjd_UT1	Modified Julian Date UT1

Returns

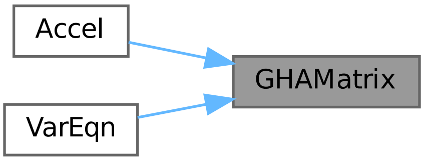
GHAmat Greenwich Hour Angle matrix

Definition at line 11 of file GHAMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.24 GHAMatrix.h

Go to the documentation of this file.

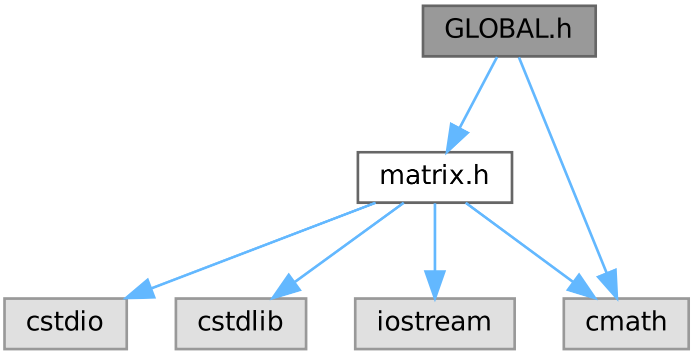
```
#ifndef _GHAMatrix_
#define _GHAMatrix_
#include "matrix.h"

using namespace std;

    Matrix& GHAMatrix (double Mjd_UT1);
#endif


```

5.25 GLOBAL.h File Reference

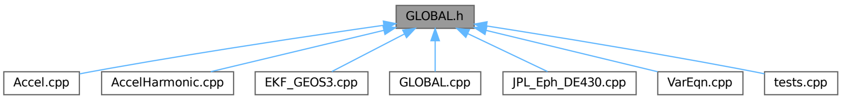
El archivo contiene la funcion GLOBAL.

```
#include "matrix.h"
#include <cmath>
```

Include dependency graph for GLOBAL.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Param

Functions

- void eop19620101 (int c=21413)
- void GGM03S (int n=181)
- void DE430Coeff (int row=2285, int column=1020)
- void AuxParamLoad ()
- void GEOS3 (int nobs=46)

Variables

- Param AuxParam
- Matrix eopdata
- Matrix Cnm
- Matrix Snm
- Matrix PC
- Matrix obs

5.25.1 Detailed Description

El archivo contiene la funcion GLOBAL.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file GLOBAL.h.

5.25.2 Function Documentation

5.25.2.1 AuxParamLoad()

```
void AuxParamLoad ()
```

Carga AuxParam

Definition at line 17 of file GLOBAL.cpp.

5.25.2.2 DE430Coeff()

```
void DE430Coeff (
             int row = 2285,
             int column = 1020)
```

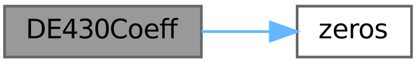
Lee el archivo DE430Coeff.txt y recoge cada fila y lo asigna a PC

Parameters

row	número de filas a recoger
column	número de columnas a recoger

Definition at line 68 of file GLOBAL.cpp.

Here is the call graph for this function:



5.25.2.3 eop19620101()

```
void eop19620101 (
            int c = 21413)
```

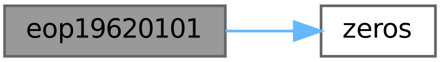
Lee el archivo eop19620101.txt y recoge cada fila y lo asigna a eopdata

Parameters

c	número de filas a recoger

Definition at line 26 of file GLOBAL.cpp.

Here is the call graph for this function:



5.25.2.4 GEOS3()

```
void GEOS3 (
            int nobs = 46)
```

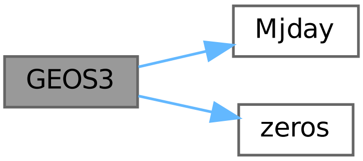
Lee el archivo GEOS3.txt y recoge cada fila y lo asigna a obs

Parameters

nobs	número de filas a recoger

Definition at line 87 of file GLOBAL.cpp.

Here is the call graph for this function:



5.25.2.5 GGM03S()

```
void GGM03S (
             int n = 181)
```

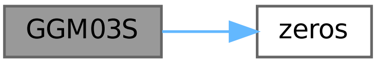
Lee el archivo GGM03S.txt y recoge cada fila y lo asigna a Cnm y Snm

Parameters

c	dimension de la matriz

Definition at line 47 of file GLOBAL.cpp.

Here is the call graph for this function:



5.25.3 Variable Documentation

5.25.3.1 AuxParam

```
Param AuxParam  [extern]
```

Definition at line 11 of file GLOBAL.cpp.

5.25.3.2 Cnm

```
Matrix Cnm  [extern]
```

Definition at line 13 of file GLOBAL.cpp.

5.25.3.3 eopdata

```
Matrix eopdata  [extern]
```

Definition at line 12 of file GLOBAL.cpp.

5.25.3.4 obs

```
Matrix obs  [extern]
```

Definition at line 16 of file GLOBAL.cpp.

5.25.3.5 PC

```
Matrix PC  [extern]
```

Definition at line 15 of file GLOBAL.cpp.

5.25.3.6 Snm

```
Matrix Snm  [extern]
```

Definition at line 14 of file GLOBAL.cpp.

5.26 GLOBAL.h

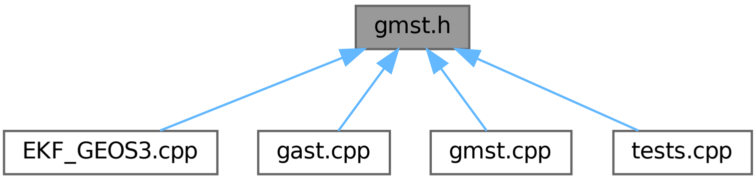
Go to the documentation of this file.

```

#ifndef _GLOBAL_
#define _GLOBAL_
using namespace std;
#include "matrix.h"
#include <cmath>

typedef struct{
    double Mjd_UTC,Mjd_TT;
    int n,m,sun,moon,planets;
} Param;

extern Param AuxParam;
extern Matrix eopdata;
extern Matrix Cnm;
extern Matrix Snm;
extern Matrix PC;
extern Matrix obs;

void eop19620101(int c=21413);

void GGM03S(int n=181);

void DE430Coeff(int row=2285,int column=1020);

void AuxParamLoad();

void GEOS3(int nobs=46);

#endif


```

5.27 gmst.h File Reference

El archivo contiene la funcion gmst.

This graph shows which files directly or indirectly include this file:



Functions

- double gmst (double Mjd_UT1)

5.27.1 Detailed Description

El archivo contiene la funcion gmst.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file gmst.h.

5.27.2 Function Documentation

5.27.2.1 gmst()

```
double gmst (
            double Mjd_UT1)
```

Greenwich Mean Sidereal Time

Parameters

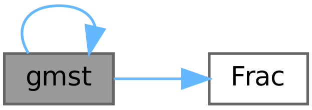
Mjd_UT1	Modified Julian Date UT1

Returns

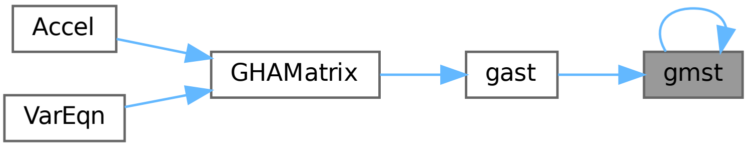
gmstime GMST in [rad]

Definition at line 12 of file gmst.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.28 gmst.h

Go to the documentation of this file.

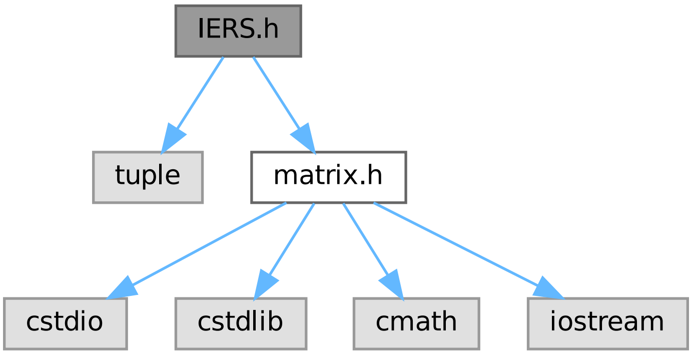
```
#ifndef _gmst_
#define _gmst_

using namespace std;

    double gmst(double Mjd_UT1);
#endif


```

5.29 IERS.h File Reference

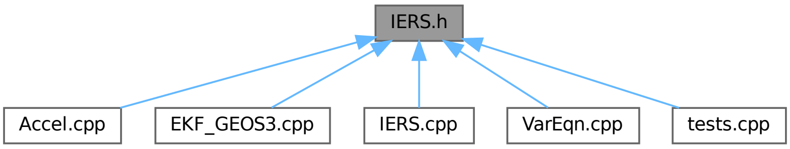
El archivo contiene la funcion IERS.

```
#include <tuple>
#include "matrix.h"
```

Include dependency graph for IERS.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< double, double, double, double, double, double, double, double, double > IERS (Matrix eop, double Mjd_UTC, char interp='n')

5.29.1 Detailed Description

El archivo contiene la funcion IERS.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file IERS.h.

5.29.2 Function Documentation

5.29.2.1 IERS()

```
tuple< double, double, double, double, double, double, double, double, double > IERS (
            Matrix eop,
            double Mjd_UTC,
            char interp = 'n')
```

IERS: Management of IERS time and polar motion data

Parameters

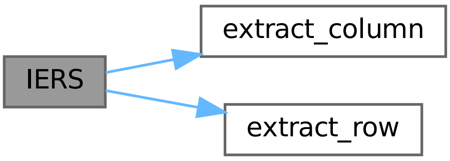
eop	matrix 4x13
Mjd_UTC	double
interp	char

Returns

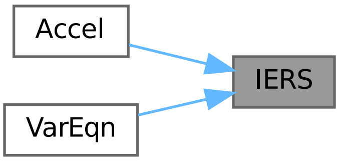
tupla <x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC>

Definition at line 10 of file IERS.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.30 IERS.h

Go to the documentation of this file.

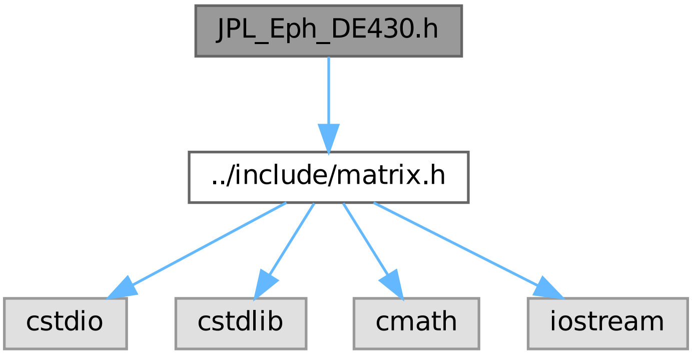
```
#ifndef _IERS_
#define _IERS_
#include <tuple>
#include "matrix.h"
using namespace std;

    tuple<double,double,double,double,double,double,double,double,double> IERS(Matrix eop,double
Mjd_UTC,char interp='n');
#endif


```

5.31 JPL_Eph_DE430.h File Reference

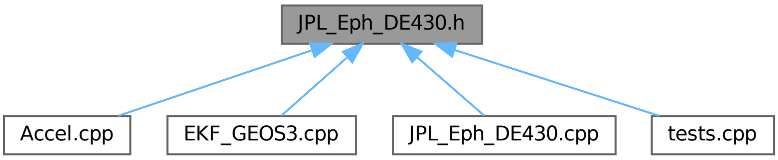
El archivo contiene la funcion JPL_Eph_DE430.

`#include "../include/matrix.h"`

Include dependency graph for JPL_Eph_DE430.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix & > JPL_Eph_DE430 (double Mjd_TDB)

5.31.1 Detailed Description

El archivo contiene la funcion JPL_Eph_DE430.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file JPL_Eph_DE430.h.

5.31.2 Function Documentation

5.31.2.1 JPL_Eph_DE430()

```
tuple< Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix
&, Matrix &, Matrix & > JPL_Eph_DE430 (
            double Mjd_TDB)
```

Parameters

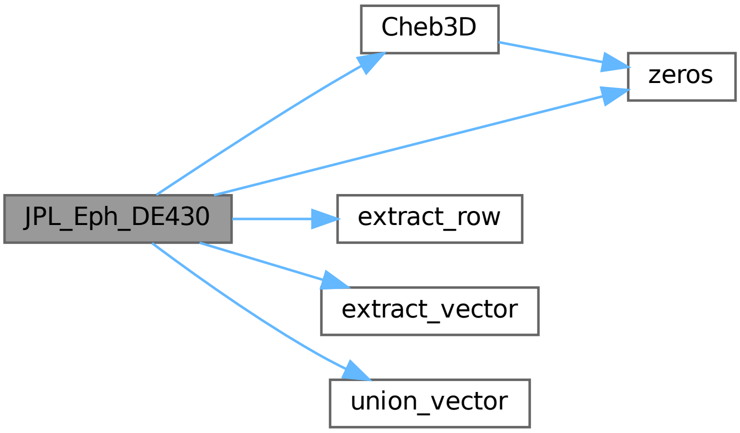
P	Matrix
Phi	Matrix
Qdt	Matrix

Returns

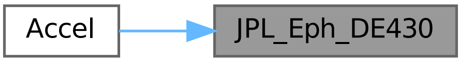
Matrix

Definition at line 12 of file JPL_Eph_DE430.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.32 JPL_Eph_DE430.h

Go to the documentation of this file.

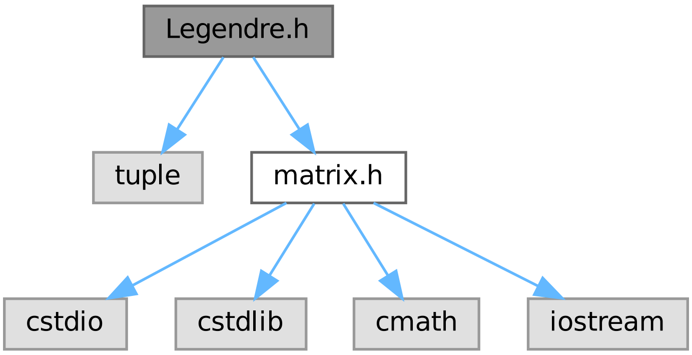
```
#ifndef _JPL_Eph_DE430_
#define _JPL_Eph_DE430_
using namespace std;
#include"../include/matrix.h"

    tuple<Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&> JPL_Eph_DE430(double Mjd_TDB);

#endif


```

5.33 Legendre.h File Reference

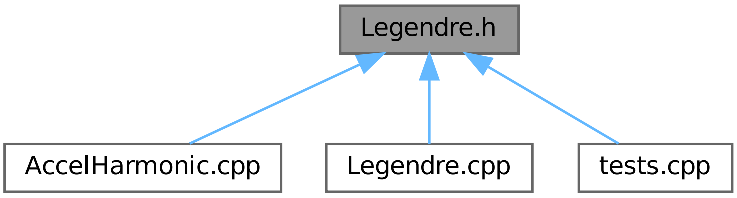
El archivo contiene la funcion Legendre.

```
#include <tuple>
#include "matrix.h"
```

Include dependency graph for Legendre.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< Matrix &, Matrix & > Legendre (int n, int m, double fi)

5.33.1 Detailed Description

El archivo contiene la funcion Legendre.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Legendre.h.

5.33.2 Function Documentation

5.33.2.1 Legendre()

```
tuple< Matrix &, Matrix & > Legendre (
                int n,
                int m,
                double fi)
```

Time differences [s]

Parameters

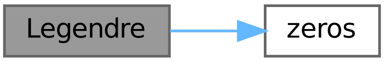
n	int
m	int
fi	double [rad]

Returns

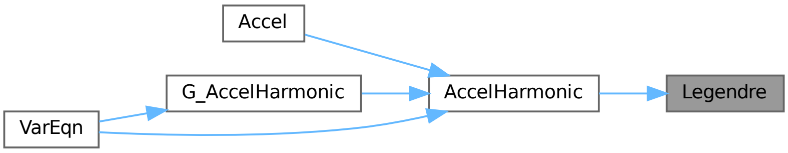
tupla <pnm,dpnm>

Definition at line 8 of file Legendre.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.34 Legendre.h

Go to the documentation of this file.

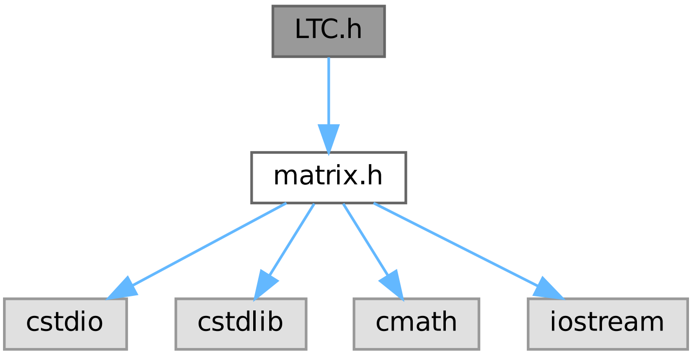
```
#ifndef _Legendre_
#define _Legendre_
#include <tuple>
#include "matrix.h"
using namespace std;

    tuple<Matrix&,Matrix&> Legendre(int n,int m,double fi);
#endif


```

5.35 LTC.h File Reference

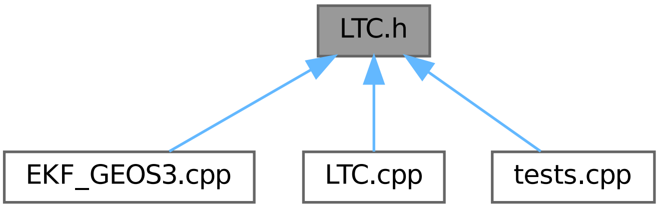
El archivo contiene la funcion LTC.

```
#include "matrix.h"
```

Include dependency graph for LTC.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & LTC (double lon, double lat)

5.35.1 Detailed Description

El archivo contiene la funcion LTC.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file LTC.h.

5.35.2 Function Documentation

5.35.2.1 LTC()

```
Matrix & LTC (
             double lon,
             double lat)
```

Transformation from Greenwich meridian system to local tangent coordinates

Parameters

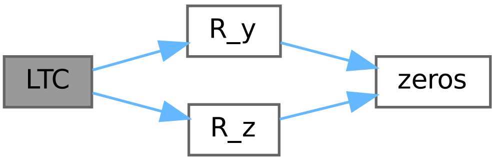
lon	-Geodetic East longitude [rad]
lat	-Geodetic latitude [rad]

Returns

M -Rotation matrix from the Earth equator and Greenwich meridian to the local tangent (East-North-Zenith) coordinate system

Definition at line 10 of file LTC.cpp.

Here is the call graph for this function:



5.36 LTC.h

Go to the documentation of this file.

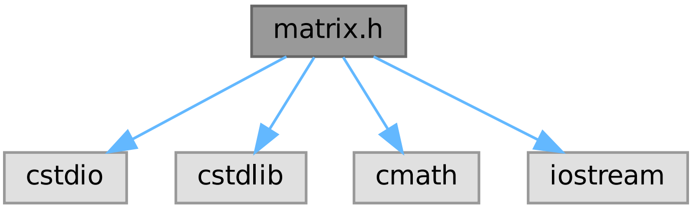
```
#ifndef _LTC_
#define _LTC_
#include "matrix.h"

using namespace std;

    Matrix& LTC(double lon,double lat);
#endif


```

5.37 matrix.h File Reference

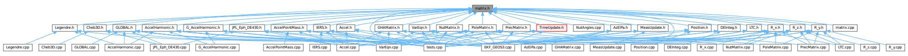
El archivo contiene las funciones para operaciones con matrices y vectores junto con su definicion.

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <iostream>
```

Include dependency graph for matrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Matrix

Functions

- ostream & operator<< (ostream &o, Matrix &m)
- Matrix & zeros (const int n_row, const int n_column)
- Matrix & eye (const int size)
- Matrix & transpose (Matrix &m)
- Matrix & inv (Matrix &m)
- Matrix & zeros (const int n)
- double norm (Matrix &m)
- double dot (Matrix &v, Matrix &w)
- Matrix & cross (Matrix &v, Matrix &w)
- Matrix & extract_vector (Matrix &v, int start, int end)
- Matrix & union_vector (Matrix &v, Matrix &w)
- Matrix & extract_row (Matrix &v, int i)
- Matrix & extract_column (Matrix &v, int i)
- Matrix & assign_row (Matrix &v, Matrix &w, int i)
- Matrix & assign_column (Matrix &v, Matrix &w, int i)

5.37.1 Detailed Description

El archivo contiene las funciones para operaciones con matrices y vectores junto con su definicion.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file matrix.h.

5.37.2 Function Documentation

5.37.2.1 assign_column()

```
Matrix & assign_column (
            Matrix & v,
            Matrix & w,
            int i)
```

Asigna la columna i-1 con w y lo devuelve

Parameters

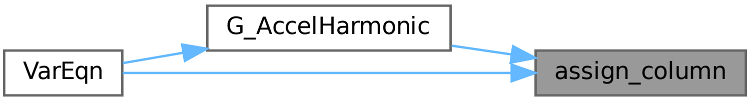
v	Matrix con tamaño x x y ,x pertenece a N y pertenece a N
w	Matrix con tamaño 1 x y ,y pertenece a N
i	es la columna tiene que ser >=1 && <=v.n_column

Returns

Matrix v con la columna i-1 cambiada por los elementos de w

Definition at line 424 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.2 assign_row()

```
Matrix & assign_row (
            Matrix & v,
            Matrix & w,
            int i)
```

Asigna la fila i-1 con w y lo devuelve

Parameters

v	Matrix con tamaño x x y ,x pertenece a N y pertenece a N
w	Matrix con tamaño 1 x y ,y pertenece a N
i	es la fila tiene que ser >=1 && <=v.n_row

Returns

Matrix v con la fila i-1 cambiada por los elementos de w

Definition at line 412 of file matrix.cpp.

5.37.2.3 cross()

```
Matrix & cross (
            Matrix & v,
            Matrix & w)
```

Devulve el producto vectorial

Parameters

v	Matrix con tamaño 1 x 3
w	Matrix con tamaño 1 x 3

Returns

producto escalar de v x w

Definition at line 353 of file matrix.cpp.

5.37.2.4 dot()

```
double dot (
            Matrix & v,
            Matrix & w)
```

Devulve el producto escalar

Parameters

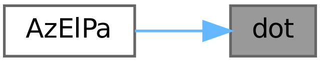
v	Matrix con tamaño 1 x 3
w	Matrix con tamaño 1 x 3

Returns

producto escalar de v·w

Definition at line 342 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.5 extract_column()

```
Matrix & extract_column (
            Matrix & v,
            int i)
```

Extrae la columna i-1 de v y lo devuelve

Parameters

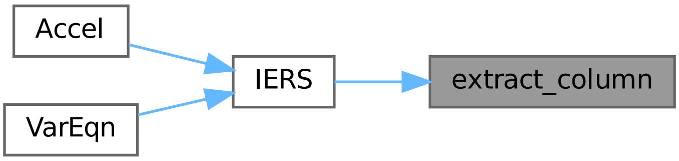
v	Matrix
i	es la columna tiene que ser >=1 && <=v.n_column

Returns

Matrix con la columna i-1 de v

Definition at line 401 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.6 extract_row()

```
Matrix & extract_row (
              Matrix & v,
              int i)
```

Extrae la fila i-1 de v y lo devuelve

Parameters

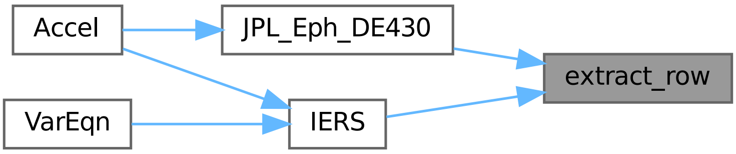
v	Matrix
i	es la fila tiene que ser >=1 && <=v.n_row

Returns

Matrix con la fila i-1 de v

Definition at line 389 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.7 extract_vector()

```
Matrix & extract_vector (
            Matrix & v,
            int start,
            int end)
```

Extrae del vector v desde la posicion start hasta end, incluidos

Parameters

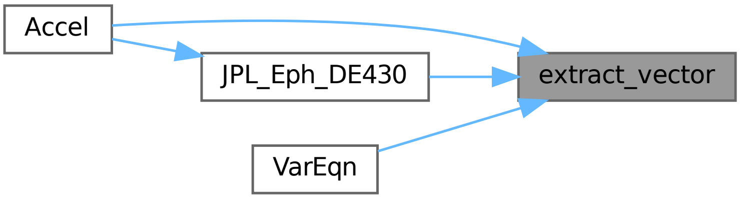
v	Matrix 1 x n
start	inicio del vector resultado
end	fin del vector resultado

Returns

Matrix 1 x (end - start + 1)

Definition at line 364 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.8 eye()

```
Matrix & eye (
            const int size)
```

Crea una Matrix identidad con tamaño size x size

Parameters

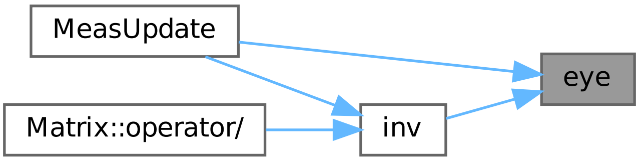
size	dimension de la matriz

Returns

una Matrix tamaño size x size

Definition at line 233 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.9 inv()

```
Matrix & inv (
            Matrix & m)
```

Crea una Matrix inversa de m, sin modificar m

Parameters

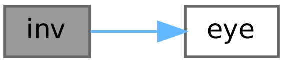
m	Matrix que tiene que ser cuadrada, es decir, con el mismo numero de columnas que filas

Returns

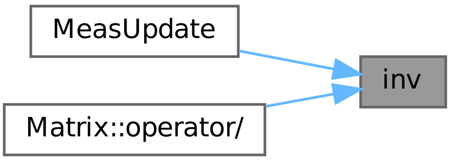
una Matrix inversa de m

Definition at line 267 of file matrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.2.10 norm()

```
double norm (
            Matrix & m)
```

Devulve la norma 2 de una Matrix que simula un vector

Parameters

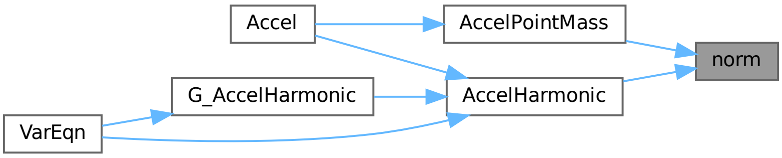
m	Matrix 1 x n_column

Returns

la norma 2 de m

Definition at line 333 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.11 operator<<()

```
ostream & operator<< (
            ostream & o,
            Matrix & m)
```

Definition at line 212 of file matrix.cpp.

5.37.2.12 transpose()

```
Matrix & transpose (
            Matrix & m)
```

Crea una Matrix traspuesta de m,sin modificar m

Parameters

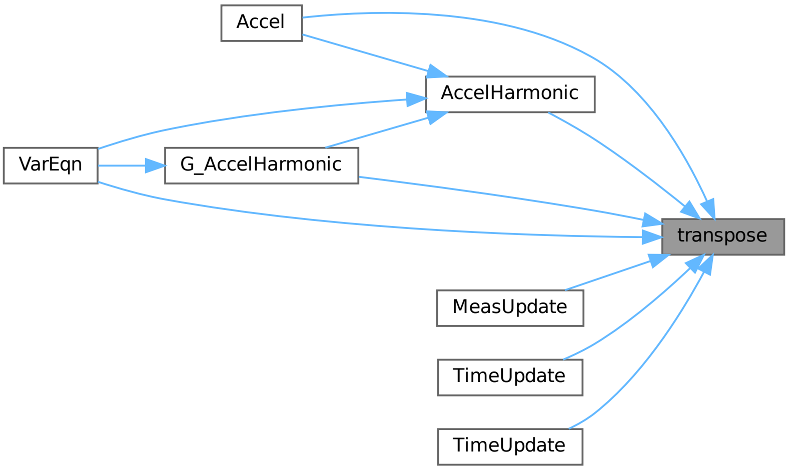
m	Matrix

Returns

una Matrix traspuesta de m

Definition at line 246 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.13 union_vector()

```
Matrix & union_vector (
            Matrix & v,
            Matrix & w)
```

Devuelve la matriz resultado de realizar la union entre v y w

Parameters

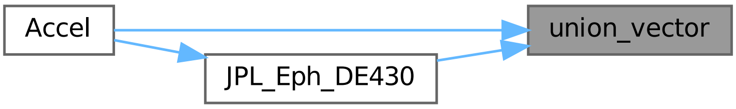
v	Matrix con tamaño 1 x n ,n pertenece a N
w	Matrix con tamaño 1 x n ,n pertenece a N

Returns

producto escalar de v x w

Definition at line 375 of file matrix.cpp.

Here is the caller graph for this function:



5.37.2.14 zeros() [1/2]

```
Matrix & zeros (
             const int n)
```

Crea una Matrix con todas sus componentes a 0

Parameters

n	numero de columnas que tiene la matriz

Returns

una Matrix tamaño 1 x n

Definition at line 323 of file matrix.cpp.

5.37.2.15 zeros() [2/2]

```
Matrix & zeros (
             const int n_row,
             const int n_column)
```

Crea una Matrix con todas sus componentes a 0

Parameters

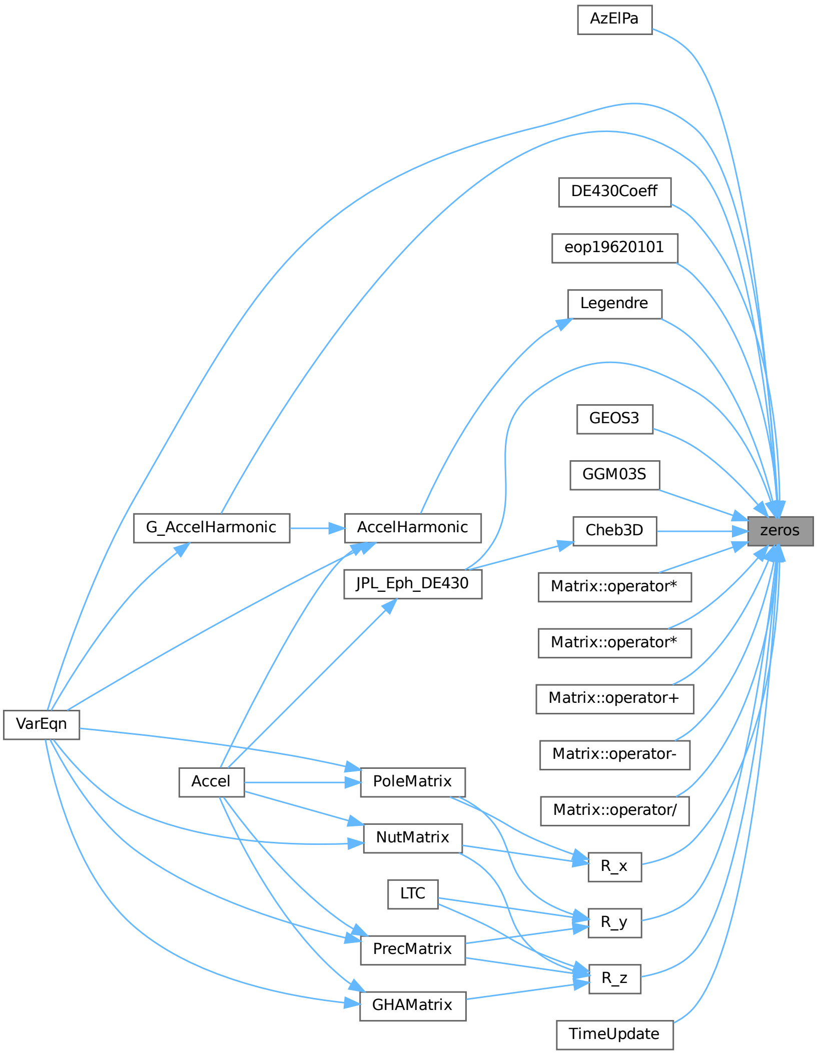
n_row	numero de filas que tiene la matriz
n_column	numero de columnas que tiene la matriz

Returns

una Matrix tamaño n_row x n_column

Definition at line 222 of file matrix.cpp.

Here is the caller graph for this function:



5.38 matrix.h

Go to the documentation of this file.

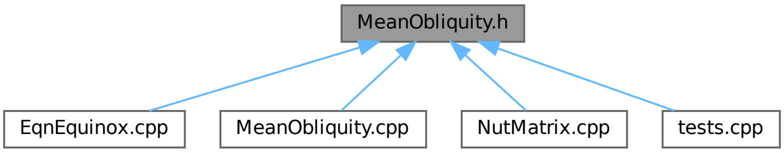
```
00001 #ifndef _MATRIX_
00002 #define _MATRIX_
00003
00004 #include <cstdio>
00005 #include <cstdlib>
00006 #include <cmath>
00007 #include <iostream>
00008
00009 using namespace std;
00010
00018 class Matrix {
00019 public:
00020     int n_row, n_column;
00021     double **data;
00022
00023     // Parameterized constructor
00027     Matrix();
00032     Matrix(const int n);
```

```
    Matrix(const int n_row, const int n_column);

    // Member operators
    double& operator () (const int n);
    double& operator () (const int row, const int column);
    Matrix& operator + (Matrix &m);
    Matrix& operator - (Matrix &m);
    Matrix& operator * (Matrix &m);
    Matrix& operator / (Matrix &m);
    Matrix& operator = (Matrix &m);
    Matrix& operator + (double d);
    Matrix& operator - (double d);
    Matrix& operator * (double d);
    Matrix& operator / (double d);

    // Non-member operators
    friend ostream& operator « (ostream &o, Matrix &m);
};

// Operator overloading
ostream& operator « (ostream &o, Matrix &m);


// Methods
    Matrix& zeros(const int n_row, const int n_column);
    Matrix& eye(const int size);
    Matrix& transpose(Matrix &m);
    Matrix& inv(Matrix &m) ;
    Matrix& zeros(const int n);
    double norm(Matrix &m) ;
    double dot(Matrix &v,Matrix &w);
    Matrix& cross(Matrix &v,Matrix &w);
    Matrix& extract_vector(Matrix &v,int start,int end);

    Matrix& union_vector(Matrix &v,Matrix &w);


    Matrix& extract_row(Matrix &v,int i);


    Matrix& extract_column(Matrix &v,int i);


    Matrix& assign_row(Matrix &v,Matrix &w,int i);


    Matrix& assign_column(Matrix &v,Matrix &w,int i);


#endif
```

5.39 MeanObliquity.h File Reference

El archivo contiene la funcion MeanObliquity.

This graph shows which files directly or indirectly include this file:



Functions

- double MeanObliquity (double Mjd_TT)

5.39.1 Detailed Description

El archivo contiene la funcion MeanObliquity.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file MeanObliquity.h.

5.39.2 Function Documentation

5.39.2.1 MeanObliquity()

```
double MeanObliquity (
              double Mjd_TT)
```

Computes the mean obliquity of the ecliptic

Parameters

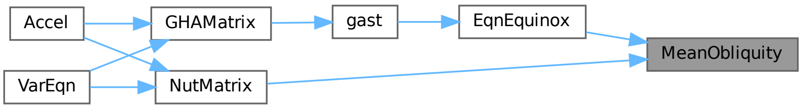
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

obliquity of the ecliptic [rad]

Definition at line 10 of file MeanObliquity.cpp.

Here is the caller graph for this function:



5.40 MeanObliquity.h

Go to the documentation of this file.

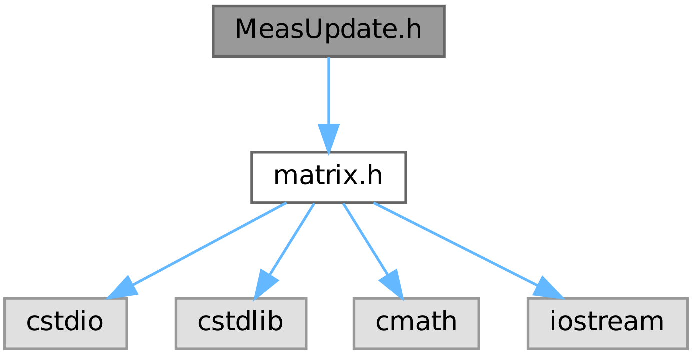
```
00001 #ifndef _MeanObliquity_
00002 #define _MeanObliquity_
00003 
00004 using namespace std;
00005 
00017      double MeanObliquity (double Mjd_TT);
00018 #endif
00019 
00020 
```

5.41 MeasUpdate.h File Reference

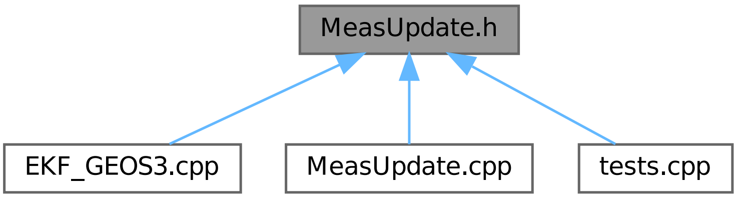
El archivo contiene la funcion MeasUpdate.

```
#include "matrix.h"
```

Include dependency graph for MeasUpdate.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< Matrix &, Matrix &, Matrix & > MeasUpdate (Matrix x, double z, double g, double s, Matrix G, Matrix P, int n)

5.41.1 Detailed Description

El archivo contiene la funcion MeasUpdate.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file MeasUpdate.h.

5.41.2 Function Documentation

5.41.2.1 MeasUpdate()

```
tuple< Matrix &, Matrix &, Matrix & > MeasUpdate (
              Matrix x,
              double z,
              double g,
              double s,
              Matrix G,
              Matrix P,
              int n)
```

Parameters

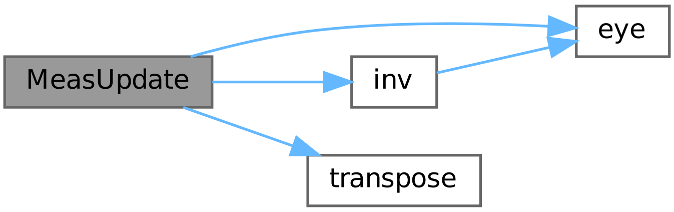
x	Matrix n∗1
z	double
g	double
s	double
G	Matrix 1∗n
P	Matrix n∗n
n	int

Returns

tupla de 3 Matrix

Definition at line 8 of file MeasUpdate.cpp.

Here is the call graph for this function:



5.42 MeasUpdate.h

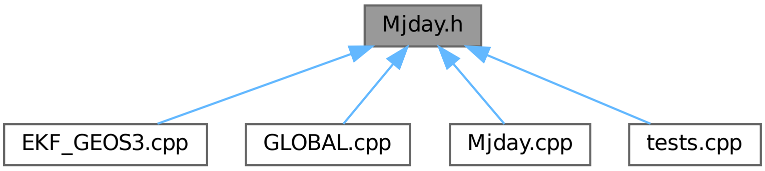
Go to the documentation of this file.

```
00001 #ifndef _MeasUpdate_
00002 #define _MeasUpdate_
00003 #include "matrix.h"
00004
00005 using namespace std;
00006
00023     tuple<Matrix&,Matrix&,Matrix&> MeasUpdate(Matrix x, double z,double g,double s,Matrix G,Matrix P,
      int n);
00024 #endif
00025
00026
```

5.43 Mjday.h File Reference

El archivo contiene la funcion Mjday.

This graph shows which files directly or indirectly include this file:



Functions

- double Mjday (int yr, int mon, int day, int hr=0, int min=0, int sec=0)

5.43.1 Detailed Description

El archivo contiene la funcion Mjday.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Mjday.h.

5.43.2 Function Documentation

5.43.2.1 Mjday()

```
double Mjday (
             int yr,
             int mon,
             int day,
             int hr = 0,
             int min = 0,
             int sec = 0)
```

Parameters

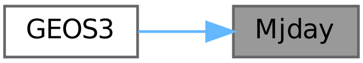
year	- year
mon	- month
day	- day
hr	- universal time hour
min	- universal time min
sec	- universal time sec

Returns

Modified julian date

Definition at line 9 of file Mjday.cpp.

Here is the caller graph for this function:



5.44 Mjday.h

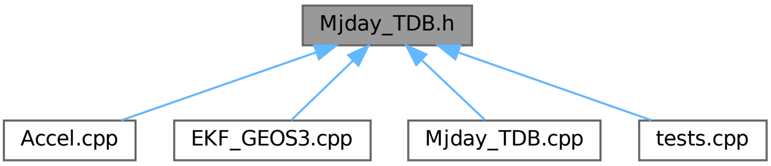
Go to the documentation of this file.

```
#ifndef _Mjday_
#define _Mjday_

using namespace std;

    double Mjday(int yr, int mon,int day,int hr=0,int min=0,int sec=0);
#endif


```

5.45 Mjday_TDB.h File Reference

El archivo contiene la funcion Mjday_TDB.

This graph shows which files directly or indirectly include this file:



Functions

- double Mjday_TDB (double Mjd_TT)

5.45.1 Detailed Description

El archivo contiene la funcion Mjday_TDB.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Mjday_TDB.h.

5.45.2 Function Documentation

5.45.2.1 Mjday_TDB()

```
double Mjday_TDB (
            double Mjd_TT)
```

Parameters

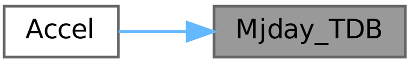
Mjd_TT	- Modified julian date (TT)

Returns

Modified julian date (TDB)

Definition at line 10 of file Mjday_TDB.cpp.

Here is the caller graph for this function:



5.46 Mjday_TDB.h

Go to the documentation of this file.

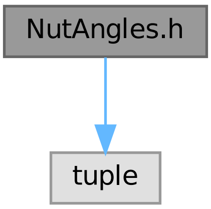
```
#ifndef _Mjday_TDB_
#define _Mjday_TDB_

using namespace std;

    double Mjday_TDB(double Mjd_TT);
#endif


```

5.47 NutAngles.h File Reference

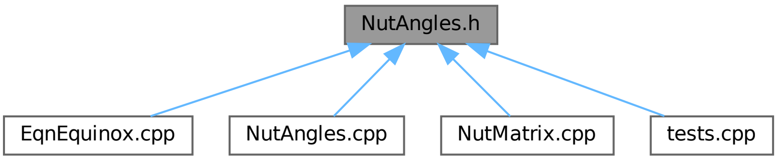
El archivo contiene la funcion NutAngles.

```
#include <tuple>
```

Include dependency graph for NutAngles.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< double, double > NutAngles (double Mjd_TT)

5.47.1 Detailed Description

El archivo contiene la funcion NutAngles.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file NutAngles.h.

5.47.2 Function Documentation

5.47.2.1 NutAngles()

```
tuple< double, double > NutAngles (
              double Mjd_TT)
```

Nutation in longitude and obliquity

Parameters

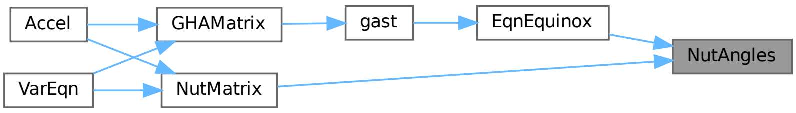
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

tupla < dpsi,deps> Nutation Angles

Definition at line 11 of file NutAngles.cpp.

Here is the caller graph for this function:



5.48 NutAngles.h

Go to the documentation of this file.

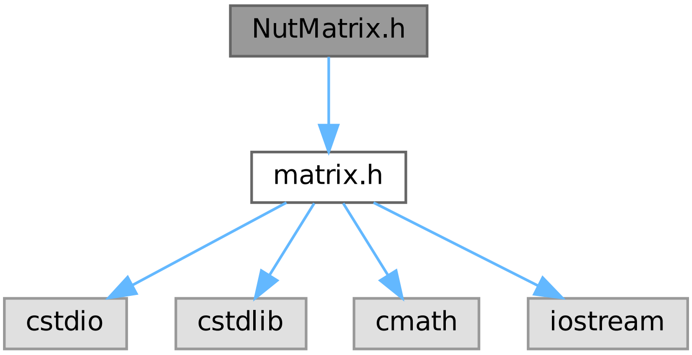
```
#ifndef _NutAngles_
#define _NutAngles_
#include <tuple>
using namespace std;

    tuple<double,double> NutAngles (double Mjd_TT);
#endif


```

5.49 NutMatrix.h File Reference

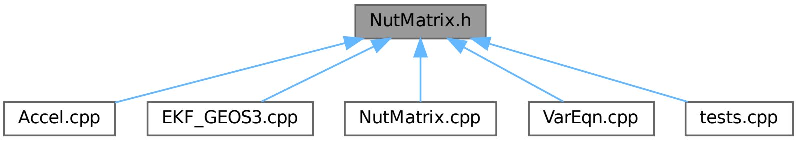
El archivo contiene la funcion NutMatrix.

```
#include "matrix.h"
```

Include dependency graph for NutMatrix.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & NutMatrix (double Mjd_TT)

5.49.1 Detailed Description

El archivo contiene la funcion NutMatrix.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file NutMatrix.h.

5.49.2 Function Documentation

5.49.2.1 NutMatrix()

```
Matrix & NutMatrix (
             double Mjd_TT)
```

Transformation from mean to true equator and equinox

Parameters

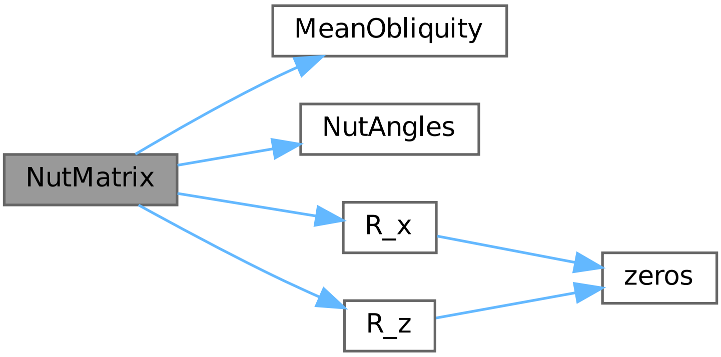
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

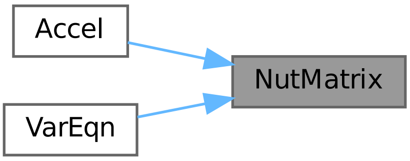
NutMat Nutation matrix

Definition at line 13 of file NutMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.50 NutMatrix.h

Go to the documentation of this file.

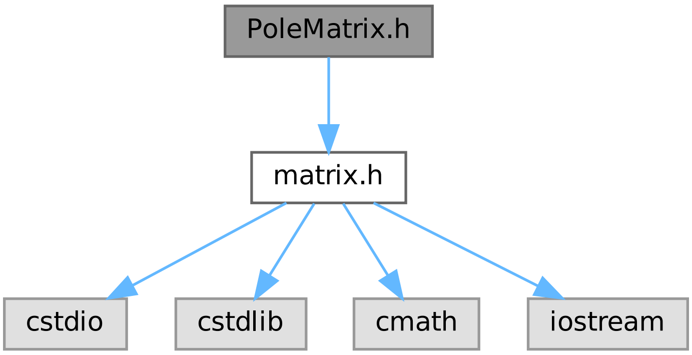
```
#ifndef _NutMatrix_
#define _NutMatrix_
#include "matrix.h"

using namespace std;

    Matrix& NutMatrix (double Mjd_TT);
#endif


```

5.51 PoleMatrix.h File Reference

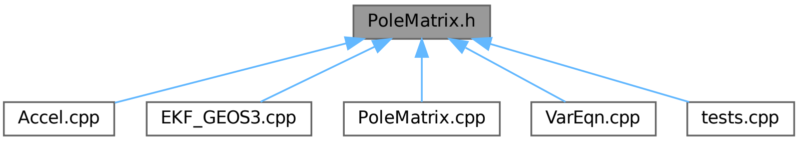
El archivo contiene la funcion PoleMatrix.

```
#include "matrix.h"
```

Include dependency graph for PoleMatrix.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & PoleMatrix (double xp, double yp)

5.51.1 Detailed Description

El archivo contiene la funcion PoleMatrix.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file PoleMatrix.h.

5.51.2 Function Documentation

5.51.2.1 PoleMatrix()

```
Matrix & PoleMatrix (
            double xp,
            double yp)
```

Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date

Parameters

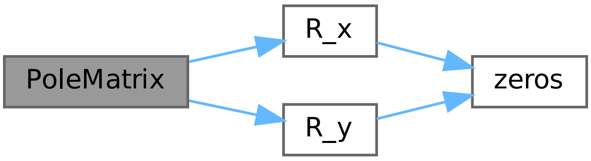
Pole	coordinte(xp,yp)

Returns

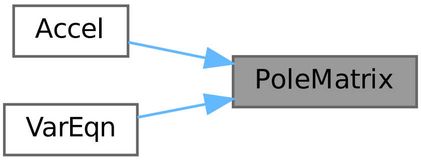
PoleMat Pole matrix

Definition at line 11 of file PoleMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.52 PoleMatrix.h

Go to the documentation of this file.

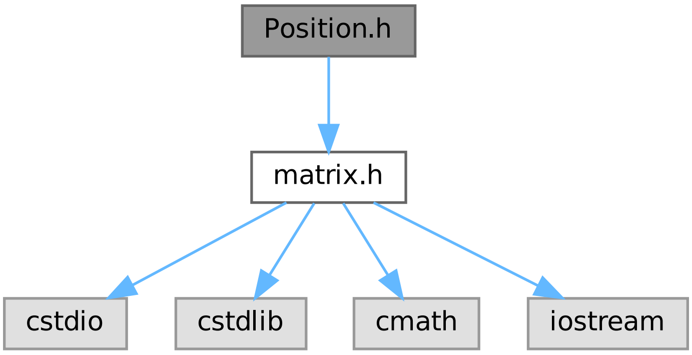
```
#ifndef _PoleMatrix_
#define _PoleMatrix_
#include "matrix.h"

using namespace std;

    Matrix& PoleMatrix (double xp,double yp);
#endif
```

5.53 Position.h File Reference

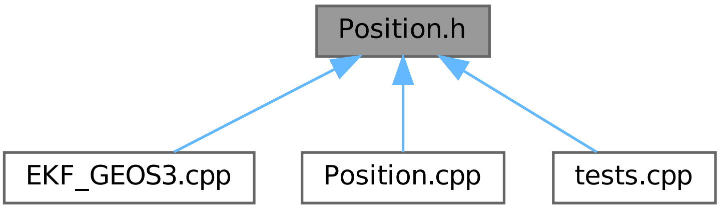
El archivo contiene la funcion Position.

```
#include "matrix.h"
```

Include dependency graph for Position.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & Position (double lon, double lat, double h)

5.53.1 Detailed Description

El archivo contiene la funcion Position.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Position.h.

5.53.2 Function Documentation

5.53.2.1 Position()

```
Matrix & Position (
            double lon,
            double lat,
            double h)
```

Parameters

lon	longitude [rad]
lat	latitude [rad]
h	altitude [m]

Returns

Position vector (r [m]) from geodetic coordinates (Longitude [rad],latitude [rad], altitude [m])

Definition at line 10 of file Position.cpp.

5.54 Position.h

Go to the documentation of this file.

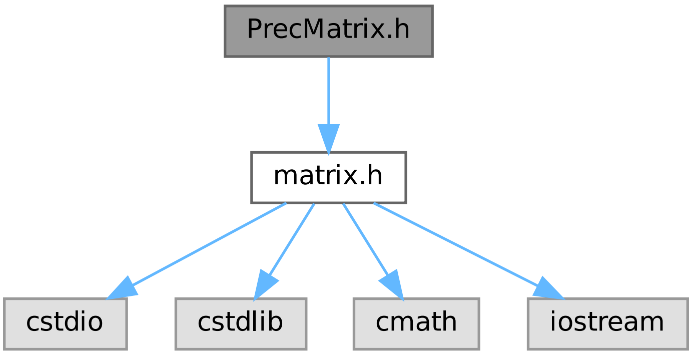
```
#ifndef _Position_
#define _Position_
#include "matrix.h"

using namespace std;

    Matrix& Position(double lon,double lat,double h);
#endif


```

5.55 PrecMatrix.h File Reference

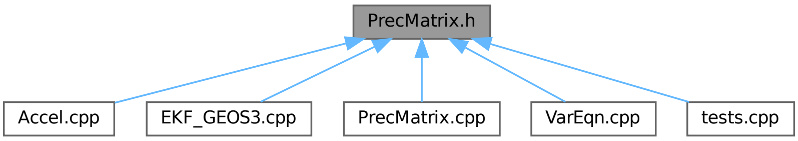
El archivo contiene la funcion PrecMatrix.

```
#include "matrix.h"
```

Include dependency graph for PrecMatrix.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & PrecMatrix (double Mjd_1, double Mjd_2)

5.55.1 Detailed Description

El archivo contiene la funcion PrecMatrix.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file PrecMatrix.h.

5.55.2 Function Documentation

5.55.2.1 PrecMatrix()

```
Matrix & PrecMatrix (
            double Mjd_1,
            double Mjd_2)
```

Precession transformation of equatorial coordinates

Parameters

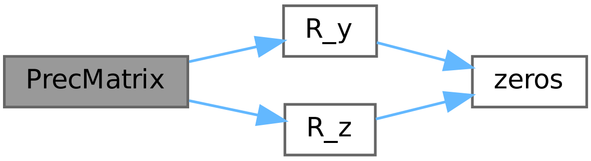
Mjd← *_1*	Epoch given (Modified Julian Date TT)
MjD← *_2*	Epoch to precess to (Modified Julian Date TT)

Returns

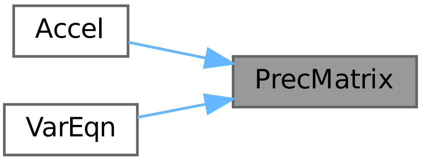
PrecMat Precession transformation matrix

Definition at line 13 of file PrecMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.56 PrecMatrix.h

Go to the documentation of this file.

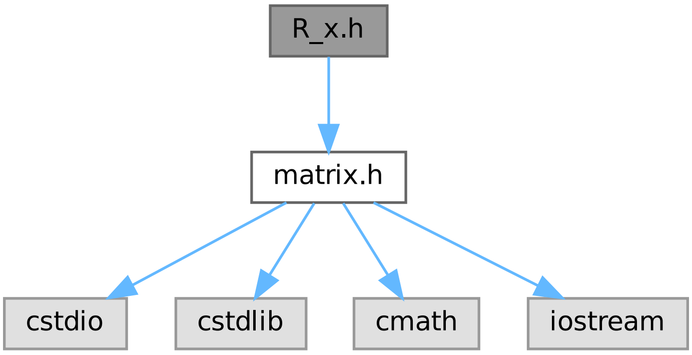
```
#ifndef _PrecMatrix_
#define _PrecMatrix_
#include "matrix.h"

using namespace std;

    Matrix& PrecMatrix (double Mjd_1, double Mjd_2);
#endif


```

5.57 R_x.h File Reference

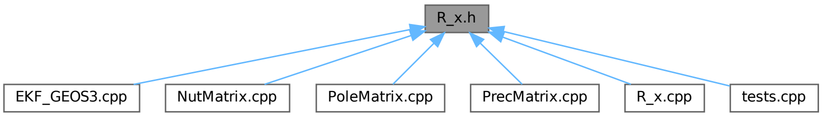
El archivo contiene la funcion R_x.

```
#include "matrix.h"
```

Include dependency graph for R_x.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & R_x (double angle)

5.57.1 Detailed Description

El archivo contiene la funcion R_x.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_x.h.

5.57.2 Function Documentation

5.57.2.1 R_x()

```
Matrix & R_x (
              double angle)
```

Parameters

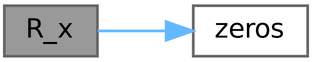
angle	angulo de rotacion

Returns

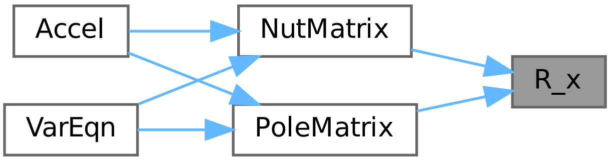
Matrix resultado

Definition at line 9 of file R_x.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.58 R_x.h

Go to the documentation of this file.

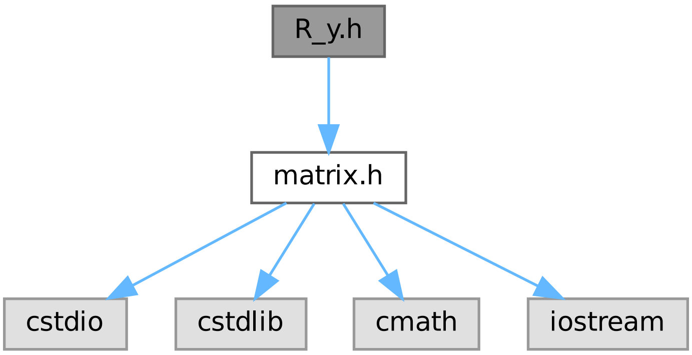
```
00001 #ifndef _R_x_
00002 #define _R_x_
00003 #include "matrix.h"
00004 
00005 using namespace std;
00006 
00017     Matrix& R_x(double angle);
00018 #endif
00019 
00020 
```

5.59 R_y.h File Reference

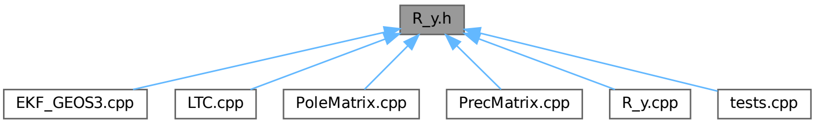
El archivo contiene la funcion R_y.

```
#include "matrix.h"
```

Include dependency graph for R_y.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & R_y (double angle)

5.59.1 Detailed Description

El archivo contiene la funcion R_y.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_y.h.

5.59.2 Function Documentation

5.59.2.1 R_y()

```
Matrix & R_y (
            double angle)
```

Parameters

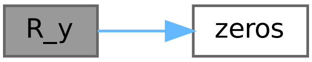
angle	angulo de rotacion

Returns

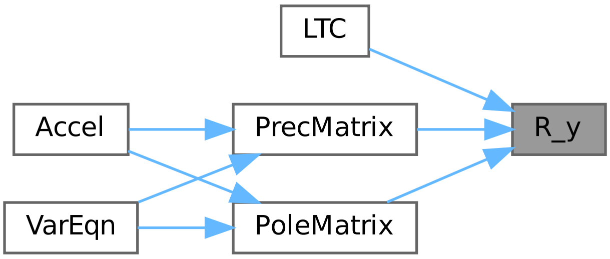
Matrix resultado

Definition at line 9 of file R_y.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.60 R_y.h

Go to the documentation of this file.

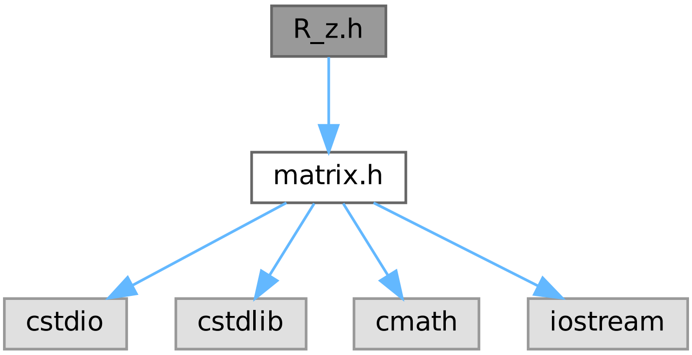
```
#ifndef _R_y_
#define _R_y_
#include "matrix.h"

using namespace std;

    Matrix& R_y(double angle);
#endif


```

5.61 R_z.h File Reference

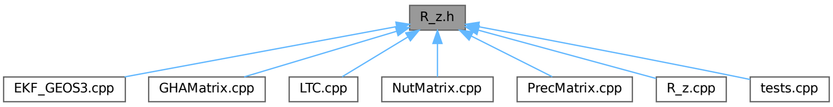
El archivo contiene la funcion R_z.

```
#include "matrix.h"
```

Include dependency graph for R_z.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & R_z (double angle)

5.61.1 Detailed Description

El archivo contiene la funcion R_z.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_z.h.

5.61.2 Function Documentation

5.61.2.1 R_z()

```
Matrix & R_z (
             double angle)
```

Parameters

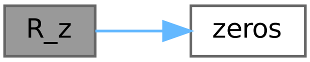
angle	angulo de rotacion

Returns

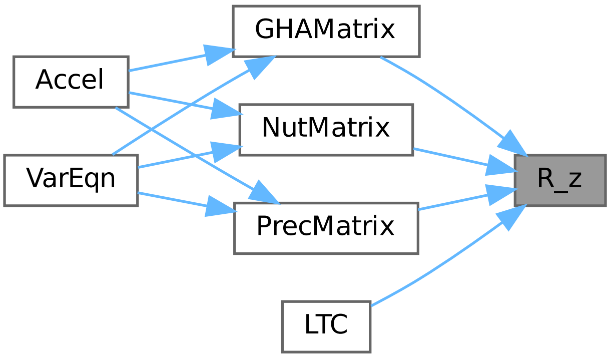
Matrix resultado

Definition at line 9 of file R_z.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.62 R_z.h

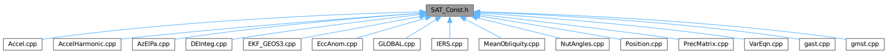
Go to the documentation of this file.

```
#ifndef _R_z_
#define _R_z_
#include "matrix.h"

using namespace std;

    Matrix& R_z(double angle);
#endif


```

5.63 SAT_Const.h File Reference

El archivo contiene SAT_Const.

This graph shows which files directly or indirectly include this file:



Variables

- const double pi =3.14159265358979323846264338327950288419716939937510582097494
- const double eps =2.22044604925031e-16
- const double pi2 = 2∗pi
- const double Rad = pi/180
- const double Deg = 180/pi
- const double Arcs = 3600∗180/pi
- const double MJD_J2000 = 51544.5
- const double T_B1950 = -0.500002108
- const double c_light = 299792458.000000000
- const double AU = 149597870700.000000
- const double R_Earth = 6378.1363e3
- const double f_Earth = 1/298.257223563
- const double R_Sun = 696000e3
- const double R_Moon = 1738e3
- const double omega_Earth = 15.04106717866910/3600∗Rad
- const double GM_Earth = 398600.435436e9
- const double GM_Sun = 132712440041.939400e9
- const double GM_Moon = GM_Earth/81.30056907419062
- const double GM_Mercury = 22031.780000e9
- const double GM_Venus = 324858.592000e9
- const double GM_Mars = 42828.375214e9
- const double GM_Jupiter = 126712764.800000e9
- const double GM_Saturn = 37940585.200000e9
- const double GM_Uranus = 5794548.600000e9
- const double GM_Neptune = 6836527.100580e9
- const double GM_Pluto = 977.0000000000009e9
- const double P_Sol = 1367/c_light

5.63.1 Detailed Description

El archivo contiene SAT_Const.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file SAT_Const.h.

5.63.2 Variable Documentation

5.63.2.1 Arcs

```
const double Arcs = 3600*180/pi
```

Definition at line 17 of file SAT_Const.h.

5.63.2.2 AU

```
const double AU = 149597870700.000000
```

Definition at line 23 of file SAT_Const.h.

5.63.2.3 c_light

```
const double c_light = 299792458.000000000
```

Definition at line 22 of file SAT_Const.h.

5.63.2.4 Deg

```
const double Deg = 180/pi
```

Definition at line 16 of file SAT_Const.h.

5.63.2.5 eps

```
const double eps =2.22044604925031e-16
```

Definition at line 12 of file SAT_Const.h.

5.63.2.6 f_Earth

```
const double f_Earth = 1/298.257223563
```

Definition at line 29 of file SAT_Const.h.

5.63.2.7 GM_Earth

```
const double GM_Earth = 398600.435436e9
```

Definition at line 37 of file SAT_Const.h.

5.63.2.8 GM_Jupiter

```
const double GM_Jupiter = 126712764.800000e9
```

Definition at line 43 of file SAT_Const.h.

5.63.2.9 GM_Mars

```
const double GM_Mars = 42828.375214e9
```

Definition at line 42 of file SAT_Const.h.

5.63.2.10 GM_Mercury

```
const double GM_Mercury = 22031.780000e9
```

Definition at line 40 of file SAT_Const.h.

5.63.2.11 GM_Moon

```
const double GM_Moon = GM_Earth/81.30056907419062
```

Definition at line 39 of file SAT_Const.h.

5.63.2.12 GM_Neptune

```
const double GM_Neptune = 6836527.100580e9
```

Definition at line 46 of file SAT_Const.h.

5.63.2.13 GM_Pluto

```
const double GM_Pluto = 977.0000000000009e9
```

Definition at line 47 of file SAT_Const.h.

5.63.2.14 GM_Saturn

```
const double GM_Saturn = 37940585.200000e9
```

Definition at line 44 of file SAT_Const.h.

5.63.2.15 GM_Sun

```
const double GM_Sun = 132712440041.939400e9
```

Definition at line 38 of file SAT_Const.h.

5.63.2.16 GM_Uranus

```
const double GM_Uranus = 5794548.600000e9
```

Definition at line 45 of file SAT_Const.h.

5.63.2.17 GM_Venus

```
const double GM_Venus = 324858.592000e9
```

Definition at line 41 of file SAT_Const.h.

5.63.2.18 MJD_J2000

```
const double MJD_J2000 = 51544.5
```

Definition at line 20 of file SAT_Const.h.

5.63.2.19 omega_Earth

```
const double omega_Earth = 15.04106717866910/3600*Rad
```

Definition at line 34 of file SAT_Const.h.

5.63.2.20 P_Sol

```
const double P_Sol = 1367/c_light
```

Definition at line 50 of file SAT_Const.h.

5.63.2.21 pi

```
const double pi =3.14159265358979323846264338327950288419716939937510582097494
```

Definition at line 11 of file SAT_Const.h.

5.63.2.22 pi2

```
const double pi2 = 2*pi
```

Definition at line 14 of file SAT_Const.h.

5.63.2.23 R_Earth

```
const double R_Earth = 6378.1363e3
```

Definition at line 28 of file SAT_Const.h.

5.63.2.24 R_Moon

```
const double R_Moon = 1738e3
```

Definition at line 31 of file SAT_Const.h.

5.63.2.25 R_Sun

```
const double R_Sun = 696000e3
```

Definition at line 30 of file SAT_Const.h.

5.63.2.26 Rad

```
const double Rad = pi/180
```

Definition at line 15 of file SAT_Const.h.

5.63.2.27 T_B1950

```
const double T_B1950 = -0.500002108
```

Definition at line 21 of file SAT_Const.h.

5.64 SAT_Const.h

Go to the documentation of this file.

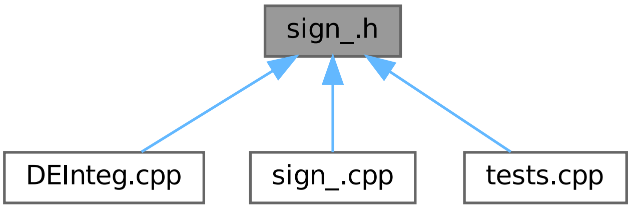
```
00001 #ifndef _SAT_Const_
00002 #define _SAT_Const_
00003
00004
00011      const double pi=3.14159265358979323846264338327950288419716939937510582097494;
00012      const double eps=2.22044604925031e-16;
00013     // Mathematical constants
00014     const double pi2       = 2*pi;                // 2pi
00015     const double Rad       = pi/180;              // Radians per degree
00016     const double Deg       = 180/pi;              // Degrees per radian
00017     const double Arcs      = 3600*180/pi;         // Arcseconds per radian
00018
00019     // General
00020     const double MJD_J2000 = 51544.5;             // Modified Julian Date of J2000
00021     const double T_B1950   = -0.500002108;        // Epoch B1950
00022     const double c_light   = 299792458.000000000; // Speed of light  [m/s]; DE430
00023     const double AU        = 149597870700.000000; // Astronomical unit [m]; DE430
00024
00025     // Physical parameters of the Earth, Sun and Moon
00026
00027     // Equatorial radius and flattening
00028     const double R_Earth   = 6378.1363e3;      // Earth's radius [m]; DE430
00029     const double f_Earth   = 1/298.257223563;  // Flattening; WGS-84
00030     const double R_Sun     = 696000e3;         // Sun's radius [m]; DE430
00031     const double R_Moon    = 1738e3;           // Moon's radius [m]; DE430
00032
00033     // Earth rotation (derivative of GMST at J2000; differs from inertial period by precession)
00034     const double omega_Earth = 15.04106717866910/3600*Rad;   // [rad/s]; WGS-84
00035
00036     // Gravitational coefficients
00037     const double GM_Earth    = 398600.435436e9;                  // [m^3/s^2]; DE430
00038     const double GM_Sun      = 132712440041.939400e9;            // [m^3/s^2]; DE430
00039     const double GM_Moon     = GM_Earth/81.30056907419062;       // [m^3/s^2]; DE430
00040     const double GM_Mercury  = 22031.780000e9;                   // [m^3/s^2]; DE430
```

```
    const double GM_Venus    = 324858.592000e9;                  // [m^3/s^2]; DE430
    const double GM_Mars     = 42828.375214e9;                   // [m^3/s^2]; DE430
    const double GM_Jupiter  = 126712764.800000e9;               // [m^3/s^2]; DE430
    const double GM_Saturn   = 37940585.200000e9;                // [m^3/s^2]; DE430
    const double GM_Uranus   = 5794548.600000e9;                 // [m^3/s^2]; DE430
    const double GM_Neptune  = 6836527.100580e9;                 // [m^3/s^2]; DE430
    const double GM_Pluto    = 977.0000000000009e9;              // [m^3/s^2]; DE430

    // Solar radiation pressure at 1 AU
    const double P_Sol       = 1367/c_light; // [N/m^2] (~1367 W/m^2); IERS 96

#endif
```

5.65 sign_.h File Reference

El archivo contiene la funcion sign_.

This graph shows which files directly or indirectly include this file:



Functions

- double sign_ (double a, double b)

5.65.1 Detailed Description

El archivo contiene la funcion sign_.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file sign_.h.

5.65.2 Function Documentation

5.65.2.1 sign_()

```
double sign_ (
               double a,
               double b)
```

Parameters

a	double
b	double

Returns

absolute value of a with sign of b

Definition at line 10 of file sign_.cpp.

5.66 sign_.h

Go to the documentation of this file.

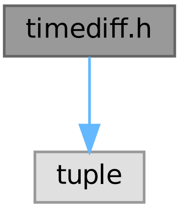
```
#ifndef _sign__
#define _sign__

using namespace std;

    double sign_(double a, double b);
#endif


```

5.67 timediff.h File Reference

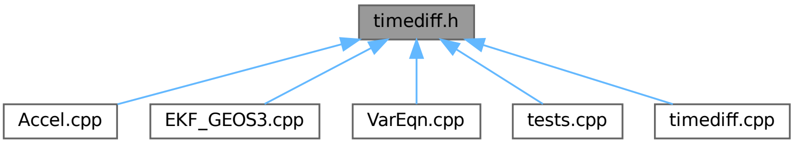
El archivo contiene la funcion timediff.

`#include <tuple>`

Include dependency graph for timediff.h:



This graph shows which files directly or indirectly include this file:



Functions

- tuple< double, double, double, double, double > timediff (double UT1_UTC, double TAI_UTC)

5.67.1 Detailed Description

El archivo contiene la funcion timediff.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file timediff.h.

5.67.2 Function Documentation

5.67.2.1 timediff()

```
tuple< double, double, double, double, double > timediff (
             double UT1_UTC,
             double TAI_UTC)
```

Time differences [s]

Parameters

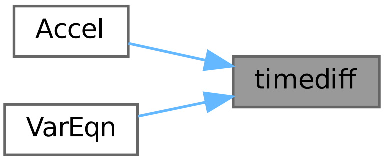
UT1_UTC	double
TAI_UTC	double

Returns

tupla <UT1_TAI, UTC_GPS, UT1_GPS, TT_UTC, GPS_UTC>

Definition at line 9 of file timediff.cpp.

Here is the caller graph for this function:



5.68 timediff.h

Go to the documentation of this file.

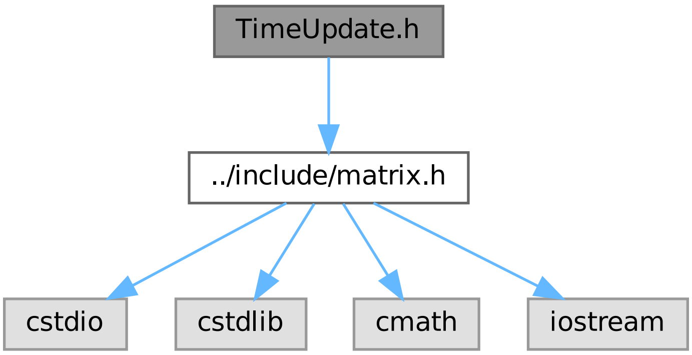
```
#ifndef _timediff_
#define _timediff_
#include <tuple>
using namespace std;

    tuple<double,double,double,double,double> timediff(double UT1_UTC,double TAI_UTC);
#endif


```

5.69 TimeUpdate.h File Reference

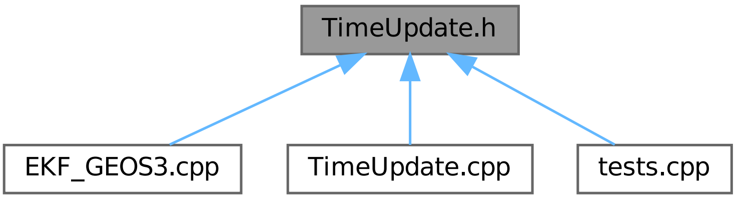
El archivo contiene la funcion TimeUpdate.

`#include "../include/matrix.h"`

Include dependency graph for TimeUpdate.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & TimeUpdate (Matrix P, Matrix Phi, Matrix Qdt)
- Matrix & TimeUpdate (Matrix P, Matrix Phi)

5.69.1 Detailed Description

El archivo contiene la funcion TimeUpdate.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file TimeUpdate.h.

5.69.2 Function Documentation

5.69.2.1 TimeUpdate() [1/2]

```
Matrix & TimeUpdate (
             Matrix P,
             Matrix Phi)
```

Parameters

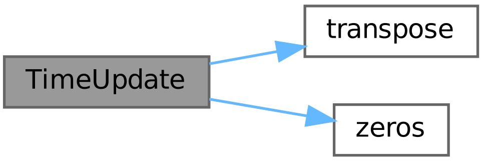
P	Matrix
Phi	Matrix

Returns

Matrix

Definition at line 15 of file TimeUpdate.cpp.

Here is the call graph for this function:



5.69.2.2 TimeUpdate() [2/2]

```
Matrix & TimeUpdate (
             Matrix P,
             Matrix Phi,
             Matrix Qdt)
```

Parameters

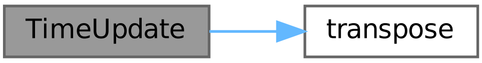
P	Matrix
Phi	Matrix
Qdt	Matrix

Returns

Matrix

Definition at line 9 of file TimeUpdate.cpp.

Here is the call graph for this function:



5.70 TimeUpdate.h

Go to the documentation of this file.

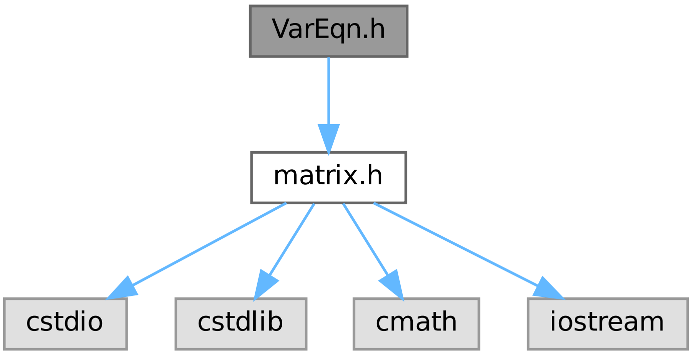
```
#ifndef _TimeUpdate_
#define _TimeUpdate_
using namespace std;
#include "../include/matrix.h"

    Matrix& TimeUpdate(Matrix P,Matrix Phi,Matrix Qdt);

    Matrix& TimeUpdate(Matrix P,Matrix Phi);
#endif


```

5.71 VarEqn.h File Reference

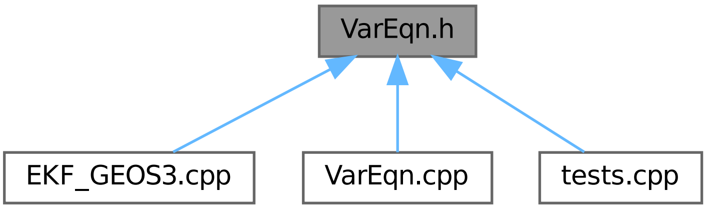
El archivo contiene la funcion VarEqn.

```
#include "matrix.h"
```

Include dependency graph for VarEqn.h:



This graph shows which files directly or indirectly include this file:



Functions

- Matrix & VarEqn (double x, Matrix yPhi)

5.71.1 Detailed Description

El archivo contiene la funcion VarEqn.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file VarEqn.h.

5.71.2 Function Documentation

5.71.2.1 VarEqn()

```
Matrix & VarEqn (
            double x,
            Matrix yPhi)
```

Computes the variational equations, i.e. the derivative of the state vector and the state transition matrix

Parameters

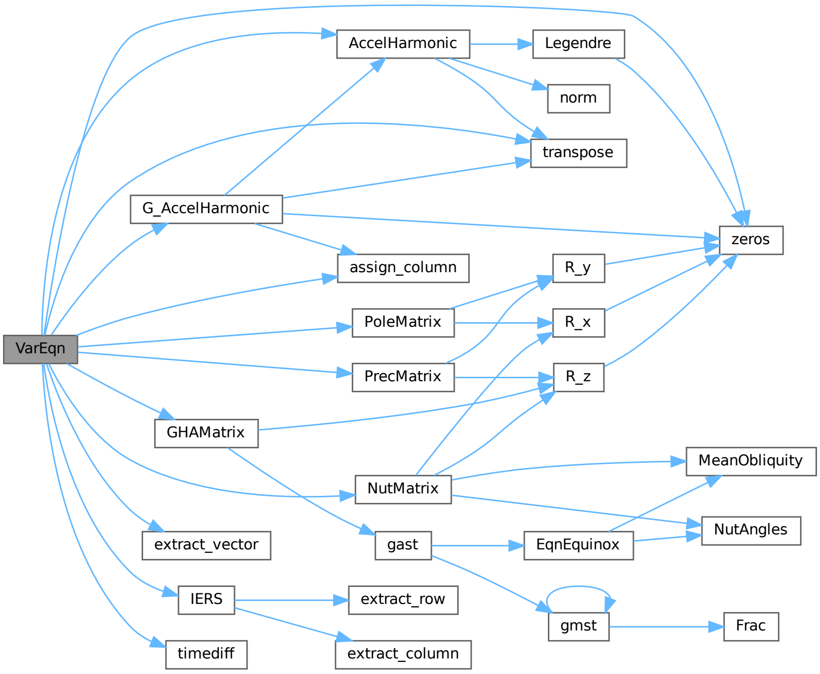
x	Time since epoch in [s]
yPhi	(6+36)-dim vector comprising the state vector (y) and the state transition matrix (Phi) in column wise storage order

Returns

yPhip Derivative of yPhi

Definition at line 18 of file VarEqn.cpp.

Here is the call graph for this function:



5.72 VarEqn.h

Go to the documentation of this file.

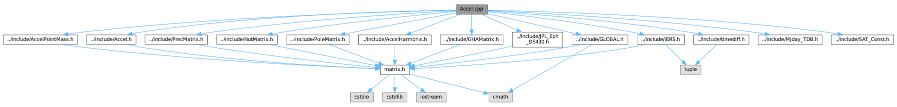
```
#ifndef _VarEqn_
#define _VarEqn_
#include "matrix.h"

using namespace std;

    Matrix& VarEqn(double x, Matrix yPhi);
#endif


```

5.73 Accel.cpp File Reference

El archivo contiene las implementaciones de Accel.h.

```
#include "../include/Accel.h"
#include "../include/PrecMatrix.h"
#include "../include/NutMatrix.h"
#include "../include/IERS.h"
#include "../include/timediff.h"
#include "../include/PoleMatrix.h"
#include "../include/AccelHarmonic.h"
#include "../include/GHAMatrix.h"
#include "../include/JPL_Eph_DE430.h"
#include "../include/GLOBAL.h"
#include "../include/AccelPointMass.h"
#include "../include/Mjday_TDB.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for Accel.cpp:



Functions

- Matrix & Accel (double x, Matrix Y)

5.73.1 Detailed Description

El archivo contiene las implementaciones de Accel.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Accel.cpp.

5.73.2 Function Documentation

5.73.2.1 Accel()

```
Matrix & Accel (
             double x,
             Matrix Y)
```

Computes the acceleration of an Earth orbiting satellite due to

- the Earth's harmonic gravity field,
- the gravitational perturbations of the Sun and Moon
- the solar radiation pressure and
- the atmospheric drag

Parameters

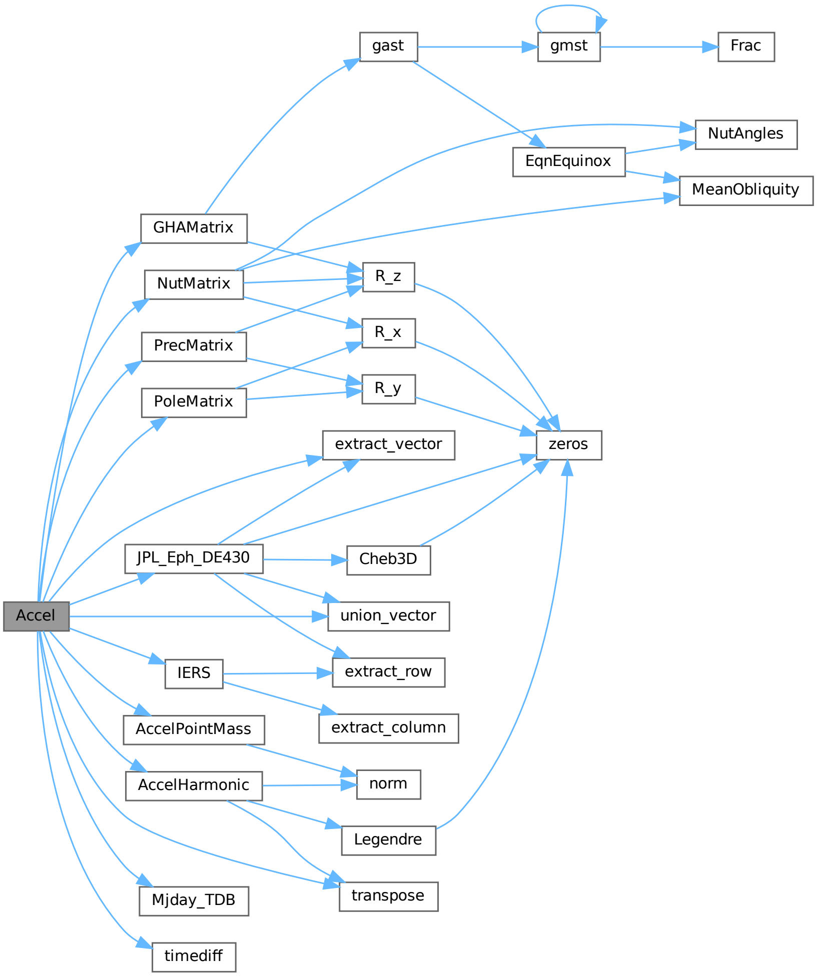
Mjd_TT	Terrestrial Time (Modified Julian Date)
Y	Satellite state vector in the ICRF/EME2000 system

Returns

dY Acceleration (a=d^2r/dt^2) in the ICRF/EME2000 system

Definition at line 20 of file Accel.cpp.

Here is the call graph for this function:



5.74 Accel.cpp

Go to the documentation of this file.

```
00001 #include "../include/Accel.h"
00002 #include "../include/PrecMatrix.h"
00003 #include "../include/NutMatrix.h"
00004 #include "../include/IERS.h"
00005 #include "../include/timediff.h"
00006 #include "../include/PoleMatrix.h"
00007 #include "../include/AccelHarmonic.h"
00008 #include "../include/GHAMatrix.h"
00009 #include "../include/JPL_Eph_DE430.h"
00010 #include "../include/GLOBAL.h"
00011 #include "../include/AccelPointMass.h"
00012 #include "../include/Mjday_TDB.h"
00013 #include "../include/SAT_Const.h"
```

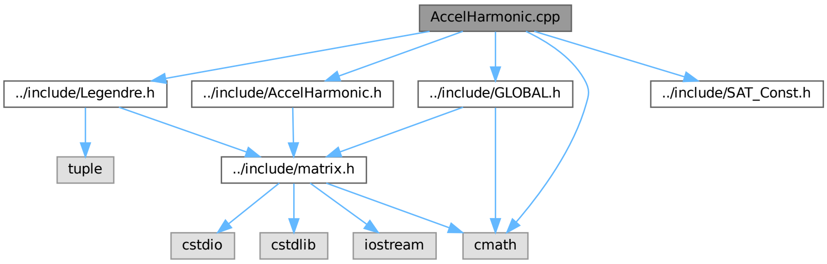
```
        Matrix& Accel(double x,Matrix Y){
            if(Y.n_row<Y.n_column){
                Y=transpose(Y);
            }
    auto [x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC] = IERS(eopdata,AuxParam.Mjd_UTC + x/86400,'l');
    auto [UT1_TAI,UTC_GPS,UT1_GPS,TT_UTC,GPS_UTC] = timediff(UT1_UTC,TAI_UTC);
    long double Mjd_UT1 = AuxParam.Mjd_UTC + x/86400 + UT1_UTC/86400;
    long double Mjd_TT = AuxParam.Mjd_UTC + x/86400 + TT_UTC/86400;

    Matrix P = PrecMatrix(MJD_J2000,Mjd_TT);
    Matrix N = NutMatrix(Mjd_TT);
    Matrix T = N * P;
    Matrix E = PoleMatrix(x_pole,y_pole) * GHAMatrix(Mjd_UT1) * T;

    long double MJD_TDB = Mjday_TDB(Mjd_TT);
    auto [r_Mercury,r_Venus,r_Earth,r_Mars,r_Jupiter,r_Saturn,r_Uranus,r_Neptune,r_Pluto,r_Moon,r_Sun] = JPL_Eph_DE430(MJD_TDB);

    // Acceleration due to harmonic gravity field
    Matrix a = AccelHarmonic(transpose(extract_vector(Y,1,3)), E, AuxParam.n, AuxParam.m);

    // Luni-solar perturbations
    if (AuxParam.sun){
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Sun,GM_Sun);}

    if (AuxParam.moon){
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Moon,GM_Moon);}
    // Planetary perturbations
    if (AuxParam.planets){
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Mercury,GM_Mercury);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Venus,GM_Venus);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Mars,GM_Mars);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Jupiter,GM_Jupiter);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Saturn,GM_Saturn);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Uranus,GM_Uranus);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Neptune,GM_Neptune);
            a = a + AccelPointMass(extract_vector(Y,1,3),r_Pluto,GM_Pluto);}

    return union_vector(extract_vector(Y,4,6),a);

    }
```

5.75 AccelHarmonic.cpp File Reference

El archivo contiene las implementaciones de AccelHarmonic.h.

```
#include "../include/AccelHarmonic.h"
#include "../include/Legendre.h"
#include "../include/GLOBAL.h"
#include "../include/SAT_Const.h"
#include <cmath>
```

Include dependency graph for AccelHarmonic.cpp:



Functions

- Matrix & AccelHarmonic (Matrix r, Matrix E, int n_max, int m_max)

5.75.1 Detailed Description

El archivo contiene las implementaciones de AccelHarmonic.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AccelHarmonic.cpp.

5.75.2 Function Documentation

5.75.2.1 AccelHarmonic()

```
Matrix & AccelHarmonic (
              Matrix r,
              Matrix E,
              int n_max,
              int m_max)
```

Parameters

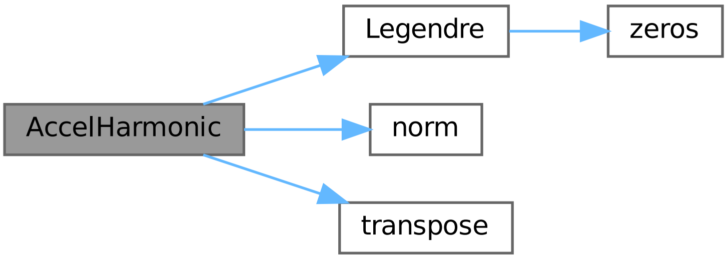
r	Satellite position vector in the inertial system
E	Transformation matrix to body-fixed system
n_max	Maximum degree
m_max	Maximum order (m_max<=n_max; m_max=0 for zonals, only)

Returns

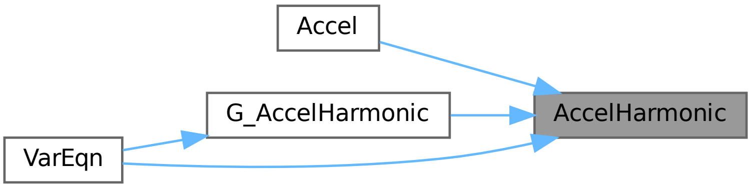
a Acceleration (a=d^2r/dt^2)

Definition at line 12 of file AccelHarmonic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.76 AccelHarmonic.cpp

Go to the documentation of this file.

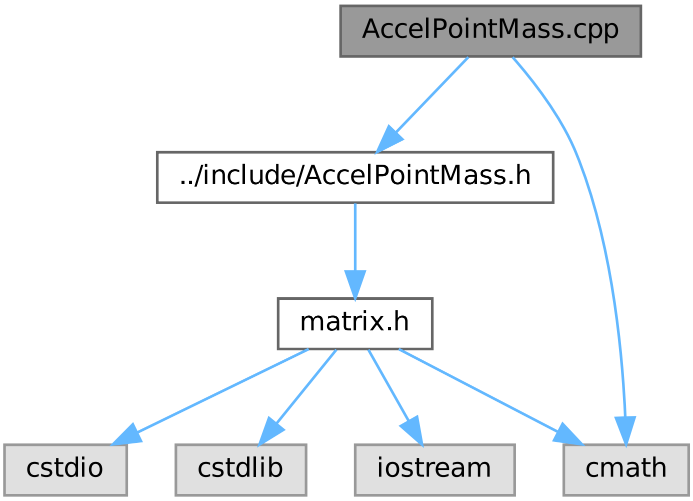
```
#include "../include/AccelHarmonic.h"
#include "../include/Legendre.h"
#include "../include/GLOBAL.h"
#include "../include/SAT_Const.h"
#include <cmath>
Matrix& AccelHarmonic(Matrix r,Matrix E,int n_max,int m_max){
        if(r.n_row<r.n_column){
            r=transpose(r);
        }
        double r_ref,gm,d,latgc,lon,b1,b2,b3,dUdr,dUdlatgc,dUdlon,q3,q2,q1,r2xy,ax,ay,az;
        Matrix a_bf(3);
    r_ref = 6378.1363e3;   // Earth's radius [m]; GGM03S
    gm    = 398600.4415e9; // [m^3/s^2]; GGM03S

    // Body-fixed position
    Matrix r_bf = E * r;

    // Auxiliary quantities
    d = norm(transpose(r_bf));                     // distance

    latgc = asin(r_bf(3)/d);
    lon = atan2(r_bf(2),r_bf(1));
    auto[pnm, dpnm] = Legendre(n_max,m_max,latgc);
    dUdr = 0;
    dUdlatgc = 0;
    dUdlon = 0;
    q3 = 0; q2 = q3; q1 = q2;
    for (int n=0;n<=n_max;n++){
        b1 = (-gm/pow(d,2))*pow((r_ref/d),n)*(n+1);
        b2 =  (gm/d)*pow((r_ref/d),n);
        b3 =  (gm/d)*pow((r_ref/d),n);

        for (int m=0;m<=m_max;m++){
            q1 = q1 + pnm(n+1,m+1)*(Cnm(n+1,m+1)*cos(m*lon)+Snm(n+1,m+1)*sin(m*lon));
            q2 = q2 + dpnm(n+1,m+1)*(Cnm(n+1,m+1)*cos(m*lon)+Snm(n+1,m+1)*sin(m*lon));
            q3 = q3 + m*pnm(n+1,m+1)*(Snm(n+1,m+1)*cos(m*lon)-Cnm(n+1,m+1)*sin(m*lon));
        }
        dUdr     = dUdr     + q1*b1;
        dUdlatgc = dUdlatgc + q2*b2;
        dUdlon   = dUdlon   + q3*b3;
        q3 = 0.0; q2 = q3; q1 = q2;
    }
    // Body-fixed acceleration
    r2xy = pow(r_bf(1),2)+pow(r_bf(2),2);

    ax = (1.0/d*dUdr-r_bf(3)/(pow(d,2)*sqrt(r2xy))*dUdlatgc)*r_bf(1)-(1.0/r2xy*dUdlon)*r_bf(2);
    ay = (1.0/d*dUdr-r_bf(3)/(pow(d,2)*sqrt(r2xy))*dUdlatgc)*r_bf(2)+(1.0/r2xy*dUdlon)*r_bf(1);
    az =  1.0/d*dUdr*r_bf(3)+sqrt(r2xy)/pow(d,2)*dUdlatgc;

    a_bf (1)=ax;
    a_bf (2)=ay ;
    a_bf (3)=az;
    a_bf=transpose(a_bf);
    // Inertial acceleration
    Matrix a = transpose(E)*a_bf;
    return transpose(a);
}
```

5.77 AccelPointMass.cpp File Reference

El archivo contiene las implementaciones de AccelPointMass.h.

```
#include "../include/AccelPointMass.h"
#include <cmath>
```

Include dependency graph for AccelPointMass.cpp:



Functions

- Matrix & AccelPointMass (Matrix &r, Matrix &s, double GM)

5.77.1 Detailed Description

El archivo contiene las implementaciones de AccelPointMass.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AccelPointMass.cpp.

5.77.2 Function Documentation

5.77.2.1 AccelPointMass()

```
Matrix & AccelPointMass (
            Matrix & r,
            Matrix & s,
            double GM)
```

Parameters

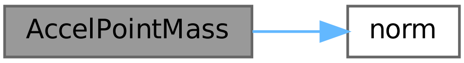
r	Satellite position vector
s	Point mass position vector
GM	Gravitational coefficient of point mass

Returns

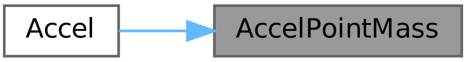
Acceleration (a=d^2r/dt^2)

Definition at line 10 of file AccelPointMass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.78 AccelPointMass.cpp

Go to the documentation of this file.

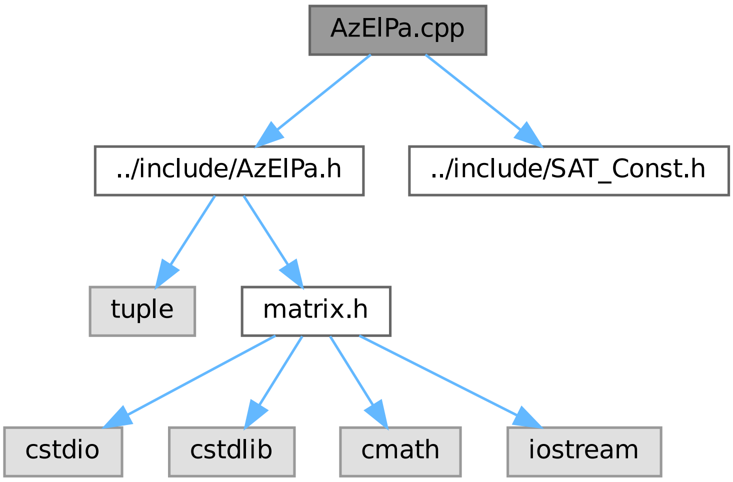
```
#include "../include/AccelPointMass.h"
#include <cmath>

    Matrix&   AccelPointMass(Matrix& r,Matrix& s,double GM){
        Matrix d = r - s;
        Matrix &a= d/pow(norm(d),3) + s/(pow(norm(s),3));
        a=a * -GM;
        return a;
    }
```

5.79 AzElPa.cpp File Reference

El archivo contiene las implementaciones de AzElPa.h.

```
#include "../include/AzElPa.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for AzElPa.cpp:



Functions

- tuple< double, double, Matrix &, Matrix & > AzElPa (Matrix s)

5.79.1 Detailed Description

El archivo contiene las implementaciones de AzElPa.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file AzElPa.cpp.

5.79.2 Function Documentation

5.79.2.1 AzElPa()

```
tuple< double, double, Matrix &, Matrix & > AzElPa (
            Matrix s)
```

Computes azimuth, elevation and partials from local tangent coordinates s

Parameters

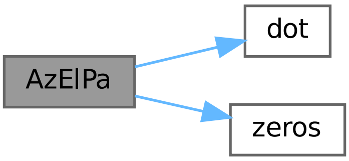
s	Matrix 1x3 Topocentric local tangent coordinates (East-North-Zenith frame)

Returns

tuple<A,E,dAds,dEds> Azimuth [rad],Elevation [rad],Partials of azimuth w.r.t. s,Partials of elevation w.r.t. s

Definition at line 9 of file AzElPa.cpp.

Here is the call graph for this function:



5.80 AzElPa.cpp

Go to the documentation of this file.

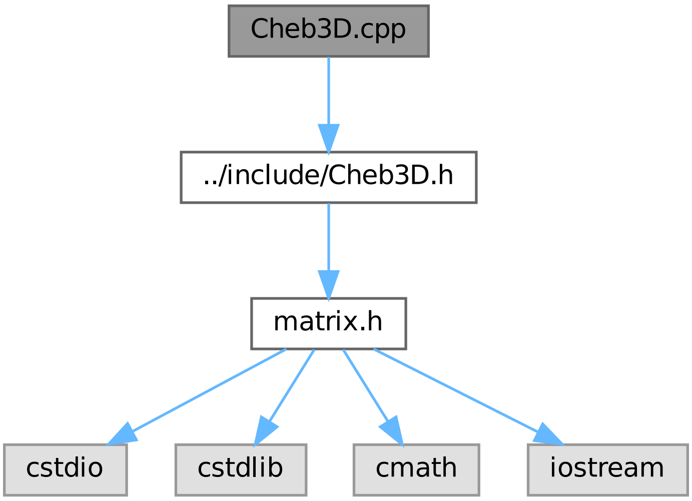
```
00001 #include "../include/AzElPa.h"
00002 #include "../include/SAT_Const.h"
00009     tuple<double,double,Matrix&,Matrix&> AzElPa(Matrix s) {
00010 
00011         long double rho = sqrt(s(1)*s(1)+s(2)*s(2));
00012 
00013         // Angles
00014         double Az = atan2(s(1),s(2));
00015 
00016         if (Az<0.0) {
00017             Az = Az+pi2;
00018         }
00019 
00020         double El = atan ( s(3) / rho );
00021 
00022         // Partials
00023         Matrix &dAds = zeros(3);
00024         dAds(1)=s(2)/(rho*rho);
00025         dAds(2)=-s(1)/(rho*rho);
00026         dAds(3)=0.0 ;
00027         Matrix &dEds= zeros(3);
00028         dEds(1)=-s(1)*s(3)/rho;
00029         dEds(2)=-s(2)*s(3)/rho;
00030         dEds(3)=rho;
00031          dEds= dEds/ dot(s,s);
00032         return tie(Az, El, dAds, dEds);
00033 
00034     }
```

5.81 Cheb3D.cpp File Reference

El archivo contiene las implementaciones de Cheb3D.h.

```
#include "../include/Cheb3D.h"
```

Include dependency graph for Cheb3D.cpp:



Functions

- Matrix & Cheb3D (double t, int N, double Ta, double Tb, Matrix &Cx, Matrix &Cy, Matrix &Cz)

5.81.1 Detailed Description

El archivo contiene las implementaciones de Cheb3D.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Cheb3D.cpp.

5.81.2 Function Documentation

5.81.2.1 Cheb3D()

```
Matrix & Cheb3D (
             double t,
             int N,
             double Ta,
             double Tb,
             Matrix & Cx,
             Matrix & Cy,
             Matrix & Cz)
```

Parameters

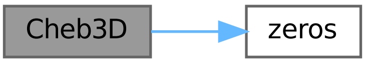
t	time
N	Number of coefficients
Ta	Begin interval
Tb	End interval
Cx	Coefficients of Chebyshev polyomial (x-coordinate)
Cy	Coefficients of Chebyshev polyomial (y-coordinate)
Cz	Coefficients of Chebyshev polyomial (z-coordinate)

Returns

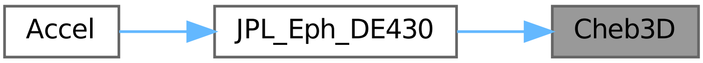
Chebyshev approximation of 3-dimensional vectors

Definition at line 9 of file Cheb3D.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.82 Cheb3D.cpp

Go to the documentation of this file.

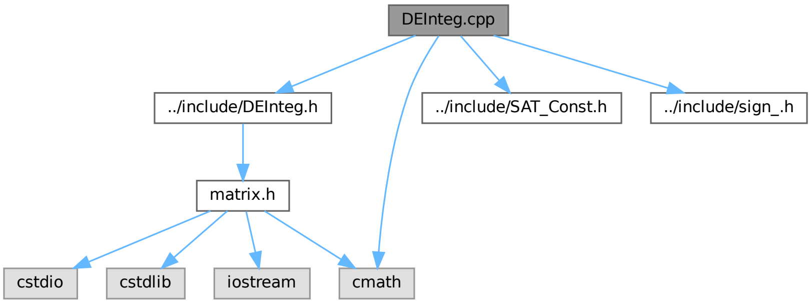
```
#include "../include/Cheb3D.h"

    Matrix& Cheb3D( double t, int N, double Ta, double Tb, Matrix& Cx,Matrix& Cy,Matrix& Cz){

        if ( (t<Ta) || (Tb<t) ){
            cerr«"ERROR: Time out of range in Cheb3D::Value\n";
            exit(EXIT_FAILURE);
            }

        double tau = (2*t-Ta-Tb)/(Tb-Ta);

        Matrix f1 = zeros(1,3);
        Matrix f2 = zeros(1,3);
        Matrix old_f1= zeros(1,3);
        Matrix aux= zeros(1,3);
        for (int i=N;i>=2;i--){
            old_f1 = f1;
            aux(1)=Cx(i);
            aux(2)=Cy(i);
            aux(3)=Cz(i);
            f1 = (f1*(2*tau))-f2+aux;
            f2 = old_f1;
        }
        Matrix *ChebApp=&zeros(1,3);
        aux(1)=Cx(1);
        aux(2)=Cy(1);
        aux(3)=Cz(1);
        (*ChebApp) = f1*tau;
        (*ChebApp)=(*ChebApp)-f2;
        (*ChebApp)=(*ChebApp)+aux;
        return *ChebApp;
    }
```

5.83 DEInteg.cpp File Reference

El archivo contiene las implementaciones de DEInteg.h.

```
#include "../include/DEInteg.h"
#include "../include/SAT_Const.h"
#include "../include/sign_.h"
#include <cmath>
```

Include dependency graph for DEInteg.cpp:



Functions

- Matrix & DEInteg (Matrix &f(double t, Matrix z), double t, double tout, double relerr, double abserr, int n_eqn, Matrix &y)

5.83.1 Detailed Description

El archivo contiene las implementaciones de DEInteg.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file DEInteg.cpp.

5.83.2 Function Documentation

5.83.2.1 DEInteg()

```
Matrix & DEInteg (
            Matrix & fdouble t, Matrix z,
            double t,
            double tout,
            double relerr,
            double abserr,
            int n_eqn,
            Matrix & y)
```

Definition at line 11 of file DEInteg.cpp.

5.84 DEInteg.cpp

Go to the documentation of this file.

```
00001 #include "../include/DEInteg.h"
00002 #include "../include/SAT_Const.h"
00003 #include "../include/sign_.h"
00004 #include <cmath>
00011 Matrix& DEInteg(Matrix& f(double t,Matrix z),double t, double tout,double relerr,double abserr,int
      n_eqn,Matrix &y){
00012   if(y.n_row<y.n_column){
00013     y=transpose(y);
00014   }
00015
00016   Matrix yout=zeros(n_eqn,1),ypout=zeros(n_eqn,1),two(14),gstr(14);
00017   bool
      start=false,phase1=false,nornd=false,crash=false,success=false,PermitTOUT=false,OldPermit=false,stiff=false;
00018   long double
      delsgn=0.0,x=0.0,hi=0.0,ki=0.0,kold=0.0,temp1=0.0,term=0.0,psijm1=0.0,eta=0.0,sum=0.0,absh=0.0,hold=0.0,hnew=0.0,
00019
      k=0.0,round=0.0,gamma=0.0,i=0.0,p5eps=0.0,ifail=0.0,kp1=0.0,kp2=0.0,km1=0.0,km2=0.0,ns=0.0,nsp1=0.0,realns=0.0,im1=0.0,t
00020
      temp3=0.0,reali=0.0,temp4=0.0,nsm2=0.0,limit1=0.0,temp5=0.0,temp6=0.0,limit2=0.0,nsp2=0.0,ip1=0.0,tau=0.0,xold=0.0,erkm2
00021
      erkm1=0.0,erk,err=0.0,knew=0.0,rhi=0.0,h=0.0,erkp1=0.0,rhodouble=0.0,told=0.0,epsilon=0.0,del=0.0,absdel=0.0,tend=0.0,n
00022   releps=0.0,abseps=0.0,twou=0.0,fouru=0.0;
00023   double r=0.0;
00024   int l=0;
00025   twou  = 2*eps;
00026   fouru = 4*eps;
00027
00028
00029   struct DE_STATE_t {
00030     int DE_INIT = 1;
00031     int DE_DONE = 2;
00032     int DE_BADACC = 3;
00033     int DE_NUMSTEPS = 4;
00034     int DE_STIFF = 5;
00035     int DE_INVPARAM = 6;
00036   };
00037
00038   DE_STATE_t DE_STATE;
00039
00040   int State_ = DE_STATE.DE_INIT;
00041 PermitTOUT = true,OldPermit;
00042 told = 0;
00043
00044
00045 double arrtwo[]  = {1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 128.0,256.0, 512.0, 1024.0, 2048.0, 4096.0,
      8192.0};
00046 double arrgstr[] = {1.0, 0.5, 0.0833, 0.0417, 0.0264, 0.0188, 0.0143, 0.0114, 0.00936, 0.00789,
      0.00679,0.00592, 0.00524, 0.00468};
00047 for(int wxy=1;wxy<=14;wxy++){
00048   two(wxy)=arrtwo[wxy-1];
00049   gstr(wxy)=arrgstr[wxy-1];
00050 }
00051
00052 Matrix yy     = zeros(n_eqn,1);
00053 Matrix wt     = zeros(n_eqn,1);
```

```
Matrix p     = zeros(n_eqn,1);
Matrix yp    = zeros(n_eqn,1);
Matrix phi   = zeros(n_eqn,17);
Matrix g     = zeros(14,1);
Matrix sig   = zeros(14,1);
Matrix rho   = zeros(14,1);
Matrix w     = zeros(13,1);
Matrix alpha = zeros(13,1);
Matrix beta  = zeros(13,1);
Matrix v     = zeros(13,1);
Matrix psi_  = zeros(13,1);

if (t==tout) {
  return y;}

  epsilon = fmax(relerr,abserr);
  if ( ( relerr <  0.0  ) || ( abserr <  0.0 ) ||  ( epsilon    <= 0.0  ) ||
    ( State_  >  DE_STATE.DE_INVPARAM ) ||  ( (State_ != DE_STATE.DE_INIT) &&  (t != told)            )
)
  {
     State_ = DE_STATE.DE_INVPARAM;
       return y;
  }

  del    = tout - t;
  absdel = fabs(del);

  tend   = t + 100.0*del;
  if (!PermitTOUT){
    tend = tout;
  }
  nostep = 0;
  kle4   = 0;
  stiff  = false;
  releps = relerr/epsilon;
  abseps = abserr/epsilon;

  if  ( (State_==DE_STATE.DE_INIT) || (!OldPermit) || (delsgn*del<=0.0) ){


    start  = true;
    x      = t;
    yy     = y;
    delsgn = sign_(1.0, del);
    h = sign_( fmax(fouru*fabs(x), fabs(tout-x)), tout-x );}
while(true){
  if (fabs(x-t) >= absdel){
    yout   = zeros(n_eqn,1);
    ypout  = zeros(n_eqn,1);
    g(2)   = 1.0;
    rho(2) = 1.0;
    hi = tout - x;
    ki = kold + 1;

    for (int i=1;i<=k;i++){
      temp1 = i;
      w(i+1) = 1.0/temp1;}

      term = 0.0;
      for (int j=2;j<=ki;j++){
        psijm1 = psi_(j);
        gamma = (hi + term)/psijm1;
        eta = hi/psijm1;
        for (int i=1;i<=ki+1-j;i++){
          w(i+1) = gamma*w(i+1) - eta*w(i+2);}
          g(j+1) = w(2);
          rho(j+1) = gamma*rho(j);
          term = psijm1;}

        if(yout.n_row>yout.n_column){
        yout=transpose(yout);}
        if(ypout.n_row>ypout.n_column){
        ypout=transpose(ypout);}
        if(y.n_row>y.n_column){
        y=transpose(y);
         }
          for (int j=1;j<=ki;j++){
            i = ki+1-j;
            yout   = yout   + extract_column(phi,i+1)*g(i+1);
            ypout = ypout + extract_column(phi,i+1)*rho(i+1);
          }
            yout = y +yout*hi;
            y    = yout;
      State_    = DE_STATE.DE_DONE;
      t         = tout;
      told      = t;
      OldPermit = PermitTOUT;
      return y;
    }

    if ( !PermitTOUT && ( fabs(tout-x) < fouru*fabs(x) ) ){
      h = tout - x;
      yp = f(x,yy);
      y = yy + yp*h;
      State_    = DE_STATE.DE_DONE;
      t         = tout;
      told      = t;
      OldPermit = PermitTOUT;
      return y;
    }

    h  = sign_(min(fabs(h), fabs(tend-x)), h);
    for (l=1;l<=n_eqn;l++){
      wt(l) = releps*fabs(yy(l)) + abseps;
    }

    if (fabs(h) < fouru*fabs(x)){
      h = sign_(fouru*fabs(x),h);
      crash = true;
      return y;
  }

  p5eps  = 0.5*epsilon;
  crash  = false;
  g(2)   = 1.0;
  g(3)   = 0.5;
  sig(2) = 1.0;

  ifail = 0;

  round = 0.0;
  for (l=1;l<=n_eqn;l++){
    round = round + (y(l)*y(l))/(wt(l)*wt(l));
  }
  round = twou*sqrt(round);
  if (p5eps<round){
    epsilon = 2.0*round*(1.0+fouru);
    crash = true;
    return y;
  }
  if (start){

    yp = transpose(f(x,y));
    sum = 0.0;
    for (l=1;l<=n_eqn;l++){
      phi(l,2) = yp(l);
      phi(l,3) = 0.0;
      sum = sum + (yp(l)*yp(l))/(wt(l)*wt(l));
    }
    sum  = sqrt(sum);
    absh = fabs(h);
    if (epsilon<16.0*sum*h*h){
      absh=0.25*sqrt(epsilon/sum);
    }
    h    = sign_(fmax(absh, fouru*fabs(x)), h);
    hold = 0.0;
    hnew = 0.0;
    k    = 1;
    kold = 0;
    start  = false;
    phase1 = true;
    nornd  = true;
    if (p5eps<=100.0*round){
      nornd = false;
      for (l=1;l<=n_eqn;l++){
        phi(l,16)=0.0;
      }
    }
  }

  while(true){

    kp1 = k+1;
    kp2 = k+2;
    km1 = k-1;
    km2 = k-2;

    if (h !=hold){
      ns=0;
    }
    if (ns<=kold){
      ns=ns+1;
    }
    nsp1 = ns+1;

    if (k>=ns){
      beta(ns+1) = 1.0;
      realns = ns;
      alpha(ns+1) = 1.0/realns;
      temp1 = h*realns;
      sig(nsp1+1) = 1.0;
      if (k>=nsp1){
        for (int i=nsp1;i<=k;i++){
          im1   = i-1;
          temp2 = psi_(im1+1);
          psi_(im1+1) = temp1;
          beta(i+1)  = beta(im1+1)*psi_(im1+1)/temp2;
          temp1    = temp2 + h;
          alpha(i+1) = h/temp1;
          reali = i;
          sig(i+2) = reali*alpha(i+1)*sig(i+1);
        }
      }
      psi_(k+1) = temp1;


      if (ns>1){

        if (k>kold){
          temp4 = k*kp1;
          v(k+1) = 1.0/temp4;
          nsm2 = ns-2;
          for (int j=1;j<=nsm2;j++){
            i = k-j;
            v(i+1) = v(i+1) - alpha(j+2)*v(i+2);
          }
        }


        limit1 = kp1 - ns;
        temp5  = alpha(ns+1);
        for (int iq=1;iq<=limit1;iq++){
          v(iq+1) = v(iq+1) - temp5*v(iq+2);
          w(iq+1) = v(iq+1);
        }
        g(nsp1+1) = w(2);}
        else{
          for (int iq=1;iq<=k;iq++){
            temp3 = iq*(iq+1);
            v(iq+1) = 1.0/temp3;
            w(iq+1) = v(iq+1);
          }
        }


        nsp2 = ns + 2;
        if (kp1>=nsp2){
          for (int i=nsp2;i<=kp1;i++){
            limit2 = kp2 - i;
            temp6  = alpha(i);
            for (int iq=1;iq<=limit2;iq++){
              w(iq+1) = w(iq+1) - temp6*w(iq+2);
            }
            g(i+1) = w(2);
          }
        }
  }


  if (k>=nsp1){
    for (int i=nsp1;i<=k;i++){
      temp1 = beta(i+1);
      for (l=1;l<=n_eqn;l++){
        phi(l,i+1) = temp1 * phi(l,i+1);
      }
    }
  }

  for (l=1;l<=n_eqn;l++){
    phi(l,kp2+1) = phi(l,kp1+1);
    phi(l,kp1+1) = 0.0;
    p(l)       = 0.0;
  }
  for (int j=1;j<=k;j++){
    i      = kp1 - j;
    ip1   = i+1;
    temp2 = g(i+1);
    for (l=1;l<=n_eqn;l++){
      p(l)      = p(l) + temp2*phi(l,i+1);
      phi(l,i+1) = phi(l,i+1) + phi(l,ip1+1);
    }
```

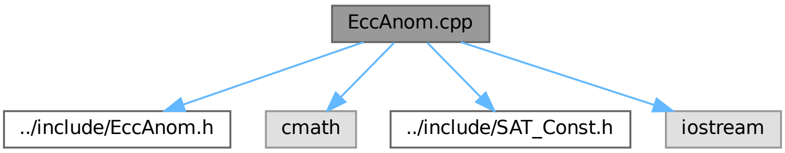
```
}
if (nornd){
 p = y + p*h;}
 else{
  for (l=1;l<=n_eqn;l++){
    tau = h*p(l) - phi(l,16);
    p(l) = y(l) + tau;
    phi(l,17) = (p(l) - y(l)) - tau;
  }
}
xold = x;
x = x + h;
absh = fabs(h);
yp = f(x,p);

erkm2 = 0.0;
erkm1 = 0.0;
erk = 0.0;
for (l=1;l<=n_eqn;l++){
  temp3 = 1.0/wt(l);
  temp4 = yp(l) - phi(l,1+1);
  if (km2> 0){
    erkm2 = erkm2 + ((phi(l,km1+1)+temp4)*temp3)*((phi(l,km1+1)+temp4)*temp3);
  }
  if (km2>=0){
    erkm1 = erkm1 + ((phi(l,k+1)+temp4)*temp3)*((phi(l,k+1)+temp4)*temp3);
  }
  erk = erk + (temp4*temp3)*(temp4*temp3);
}

if (km2> 0){
  erkm2 = absh*sig(km1+1)*gstr(km2+1)*sqrt(erkm2);
}
if (km2>=0){
  erkm1 = absh*sig(k+1)*gstr(km1+1)*sqrt(erkm1);
}
temp5 = absh*sqrt(erk);
err = temp5*(g(k+1)-g(kp1+1));
erk = temp5*sig(kp1+1)*gstr(k+1);
knew = k;


if (km2 >0){
  if (fmax(erkm1,erkm2)<=erk){
    knew=km1;
  }
}
if (km2==0){
  if (erkm1<=0.5*erk){
    knew=km1;
  }
}

success = (err<=epsilon);

if (!success){
  phase1 = false;
  x = xold;
  for (int i=1;i<=k;i++){
    temp1 = 1.0/beta(i+1);
    ip1 = i+1;
    for (l=1;l<=n_eqn;l++){
      phi(l,i+1)=temp1*(phi(l,i+1)-phi(l,ip1+1));
    }
  }

  if (k>=2){
    for (int i=2;i<=k;i++){
      psi_(i) = h-psi_(i+1);
    }
  }



  ifail = ifail+1;
  temp2 = 0.5;
  if (ifail>3) {
    if (p5eps < 0.25*erk){
      temp2 = sqrt(p5eps/erk);
    }
  }
  if (ifail>=3){
    knew = 1;
  }
  h = temp2*h;
  k = knew;
  if (fabs(h)<fouru*fabs(x)){
      crash = true;
      h = sign_(fouru*fabs(x), h);
      epsilon = epsilon*2.0;
        return y;
      }
  }

  if (success){
    break;
  }

}

kold = k;
hold = h;

temp1 = h*g(kp1+1);
if (nornd){
  for (l=1;l<=n_eqn;l++){
    y(l) = p(l) + temp1*(yp(l) - phi(l,2));
  }
}
else{
  for (l=1;l<=n_eqn;l++){
    rhodouble = temp1*(yp(l) - phi(l,2)) - phi(l,17);
    y(l) = rhi + p(l);
    phi(l,16) = (y(l) - p(l)) - rhodouble;
  }
}
yp = f(x,y);


for (l=1;l<=n_eqn;l++){
  phi(l,kp1+1) = yp(l) - phi(l,2);
  phi(l,kp2+1) = phi(l,kp1+1) - phi(l,kp2+1);
}
for (int i=1;i<=k;i++){
  for (l=1;l<=n_eqn;l++){
    phi(l,i+1) = phi(l,i+1) + phi(l,kp1+1);
  }
}


erkp1 = 0.0;
if ( (knew==km1) || (k==12) ){
  phase1 = false;
}

if (phase1){
  k = kp1;
  erk = erkp1;}
  else{
    if (knew==km1){

      k = km1;
      erk = erkm1;}
      else{
        if (kp1<=ns){
          for (l=1;l<=n_eqn;l++){
            erkp1 = erkp1 + (phi(l,kp2+1)/wt(l))*(phi(l,kp2+1)/wt(l));
          }
          erkp1 = absh*gstr(kp1+1)*sqrt(erkp1);


          if (k>1){
            if ( erkm1<=min(erk,erkp1)){

              k=km1; erk=erkm1;}
              else{
                if ( (erkp1<erk) && (k!=12) ){

                  k=kp1;
                  erkp1;
                }
              }
            }
            else if (erkp1<0.5*erk){

              k = kp1;
              erk = erkp1;
            }
        }
    }
}


if ( phase1 || (p5eps>=erk*two(k+2)) ){
  hnew = 2.0*h;}
  else{
    if (p5eps<erk){
      temp2 = k+1;
      r = pow(p5eps/erk,(1.0/temp2));
      hnew = absh*fmax(0.5, min(0.9,r));
      hnew = sign_(fmax(hnew, fouru*fabs(x)), h);}
      else{
        hnew = h;
      }
    }
    h = hnew;

    if (crash){
      State_     = DE_STATE.DE_BADACC;
      relerr     = epsilon*releps;
      abserr     = epsilon*abseps;
      y          = yy;
      t          = x;
      told       = t;
      OldPermit = true;
      return y;
    }

  nostep = nostep+1;



  kle4 = kle4+1;
  if (kold>  4){
    kle4 = 0;
  }
  if (kle4>=50){
    stiff = true;
  }
}
/*
cout « "delsgn: " « delsgn « endl;
cout « "x: " « x « endl;
cout « "hi: " « hi « endl;
cout « "ki: " « ki « endl;
cout « "kold: " « kold « endl;
cout « "temp1: " « temp1 « endl;
cout « "term: " « term « endl;
cout « "psijm1: " « psijm1 « endl;
cout « "eta: " « eta « endl;
cout « "sum: " « sum « endl;
cout « "absh: " « absh « endl;
cout « "hold: " « hold « endl;
cout « "hnew: " « hnew « endl;
cout « "k: " « k « endl;
cout « "round: " « round « endl;
cout « "gamma: " « gamma « endl;
cout « "i: " « i « endl;
cout « "p5eps: " « p5eps « endl;
cout « "ifail: " « ifail « endl;
cout « "kp1: " « kp1 « endl;
cout « "kp2: " « kp2 « endl;
cout « "km1: " « km1 « endl;
cout « "km2: " « km2 « endl;
cout « "ns: " « ns « endl;
cout « "nsp1: " « nsp1 « endl;
cout « "realns: " « realns « endl;
cout « "im1: " « im1 « endl;
cout « "temp2: " « temp2 « endl;
cout « "temp3: " « temp3 « endl;
cout « "reali: " « reali « endl;
cout « "temp4: " « temp4 « endl;
cout « "nsm2: " « nsm2 « endl;
cout « "limit1: " « limit1 « endl;
cout « "temp5: " « temp5 « endl;
cout « "temp6: " « temp6 « endl;
cout « "limit2: " « limit2 « endl;
cout « "nsp2: " « nsp2 « endl;
cout « "ip1: " « ip1 « endl;
cout « "tau: " « tau « endl;
cout « "xold: " « xold « endl;
cout « "erkm2: " « erkm2 « endl;
cout « "erkm1: " « erkm1 « endl;
cout « "erk: " « erk « endl;
cout « "err: " « err « endl;
cout « "knew: " « knew « endl;
cout « "rhi: " « rhi « endl;
cout « "h: " « h « endl;
cout « "r: " « r « endl;
cout « "erkp1: " « erkp1 « endl;
cout « "rhodouble: " « rhodouble « endl;
cout « "told: " « told « endl;
cout « "epsilon: " « epsilon « endl;
cout « "del: " « del « endl;
cout « "absdel: " « absdel « endl;
cout « "tend: " « tend « endl;
cout « "nostep: " « nostep « endl;
cout « "kle4: " « kle4 « endl;
cout « "releps: " « releps « endl;
cout « "abseps: " « abseps « endl;
cout « "twou: " « twou « endl;
cout « "fouru: " « fouru « endl;

cout « "l: " « l « endl;
cout « "y: \n" « y « endl;*/

  cout«69«endl;
return y;
}
```

5.85 EccAnom.cpp File Reference

El archivo contiene las implementaciones de EccAnom.h.

```
#include "../include/EccAnom.h"
#include <cmath>
#include "../include/SAT_Const.h"
#include <iostream>
```

Include dependency graph for EccAnom.cpp:



Functions

- double EccAnom (double M, double e)

5.85.1 Detailed Description

El archivo contiene las implementaciones de EccAnom.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file EccAnom.cpp.

5.85.2 Function Documentation

5.85.2.1 EccAnom()

```
double EccAnom (
              double M,
              double e)
```

Computes the eccentric anomaly for elliptic orbits

Parameters

M	Mean anomaly in [rad]
e	Eccentricity of the orbit [0,1]

Returns

double resultado

Definition at line 12 of file EccAnom.cpp.

5.86 EccAnom.cpp

Go to the documentation of this file.

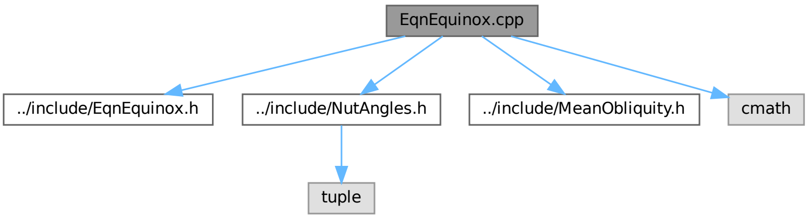
```
00001 #include "../include/EccAnom.h"
00002 #include <cmath>
00003 #include "../include/SAT_Const.h"
00004 #include <iostream>
00005 using namespace std;
00012     double EccAnom (double M,double e){
00013 
00014     double maxit = 15;
00015     int i = 1;
00016 
00017     M = fmod(M, 2.0∗pi);
00018     double E;
00019 
00020     if (e<0.8){
00021         E = M; }
00022     else{
00023         E = pi;}
00024 
00025     double f = E - e∗sin(E) - M;
00026     E = E - f / ( 1.0 - e∗cos(E) );
00027 
00028     while (abs(f) > 1e2∗eps)    {
00029         f = E - e∗sin(E) - M;
00030         E = E - f / ( 1.0 - e∗cos(E) );
00031         i = i+1;
00032         if (i==maxit){
00033             cerr« "convergence problems in EccAnom";
00034         }
00035     }
00036     return E;
00037     }
```

5.87 EqnEquinox.cpp File Reference

El archivo contiene las implementaciones de EqnEquinox.h.

```
#include "../include/EqnEquinox.h"
#include "../include/NutAngles.h"
#include "../include/MeanObliquity.h"
#include <cmath>
```

Include dependency graph for EqnEquinox.cpp:



Functions

- double EqnEquinox (double Mjd_TT)

5.87.1 Detailed Description

El archivo contiene las implementaciones de EqnEquinox.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file EqnEquinox.cpp.

5.87.2 Function Documentation

5.87.2.1 EqnEquinox()

```
double EqnEquinox (
            double Mjd_TT)
```

Computation of the equation of the equinoxes

Parameters

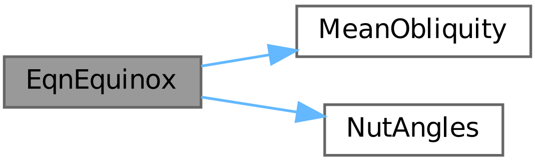
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

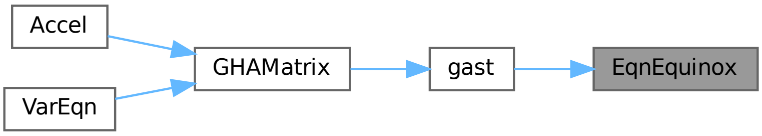
double Equation of the equinoxes

Definition at line 13 of file EqnEquinox.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.88 EqnEquinox.cpp

Go to the documentation of this file.

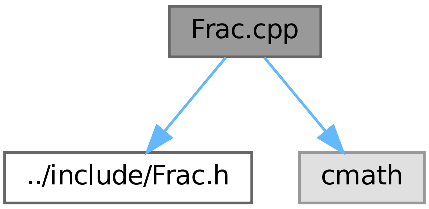
```
#include "../include/EqnEquinox.h"
#include "../include/NutAngles.h"
#include "../include/MeanObliquity.h"
#include <cmath>


    double EqnEquinox (double Mjd_TT){

    auto [dpsi, deps] = NutAngles (Mjd_TT);

    return dpsi * cos ( MeanObliquity(Mjd_TT) );
}

```

5.89 Frac.cpp File Reference

El archivo contiene las implementaciones de Frac.h.

```
#include "../include/Frac.h"
#include <cmath>
```

Include dependency graph for Frac.cpp:



Functions

- double Frac (double x)

5.89.1 Detailed Description

El archivo contiene las implementaciones de Frac.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Frac.cpp.

5.89.2 Function Documentation

5.89.2.1 Frac()

```
double Frac (
              double x)
```

Parameters

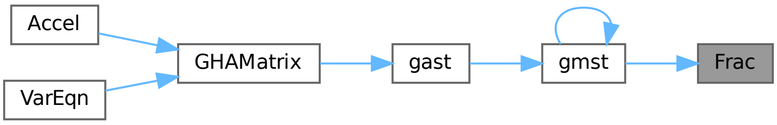
x	double

Returns

double parte fraccion de x

Definition at line 10 of file Frac.cpp.

Here is the caller graph for this function:



5.90 Frac.cpp

Go to the documentation of this file.

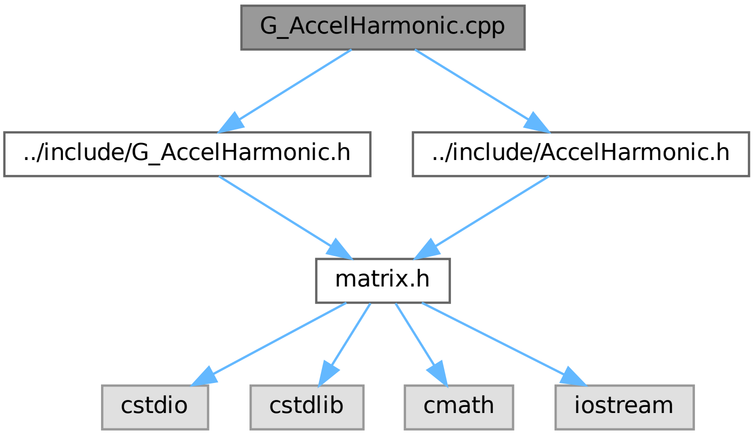
```
00001 #include "../include/Frac.h"
00002 #include <cmath>
00003
00010     double Frac(double x){
00011
00012     return x-floor(x);
00013     }
```

5.91 G_AccelHarmonic.cpp File Reference

El archivo contiene las implementaciones de G_AccelHarmonic.h.

```
#include "../include/G_AccelHarmonic.h"
#include "../include/AccelHarmonic.h"
```

Include dependency graph for G_AccelHarmonic.cpp:



Functions

- Matrix & G_AccelHarmonic (Matrix r, Matrix U, int n_max, int m_max)

5.91.1 Detailed Description

El archivo contiene las implementaciones de G_AccelHarmonic.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file G_AccelHarmonic.cpp.

5.91.2 Function Documentation

5.91.2.1 G_AccelHarmonic()

```
Matrix & G_AccelHarmonic (
            Matrix r,
            Matrix U,
            int n_max,
            int m_max)
```

Parameters

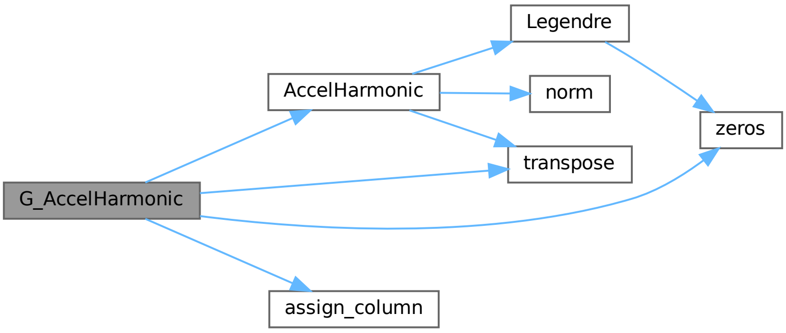
r	Satellite position vector in the true-of-date system
U	Transformation matrix to body-fixed syste
n	Gravity model degree
m	Gravity model order

Returns

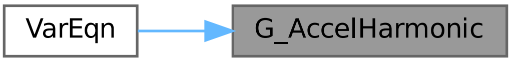
G Gradient (G=da/dr) in the true-of-date system

Definition at line 10 of file G_AccelHarmonic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.92 G_AccelHarmonic.cpp

Go to the documentation of this file.

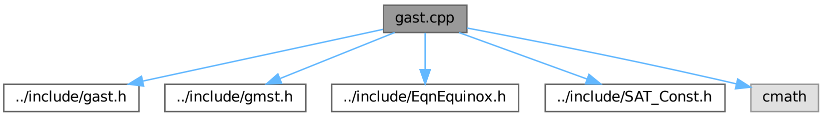
```
#include "../include/G_AccelHarmonic.h"
#include "../include/AccelHarmonic.h"

    Matrix& G_AccelHarmonic( Matrix r,Matrix U, int n_max, int m_max ){

            if(r.n_row<r.n_column){
                r=transpose(r);
            }
        Matrix da;
    double d = 1.0;

    Matrix &G = zeros(3,3);
    Matrix dr = zeros(3,1);

    for (int i=1;i<=3;i++){
            dr = zeros(3,1);
            dr(i) = d/2;

            da = AccelHarmonic ( r+dr,U, n_max, m_max ) -
            AccelHarmonic ( r-dr,U, n_max, m_max );
            G=assign_column(G,da/d,i) ;
               }
    return G;
    }
```

5.93 gast.cpp File Reference

El archivo contiene las implementaciones de gast.h.

```
#include "../include/gast.h"
#include "../include/gmst.h"
#include "../include/EqnEquinox.h"
#include "../include/SAT_Const.h"
#include <cmath>
```

Include dependency graph for gast.cpp:



Functions

- double gast (double Mjd_UT1)

5.93.1 Detailed Description

El archivo contiene las implementaciones de gast.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file gast.cpp.

5.93.2 Function Documentation

5.93.2.1 gast()

```
double gast (
            double Mjd_UT1)
```

Greenwich Apparent Sidereal Time

Parameters

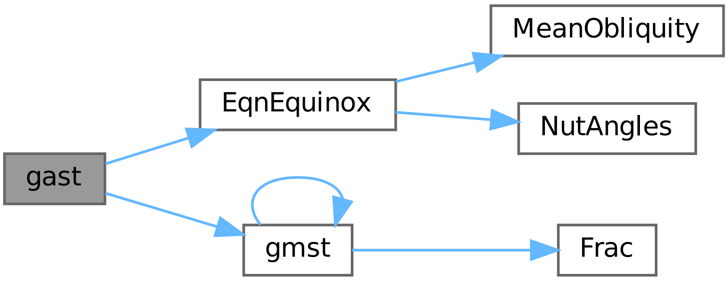
Mjd_UT1	Modified Julian Date UT1

Returns

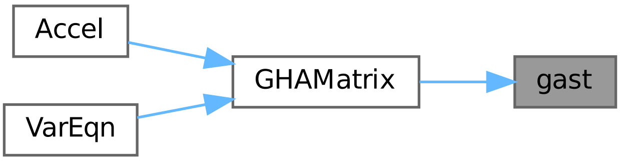
gstime GAST in [rad]

Definition at line 14 of file gast.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.94 gast.cpp

Go to the documentation of this file.

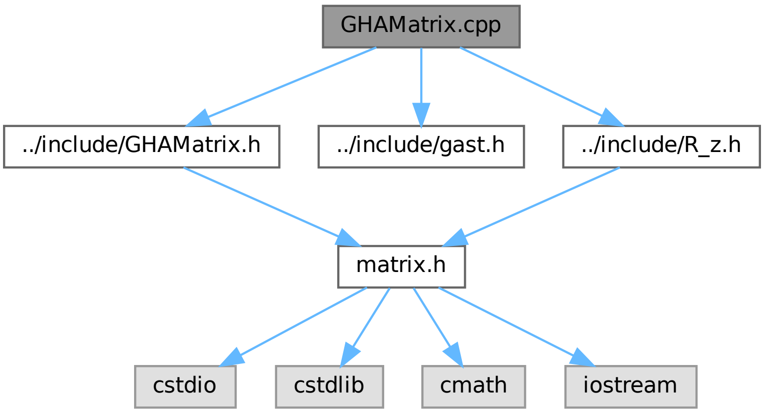
```
#include "../include/gast.h"
#include "../include/gmst.h"
#include "../include/EqnEquinox.h"
#include "../include/SAT_Const.h"
#include <cmath>


    double gast(double Mjd_UT1){
        return fmod(gmst(Mjd_UT1) + EqnEquinox(Mjd_UT1), 2*pi );
    }
```

5.95 GHAMatrix.cpp File Reference

El archivo contiene las implementaciones de GHAMatrix.h.

```
#include "../include/GHAMatrix.h"
#include "../include/gast.h"
#include "../include/R_z.h"
```

Include dependency graph for GHAMatrix.cpp:



Functions

- Matrix & GHAMatrix (double Mjd_UT1)

5.95.1 Detailed Description

El archivo contiene las implementaciones de GHAMatrix.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file GHAMatrix.cpp.

5.95.2 Function Documentation

5.95.2.1 GHAMatrix()

```
Matrix & GHAMatrix (
            double Mjd_UT1)
```

Transformation from true equator and equinox to Earth equator and Greenwich meridian system

Parameters

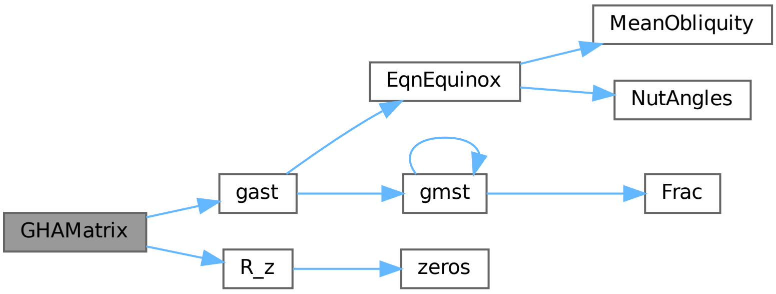
Mjd_UT1	Modified Julian Date UT1

Returns

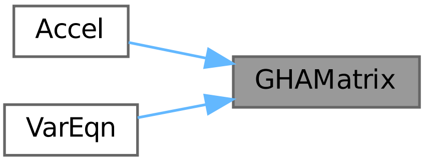
GHAmat Greenwich Hour Angle matrix

Definition at line 11 of file GHAMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.96 GHAMatrix.cpp

Go to the documentation of this file.

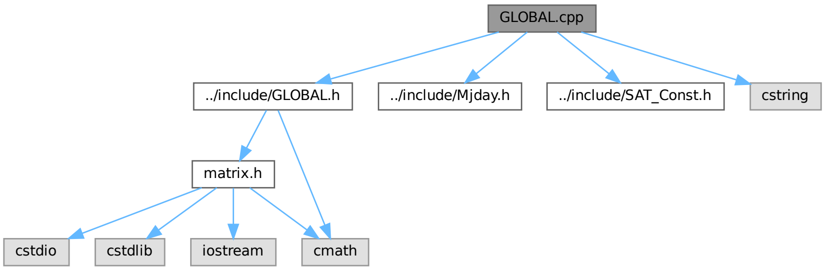
```
#include "../include/GHAMatrix.h"
#include "../include/gast.h"
#include "../include/R_z.h"

    Matrix& GHAMatrix (double Mjd_UT1){
        return  R_z( gast(Mjd_UT1) );
    }
```

5.97 GLOBAL.cpp File Reference

El archivo contiene las implementaciones de GLOBAL.h.

```
#include "../include/GLOBAL.h"
#include "../include/Mjday.h"
#include "../include/SAT_Const.h"
#include <cstring>
```

Include dependency graph for GLOBAL.cpp:



Functions

- void AuxParamLoad ()
- void eop19620101 (int c)
- void GGM03S (int n)
- void DE430Coeff (int row, int column)
- void GEOS3 (int nobs)

Variables

- Param AuxParam
- Matrix eopdata
- Matrix Cnm
- Matrix Snm
- Matrix PC
- Matrix obs

5.97.1 Detailed Description

El archivo contiene las implementaciones de GLOBAL.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file GLOBAL.cpp.

5.97.2 Function Documentation

5.97.2.1 AuxParamLoad()

```
void AuxParamLoad ()
```

Carga AuxParam

Definition at line 17 of file GLOBAL.cpp.

5.97.2.2 DE430Coeff()

```
void DE430Coeff (
             int row = 2285,
             int column = 1020)
```

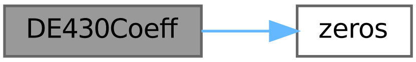
Lee el archivo DE430Coeff.txt y recoge cada fila y lo asigna a PC

Parameters

row	número de filas a recoger
column	número de columnas a recoger

Definition at line 68 of file GLOBAL.cpp.

Here is the call graph for this function:



5.97.2.3 eop19620101()

```
void eop19620101 (
            int c = 21413)
```

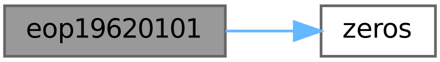
Lee el archivo eop19620101.txt y recoge cada fila y lo asigna a eopdata

Parameters

c	número de filas a recoger

Definition at line 26 of file GLOBAL.cpp.

Here is the call graph for this function:



5.97.2.4 GEOS3()

```
void GEOS3 (
            int nobs = 46)
```

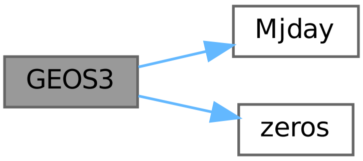
Lee el archivo GEOS3.txt y recoge cada fila y lo asigna a obs

Parameters

nobs	número de filas a recoger

Definition at line 87 of file GLOBAL.cpp.

Here is the call graph for this function:



5.97.2.5 GGM03S()

```
void GGM03S (
             int n = 181)
```

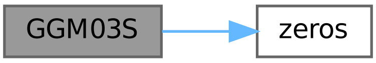
Lee el archivo GGM03S.txt y recoge cada fila y lo asigna a Cnm y Snm

Parameters

c	dimension de la matriz

Definition at line 47 of file GLOBAL.cpp.

Here is the call graph for this function:



5.97.3 Variable Documentation

5.97.3.1 AuxParam

```
Param AuxParam
```

Definition at line 11 of file GLOBAL.cpp.

5.97.3.2 Cnm

```
Matrix Cnm
```

Definition at line 13 of file GLOBAL.cpp.

5.97.3.3 eopdata

```
Matrix eopdata
```

Definition at line 12 of file GLOBAL.cpp.

5.97.3.4 obs

```
Matrix obs
```

Definition at line 16 of file GLOBAL.cpp.

5.97.3.5 PC

```
Matrix PC
```

Definition at line 15 of file GLOBAL.cpp.

5.97.3.6 Snm

```
Matrix Snm
```

Definition at line 14 of file GLOBAL.cpp.

5.98 GLOBAL.cpp

Go to the documentation of this file.

```
00001 #include "../include/GLOBAL.h"
00002 #include "../include/Mjday.h"
00003 #include "../include/SAT_Const.h"
00004 #include <cstring>
00011 Param AuxParam;
00012 Matrix eopdata;
00013 Matrix Cnm;
00014 Matrix Snm;
00015 Matrix PC;
00016 Matrix obs;
00017 void AuxParamLoad(){
00018     AuxParam.Mjd_UTC=4.974611635416653e+04;
00019     AuxParam.Mjd_TT=4.974611706231468e+04;
00020     AuxParam.n=20;
00021     AuxParam.m=20;
00022     AuxParam.sun=1;
00023     AuxParam.moon=1;
00024     AuxParam.planets=1;
00025 }
00026 void eop19620101(int c){
00027     eopdata=zeros(13,c);
00028 
00029     FILE *fid = fopen("../data/eop19620101.txt","r");
00030     if(fid==NULL){
00031         cout « "Fail open eop19620101.txt file \n";
00032         perror("Error");
00033         exit(EXIT_FAILURE);
00034     }
00035     for (int j=1;j<=c;j++){
00036         fscanf(fid,"%lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf %lf",
00037             &(eopdata(1,j)),&(eopdata (2,j)),&(eopdata (3,j)),
00038             &(eopdata (4,j)),&(eopdata (5,j)),&(eopdata (6,j)),
00039             &(eopdata (7,j)),&(eopdata (8,j)),&(eopdata (9,j)),
00040             &(eopdata (10,j)),&(eopdata (11,j)),&(eopdata (12,j)),
00041             &(eopdata (13,j))
00042             );
00043     }
00044     fclose(fid);
00045 }
00046 
00047 void GGM03S(int n){
00048     Cnm=zeros(n,n);
00049     Snm=zeros(n,n);
00050     FILE *fid = fopen("../data/GGM03S.txt","r");
00051     if(fid==NULL){
00052         cout « "Fail open GGM03S.txt file \n";
00053         perror("Error");
00054         exit(EXIT_FAILURE);
00055     }
00056     double aux;
00057     for(int i=1;i<=n;i++){
00058         for (int j=1;j<=i;j++){
00059             fscanf(fid,"%lf %lf %lf %lf %lf %lf",
00060                 &aux,&aux,
00061                 &Cnm(i,j),&Snm(i,j),
00062                 &aux,&aux
00063                 );
```

```
        }
    }
    fclose(fid);
}
void DE430Coeff(int row,int column){
    PC=zeros(row,column);
    FILE *fid = fopen("../data/DE430Coeff.txt","r");
    if(fid==NULL){
        cout « "Fail open DE430Coeff.txt file \n";
        perror("Error");
        exit(EXIT_FAILURE);
    }
    double aux;
    for(int i=1;i<=row;i++){
        for (int j=1;j<=column;j++){
            fscanf(fid,"%lf",
                &PC(i,j)
                );
        }
    }
    fclose(fid);
}

void GEOS3(int nobs){
    obs=zeros(nobs,4);
    FILE *fid = fopen("../data/GEOS3.txt","r");
    if(fid==NULL){
        cout « "Fail open GEOS3.txt file \n";
        perror("Error");
        exit(EXIT_FAILURE);
    }
    int Y,MO,D,H,M,MI,S;
    double AZ,EL,DIST;
    char tline[57],y[5],mo[3],d[3],h[3],mi[3],s[6],az[10],el[9],dist[10],aux[2];
    for (int i=1;i<=nobs;i++)
    {
        fgets(tline,sizeof(tline),fid);
        strncpy(y,&(tline[0]),4);
        y[4]='\0';
        Y=atoi(y);
        strncpy(mo,&(tline[5]),2);
        mo[2]='\0';
        MO=atoi(mo);
        strncpy(d,&(tline[8]),2);
        d[2]='\0';
        D=atoi(d);
        strncpy(h,&(tline[12]),2);
        h[2]='\0';
        H=atoi(h);
        strncpy(mi,&(tline[15]),2);
        mi[2]='\0';
        MI=atoi(mi);
        strncpy(s,&(tline[18]),5);
        s[5]='\0';
        S=atof(s);
        strncpy(az,&(tline[25]),9);
        az[9]='\0';
        AZ=atof(az);
        strncpy(el,&(tline[35]),8);
        el[8]='\0';
        EL=atof(el);
        strncpy(dist,&(tline[44]),9);
        dist[9]='\0';
        DIST=atof(dist);
        obs(i,1) = Mjday(Y,MO,D,H,MI,S);
        obs(i,2) = Rad*AZ;
        obs(i,3) = Rad*EL;
        obs(i,4) = 1e3*DIST;
    }
    fclose(fid);
}
```

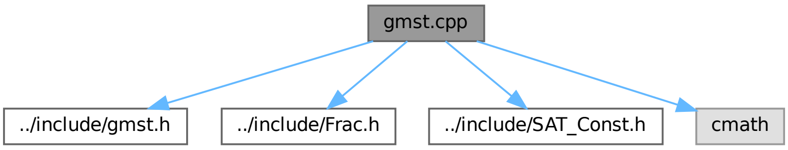
5.99 gmst.cpp File Reference

El archivo contiene las implementaciones de gmst.h.

```
#include "../include/gmst.h"
#include "../include/Frac.h"
#include "../include/SAT_Const.h"
```

```
#include <cmath>
```

Include dependency graph for gmst.cpp:



Functions

- double gmst (double Mjd_UT1)

5.99.1 Detailed Description

El archivo contiene las implementaciones de gmst.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file gmst.cpp.

5.99.2 Function Documentation

5.99.2.1 gmst()

```
double gmst (
            double Mjd_UT1)
```

Greenwich Mean Sidereal Time

Parameters

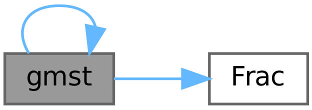
Mjd_UT1	Modified Julian Date UT1

Returns

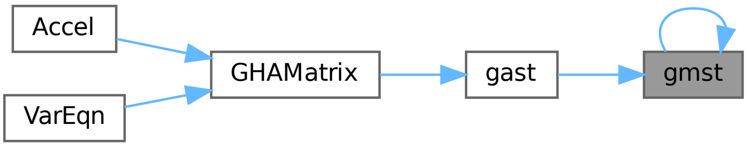
gmstime GMST in [rad]

Definition at line 12 of file gmst.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.100 gmst.cpp

Go to the documentation of this file.

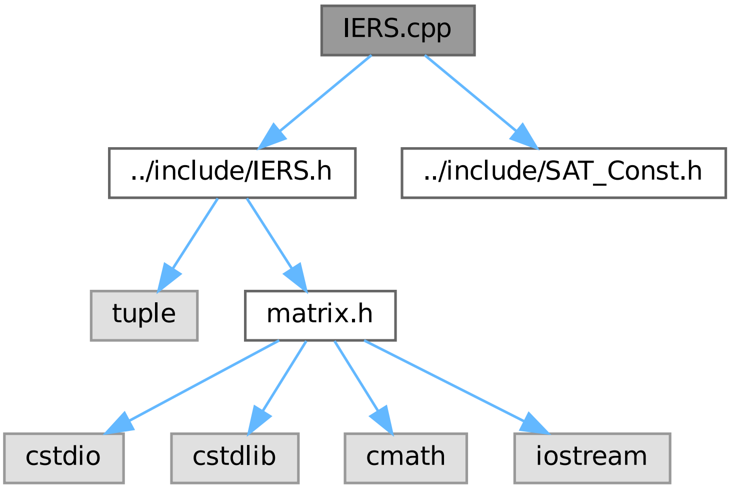
```
00001 #include "../include/gmst.h"
00002 #include "../include/Frac.h"
00003 #include "../include/SAT_Const.h"
00004 #include <cmath>
00005 
00012     double gmst(double Mjd_UT1){
00013         double Secs,MJD_J2000,Mjd_0,UT1,T_0,T,gmst;
00014         Secs = 86400.0;                       // Seconds per day
00015         MJD_J2000 = 51544.5;
00016 
00017         Mjd_0 = floor(Mjd_UT1);
00018         UT1   = Secs*(Mjd_UT1-Mjd_0);         // [s]
00019         T_0   = (Mjd_0  -MJD_J2000)/36525.0;
00020         T     = (Mjd_UT1-MJD_J2000)/36525.0;
00021 
00022         gmst  = 24110.54841 + 8640184.812866*T_0 + 1.002737909350795*UT1 + (0.093104-6.2e-6*T)*T*T;
      // [s]
00023 
00024         return 2*pi*Frac(gmst/Secs);       // [rad], 0..2pi
00025 
00026     }
```

5.101 IERS.cpp File Reference

El archivo contiene las implementaciones de IERS.h.

```
#include "../include/IERS.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for IERS.cpp:



Functions

- tuple< double, double, double, double, double, double, double, double, double > IERS (Matrix eop, double Mjd_UTC, char interp)

5.101.1 Detailed Description

El archivo contiene las implementaciones de IERS.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file IERS.cpp.

5.101.2 Function Documentation

5.101.2.1 IERS()

```
tuple< double, double, double, double, double, double, double, double, double > IERS (
            Matrix eop,
            double Mjd_UTC,
            char interp = 'n')
```

IERS: Management of IERS time and polar motion data

Parameters

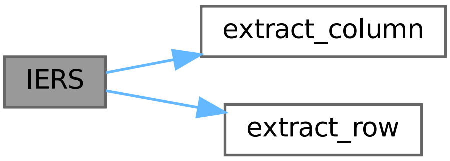
eop	matrix 4x13
Mjd_UTC	double
interp	char

Returns

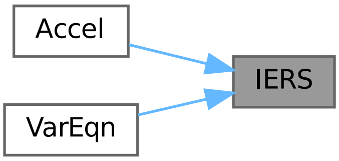
tupla <x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC>

Definition at line 10 of file IERS.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.102 IERS.cpp

Go to the documentation of this file.

```
#include "../include/IERS.h"
#include "../include/SAT_Const.h"

    tuple<double,double,double,double,double,double,double,double,double> IERS(Matrix eop,double
    Mjd_UTC,char interp){

    double x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC,i,fixf;
    int mjd;
    Matrix preeop,nexteop,aux;
    bool contin=true;
```

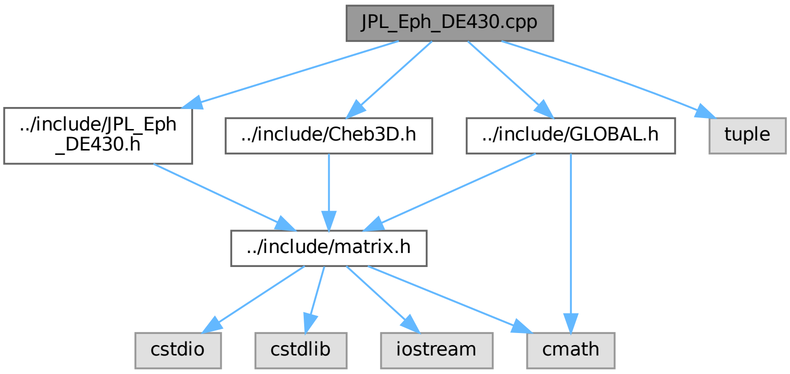
```

        mjd = (floor(Mjd_UTC));
        i = 1;
        aux=extract_row(eop,4);

        for (int j = 1;  contin && j <= aux.n_column ; j++) {
            if (aux(j)==mjd){
                i = j;
                contin=false;
            }
        }
    if (interp ==’l’){
        // linear interpolation
        preeop = extract_column(eop,i);
        nexteop = extract_column(eop,i+1);
        fixf = (Mjd_UTC-floor(Mjd_UTC));
        // Setting of IERS Earth rotation parameters
        // (UT1-UTC [s], TAI-UTC [s], x ["], y ["])
        x_pole  = preeop(5)+(nexteop(5)-preeop(5))*fixf;
        y_pole  = preeop(6)+(nexteop(6)-preeop(6))*fixf;
        UT1_UTC = preeop(7)+(nexteop(7)-preeop(7))*fixf;
        LOD     = preeop(8)+(nexteop(8)-preeop(8))*fixf;
        dpsi    = preeop(9)+(nexteop(9)-preeop(9))*fixf;
        deps    = preeop(10)+(nexteop(10)-preeop(10))*fixf;
        dx_pole = preeop(11)+(nexteop(11)-preeop(11))*fixf;
        dy_pole = preeop(12)+(nexteop(12)-preeop(12))*fixf;
        TAI_UTC = preeop(13);

        x_pole  = x_pole/Arcs;  // Pole coordinate [rad]
        y_pole  = y_pole/Arcs;  // Pole coordinate [rad]
        dpsi    = dpsi/Arcs;
        deps    = deps/Arcs;
        dx_pole = dx_pole/Arcs; // Pole coordinate [rad]
        dy_pole = dy_pole/Arcs; // Pole coordinate [rad]
    }
    else if (interp ==’n’)    {
        aux = extract_row(eop,i);
        eop=aux;
        // Setting of IERS Earth rotation parameters
        // (UT1-UTC [s], TAI-UTC [s], x ["], y ["])
        x_pole  = eop(5)/Arcs;  // Pole coordinate [rad]
        y_pole  = eop(6)/Arcs;  // Pole coordinate [rad]
        UT1_UTC = eop(7);             // UT1-UTC time difference [s]
        LOD     = eop(8);             // Length of day [s]
        dpsi    = eop(9)/Arcs;
        deps    = eop(10)/Arcs;
        dx_pole = eop(11)/Arcs; // Pole coordinate [rad]
        dy_pole = eop(12)/Arcs; // Pole coordinate [rad]
        TAI_UTC = eop(13);            // TAI-UTC time difference [s]
    }

return tie(x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC);

    }
```

5.103 JPL_Eph_DE430.cpp File Reference

El archivo contiene las implementaciones de JPL_Eph_DE430.h.

```
#include "../include/JPL_Eph_DE430.h"
#include "../include/Cheb3D.h"
#include "../include/GLOBAL.h"
#include <tuple>
```

Include dependency graph for JPL_Eph_DE430.cpp:



Functions

- tuple< Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix & > JPL_Eph_DE430 (double Mjd_TDB)

5.103.1 Detailed Description

El archivo contiene las implementaciones de JPL_Eph_DE430.h.

Author

Pedro Zhuzhan

Bug Noknownbugs

Definition in file JPL_Eph_DE430.cpp.

5.103.2 Function Documentation

5.103.2.1 JPL_Eph_DE430()

```
tuple< Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix &, Matrix
&, Matrix &, Matrix & > JPL_Eph_DE430 (
            double Mjd_TDB)
```

Parameters

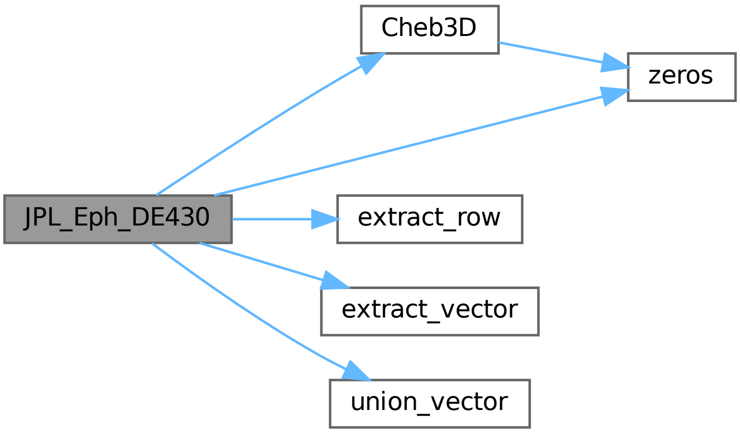
P	Matrix
Phi	Matrix
Qdt	Matrix

Returns

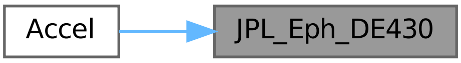
Matrix

Definition at line 12 of file JPL_Eph_DE430.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.104 JPL_Eph_DE430.cpp

Go to the documentation of this file.

```
00001 #include"../include/JPL_Eph_DE430.h"
00002 #include"../include/Cheb3D.h"
00003 #include"../include/GLOBAL.h"
00004 #include<tuple>
00005
00012 tuple<Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&,Matrix&>
      JPL_Eph_DE430(double Mjd_TDB){
00013
00014 double JD,t1,dt,j,Mjd0;
00015 Matrix Cx_Earth(12),Cy_Earth(12),Cz_Earth(12),Cx(12),Cy(12),Cz(12),
00016
      Cx_Moon(12),Cy_Moon(12),Cz_Moon(12),temp(4),Cx_Sun(12),Cy_Sun(12),Cz_Sun(12),Cx_Venus(12),Cy_Venus(12),Cz_Venus(12),Cx_M
00017
      Cx_Mars(12),Cy_Mars(12),Cz_Mars(12),Cx_Jupiter(12),Cy_Jupiter(12),Cz_Jupiter(12),Cx_Saturn(12),Cy_Saturn(12),Cz_Saturn(1
00018 ,Cx_Neptune(12),Cy_Neptune(12),Cz_Neptune(12),Cx_Pluto(12),Cy_Pluto(12),Cz_Pluto(12),
      &r_Mercury=zeros(3),&r_Venus=zeros(3),&r_Earth=zeros(3),&r_Mars=zeros(3),&r_Jupiter=zeros(3),

      &r_Saturn=zeros(3),&r_Uranus=zeros(3),&r_Neptune=zeros(3),&r_Pluto=zeros(3),&r_Moon=zeros(3),&r_Sun=zeros(3);
      JD=Mjd_TDB+2400000.5;
      int i;
      for(i=1;i<=PC.n_row;i++){
          if(PC(i,1)<=JD&&JD<=PC(i,2)){
              break;
          }
      }
      Matrix PCtemp=extract_row(PC,i);
      t1=PCtemp(1)-2400000.5;

      dt=Mjd_TDB-t1;
      for(int k=1;k<=4;k++){
          temp(k)=231+(k-1)*13;
      }
          Cx_Earth=extract_vector(PCtemp,temp(1),temp(2)-1);
          Cy_Earth=extract_vector(PCtemp,temp(2),temp(3)-1);
          Cz_Earth=extract_vector(PCtemp,temp(3),temp(4)-1);
          temp=temp+39;
          Cx=extract_vector(PCtemp,temp(1),temp(2)-1);
          Cy=extract_vector(PCtemp,temp(2),temp(3)-1);
          Cz=extract_vector(PCtemp,temp(3),temp(4)-1);
          Cx_Earth=union_vector(Cx_Earth,Cx);
          Cy_Earth=union_vector(Cy_Earth,Cy);
          Cz_Earth=union_vector(Cz_Earth,Cz);
      if(0<=dt&&dt<=16){
          j=0;
      Mjd0=t1;
      }
      else if(16<dt&&dt<=32){
          j=1;
      Mjd0=t1+16*j;
      }

      r_Earth=Cheb3D(Mjd_TDB,13,Mjd0,Mjd0+16,extract_vector(Cx_Earth,13*j+1,13*j+13),extract_vector(Cy_Earth,13*j+1,13*j+13),e


      for(int k=1;k<=4;k++){
          temp(k)=441+(k-1)*13;
      }
      Cx_Moon=extract_vector(PCtemp,temp(1),temp(2)-1);
      Cy_Moon=extract_vector(PCtemp,temp(2),temp(3)-1);
      Cz_Moon=extract_vector(PCtemp,temp(3),temp(4)-1);
      for(i=1;i<7;i++){
          temp=temp+39;
      Cx=extract_vector(PCtemp,temp(1),temp(2)-1);
      Cy=extract_vector(PCtemp,temp(2),temp(3)-1);
      Cz=extract_vector(PCtemp,temp(3),temp(4)-1);
      Cx_Moon=union_vector(Cx_Moon,Cx);
      Cy_Moon=union_vector(Cy_Moon,Cy);
      Cz_Moon=union_vector(Cz_Moon,Cz);
      }
      if(0<=dt&&dt<=4){
      j=0;
      Mjd0=t1;}
      else if(4<dt&&dt<=8){
      j=1;
      Mjd0=t1+4*j;}
      else if(8<dt&&dt<=12){
      j=2;
      Mjd0=t1+4*j;}
      else if(12<dt&&dt<=16){
      j=3;
      Mjd0=t1+4*j;}
      else if(16<dt&&dt<=20){
      j=4;
      Mjd0=t1+4*j;}
      else if(20<dt&&dt<=24){
      j=5;
      Mjd0=t1+4*j;}
      else if(24<dt&&dt<=28){
      j=6;
      Mjd0=t1+4*j;}
      else if(28<dt&&dt<=32){
      j=7;
      Mjd0=t1+4*j;}

      r_Moon=Cheb3D(Mjd_TDB,13,Mjd0,Mjd0+4,extract_vector(Cx_Moon,13*j+1,13*j+13),extract_vector(Cy_Moon,13*j+1,13*j+13),extra
      for(int k=1;k<=4;k++){
          temp(k)=753+(k-1)*11;
      }
      Cx_Sun=extract_vector(PCtemp,temp(1),temp(2)-1);
      Cy_Sun=extract_vector(PCtemp,temp(2),temp(3)-1);
      Cz_Sun=extract_vector(PCtemp,temp(3),temp(4)-1);
      temp=temp+33;
```

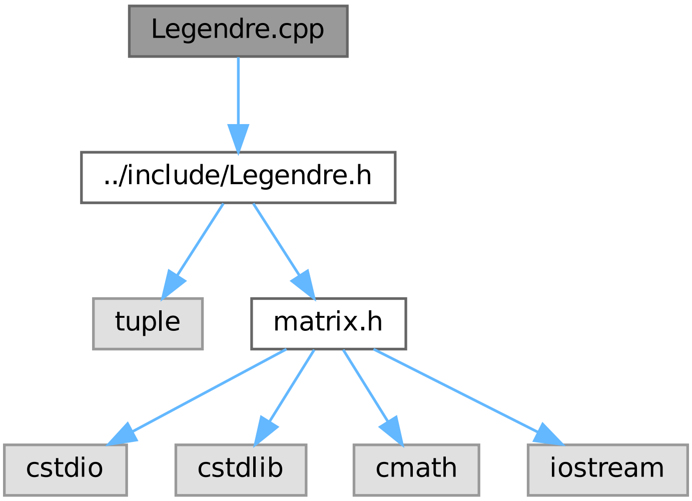
```
Cx=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz=extract_vector(PCtemp,temp(3),temp(4)-1);
Cx_Sun=union_vector(Cx_Sun,Cx);
Cy_Sun=union_vector(Cy_Sun,Cy);
Cz_Sun=union_vector(Cz_Sun,Cz);
if(0<=dt&&dt<=16){
j=0;
Mjd0=t1;}
else if(16<dt&&dt<=32){
j=1;
Mjd0=t1+16*j;}
r_Sun=Cheb3D(Mjd_TDB,11,Mjd0,Mjd0+16,extract_vector(Cx_Sun,11*j+1,11*j+11),extract_vector(Cy_Sun,11*j+1,11*j+11),extract
for(int k=1;k<=4;k++){
    temp(k)=3+(k-1)*14;
}
Cx_Mercury=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Mercury=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Mercury=extract_vector(PCtemp,temp(3),temp(4)-1);
temp=temp+42;
Cx=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz=extract_vector(PCtemp,temp(3),temp(4)-1);
Cx_Mercury=union_vector(Cx_Mercury,Cx);
Cy_Mercury=union_vector(Cy_Mercury,Cy);
Cz_Mercury=union_vector(Cz_Mercury,Cz);
if(0<=dt&&dt<=8){
j=0;
Mjd0=t1;}
else if(8<dt&&dt<=16){
j=1;
Mjd0=t1+8*j;}
else if(16<dt&&dt<=24){
j=2;
Mjd0=t1+8*j;}
else if(24<dt&&dt<=32){
j=3;
Mjd0=t1+8*j;}
r_Mercury=Cheb3D(Mjd_TDB,14,Mjd0,Mjd0+8,extract_vector(Cx_Mercury,14*j+1,14*j+14),extract_vector(Cy_Mercury,14*j+1,14*j
for(int k=1;k<=4;k++){
    temp(k)=171+(k-1)*10;
}
Cx_Venus=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Venus=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Venus=extract_vector(PCtemp,temp(3),temp(4)-1);
temp=temp+30;
Cx=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz=extract_vector(PCtemp,temp(3),temp(4)-1);
Cx_Venus=union_vector(Cx_Venus,Cx);
Cy_Venus=union_vector(Cy_Venus,Cy);
Cz_Venus=union_vector(Cz_Venus,Cz);
if(0<=dt&&dt<=16){
j=0;
Mjd0=t1;}
else if(16<dt&&dt<=32){
j=1;
Mjd0=t1+16*j;}
r_Venus=Cheb3D(Mjd_TDB,10,Mjd0,Mjd0+16,extract_vector(Cx_Venus,10*j+1,10*j+10),extract_vector(Cy_Venus,10*j+1,10*j+10),e
for(int k=1;k<=4;k++){
    temp(k)=309+(k-1)*11;
}
Cx_Mars=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Mars=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Mars=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Mars=Cheb3D(Mjd_TDB,11,Mjd0,Mjd0+32,Cx_Mars,Cy_Mars,Cz_Mars)*1e3;

for(int k=1;k<=4;k++){
    temp(k)=342+(k-1)*8;
}
Cx_Jupiter=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Jupiter=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Jupiter=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Jupiter=Cheb3D(Mjd_TDB,8,Mjd0,Mjd0+32,Cx_Jupiter,Cy_Jupiter,Cz_Jupiter)*1e3;

for(int k=1;k<=4;k++){
    temp(k)=366+(k-1)*7;
}
Cx_Saturn=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Saturn=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Saturn=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Saturn=Cheb3D(Mjd_TDB,7,Mjd0,Mjd0+32,Cx_Saturn,
Cy_Saturn,Cz_Saturn)*1e3;

for(int k=1;k<=4;k++){
    temp(k)=387+(k-1)*6;
}
Cx_Uranus=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Uranus=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Uranus=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Uranus=Cheb3D(Mjd_TDB,6,Mjd0,Mjd0+32,Cx_Uranus,
Cy_Uranus,Cz_Uranus)*1e3;

for(int k=1;k<=4;k++){
    temp(k)=405+(k-1)*6;
}
Cx_Neptune=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Neptune=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Neptune=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Neptune=Cheb3D(Mjd_TDB,6,Mjd0,Mjd0+32,Cx_Neptune,
Cy_Neptune,Cz_Neptune)*1e3;
for(int k=1;k<=4;k++){
    temp(k)=423+(k-1)*6;
}
Cx_Pluto=extract_vector(PCtemp,temp(1),temp(2)-1);
Cy_Pluto=extract_vector(PCtemp,temp(2),temp(3)-1);
Cz_Pluto=extract_vector(PCtemp,temp(3),temp(4)-1);
Mjd0=t1;
r_Pluto=Cheb3D(Mjd_TDB,6,Mjd0,Mjd0+32,Cx_Pluto,Cy_Pluto,Cz_Pluto)*1e3;
double EMRAT=81.30056907419062;
double EMRAT1=1/(1+EMRAT);
Matrix aux(3);
aux=r_Moon;
 r_Earth=r_Earth-aux*EMRAT1;
 aux=r_Earth;
 aux=aux*(-1);
 r_Mercury=aux+r_Mercury;
 r_Venus=aux+r_Venus;
 r_Mars=aux+r_Mars;
 r_Jupiter=aux+r_Jupiter;
 r_Saturn=aux+r_Saturn;
 r_Uranus=aux+r_Uranus;
 r_Neptune=aux+r_Neptune;
 r_Pluto=aux+r_Pluto;
 r_Sun=aux+r_Sun;
 return
tie(r_Mercury,r_Venus,r_Earth,r_Mars,r_Jupiter,r_Saturn,r_Uranus,r_Neptune,r_Pluto,r_Moon,r_Sun);
    }
```

5.105 Legendre.cpp File Reference

El archivo contiene las implementaciones de Legendre.h.

```
#include "../include/Legendre.h"
```

Include dependency graph for Legendre.cpp:



Functions

- tuple< Matrix &, Matrix & > Legendre (int n, int m, double fi)

5.105.1 Detailed Description

El archivo contiene las implementaciones de Legendre.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Legendre.cpp.

5.105.2 Function Documentation

5.105.2.1 Legendre()

```
tuple< Matrix &, Matrix & > Legendre (
            int n,
            int m,
            double fi)
```

Time differences [s]

Parameters

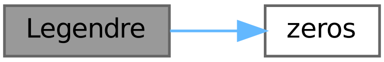
n	int
m	int
fi	double [rad]

Returns

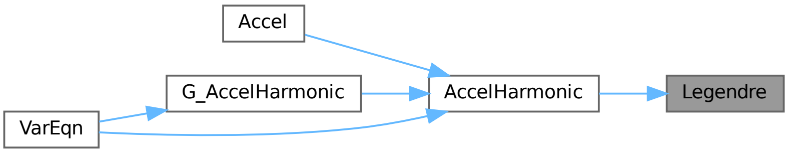
tupla <pnm,dpnm>

Definition at line 8 of file Legendre.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.106 Legendre.cpp

Go to the documentation of this file.

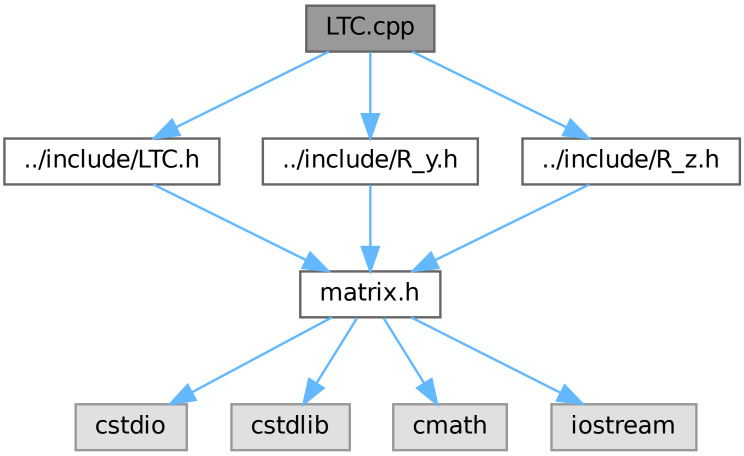
```
00001 #include "../include/Legendre.h"
00008     tuple<Matrix&,Matrix&>  Legendre(int n,int m,double fi){
00009     Matrix &pnm = zeros(n+1,m+1);
00010     Matrix &dpnm = zeros(n+1,m+1);
00011     int i=0;
00012 
00013     pnm(1,1)=1;
00014     dpnm(1,1)=0;
00015     pnm(2,2)=sqrt(3)*cos(fi);
00016     dpnm(2,2)=-sqrt(3)*sin(fi);
00017     // diagonal coefficients
00018     for (i=2;i<=n;i++){
00019         pnm(i+1,i+1)= sqrt((2.0*i+1.0)/(2.0*i))*cos(fi)*pnm(i,i);
00020     }
00021     for (i=2;i<=n;i++){
00022         dpnm(i+1,i+1)= sqrt((2.0*i+1.0)/(2.0*i))*((cos(fi)*dpnm(i,i))-(sin(fi)*pnm(i,i)));}
00023     // horizontal first step coefficients
00024     for (i=1;i<=n;i++){
        pnm(i+1,i)= sqrt(2.0*i+1)*sin(fi)*pnm(i,i);}
    for (i=1;i<=n;i++){
        dpnm(i+1,i)= sqrt(2.0*i+1)*((cos(fi)*pnm(i,i))+(sin(fi)*dpnm(i,i)));}
    // horizontal second step coefficients
    int j=0;
    int k=2;
    while(true){
        for (i=k;i<=n;i++){
            pnm(i+1,j+1)=sqrt((2.0*i+1.0)/((i-j)*(i+j)))*((sqrt(2.0*i-1.0)*sin(fi)*pnm(i,j+1.0))
-(sqrt(((i+j-1.0)*(i-j-1.0))/(2.0*i-3.0))*pnm(i-1,j+1.0)));}
        j = j+1;
        k = k+1;
        if (j>m){
            break;
        }
    }

    j = 0;
    k = 2;
    while(true){
        for (i=k;i<=n;i++){

dpnm(i+1,j+1)=sqrt((2.0*i+1.0)/((i-j)*(i+j)))*((sqrt(2.0*i-1)*sin(fi)*dpnm(i,j+1))+(sqrt(2.0*i-1.0)*cos(fi)*pnm(i,j+1.0
                -(sqrt(((i+j-1.0)*(i-j-1))/(2.0*i-3.0))*dpnm(i-1,j+1)));}
        j = j+1;
        k = k+1;
        if (j>m){
            break;
        }

        }
        return tie(pnm, dpnm);
}
```

5.107 LTC.cpp File Reference

El archivo contiene las implementaciones de LTC.h.

```
#include "../include/LTC.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
```

Include dependency graph for LTC.cpp:



Functions

- Matrix & LTC (double lon, double lat)

5.107.1 Detailed Description

El archivo contiene las implementaciones de LTC.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file LTC.cpp.

5.107.2 Function Documentation

5.107.2.1 LTC()

```
Matrix & LTC (
             double lon,
             double lat)
```

Transformation from Greenwich meridian system to local tangent coordinates

Parameters

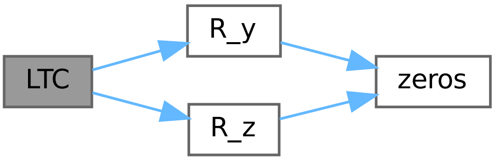
lon	-Geodetic East longitude [rad]
lat	-Geodetic latitude [rad]

Returns

M -Rotation matrix from the Earth equator and Greenwich meridian to the local tangent (East-North-Zenith) coordinate system

Definition at line 10 of file LTC.cpp.

Here is the call graph for this function:



5.108 LTC.cpp

Go to the documentation of this file.

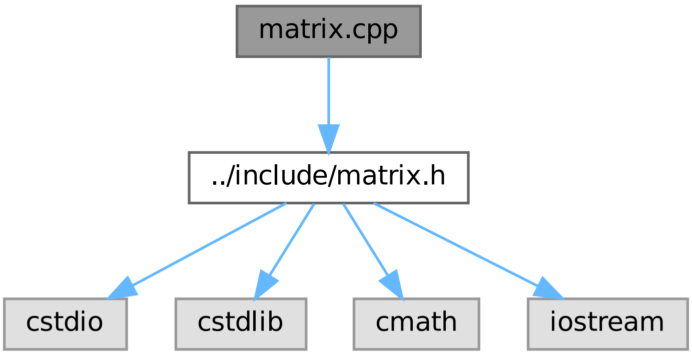
```
#include "../include/LTC.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
    Matrix& LTC(double lon,double lat){
        Matrix &M= R_y(-1.0*lat)*R_z(lon);
        double Aux;
    for (int j=1;j<=3;j++){
        Aux=M(1,j);
        M(1,j)=M(2,j);
        M(2,j)=M(3,j);
        M(3,j)= Aux;}
    return M;

    }
```

5.109 matrix.cpp File Reference

El archivo contiene las implementaciones de matrix.h.

```
#include "../include/matrix.h"
```

Include dependency graph for matrix.cpp:



Functions

- ostream & operator<< (ostream &o, Matrix &m)
- Matrix & zeros (const int n_row, const int n_column)
- Matrix & eye (const int size)
- Matrix & transpose (Matrix &m)
- void swap_row (Matrix &m, int i, int index)
- Matrix & inv (Matrix &m)
- Matrix & zeros (const int n)
- double norm (Matrix &m)
- double dot (Matrix &v, Matrix &w)
- Matrix & cross (Matrix &v, Matrix &w)
- Matrix & extract_vector (Matrix &v, int start, int end)

- Matrix & union_vector (Matrix &v, Matrix &w)
- Matrix & extract_row (Matrix &v, int j)
- Matrix & extract_column (Matrix &v, int j)
- Matrix & assign_row (Matrix &v, Matrix &w, int j)
- Matrix & assign_column (Matrix &v, Matrix &w, int j)

5.109.1 Detailed Description

El archivo contiene las implementaciones de matrix.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file matrix.cpp.

5.109.2 Function Documentation

5.109.2.1 assign_column()

```
Matrix & assign_column (
             Matrix & v,
             Matrix & w,
             int i)
```

Asigna la columna i-1 con w y lo devuelve

Parameters

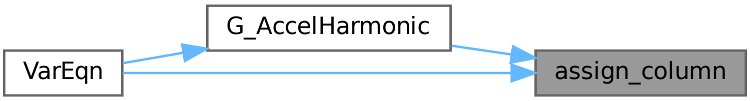
v	Matrix con tamaño x x y ,x pertenece a N y pertenece a N
w	Matrix con tamaño 1 x y ,y pertenece a N
i	es la columna tiene que ser >=1 && <=v.n_column

Returns

Matrix v con la columna i-1 cambiada por los elementos de w

Definition at line 424 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.2 assign_row()

```
Matrix & assign_row (
            Matrix & v,
            Matrix & w,
            int i)
```

Asigna la fila i-1 con w y lo devuelve

Parameters

v	Matrix con tamaño x x y ,x pertenece a N y pertenece a N
w	Matrix con tamaño 1 x y ,y pertenece a N
i	es la fila tiene que ser >=1 && <=v.n_row

Returns

Matrix v con la fila i-1 cambiada por los elementos de w

Definition at line 412 of file matrix.cpp.

5.109.2.3 cross()

```
Matrix & cross (
            Matrix & v,
            Matrix & w)
```

Devulve el producto vectorial

Parameters

v	Matrix con tamaño 1 x 3
w	Matrix con tamaño 1 x 3

Returns

producto escalar de v x w

Definition at line 353 of file matrix.cpp.

5.109.2.4 dot()

```
double dot (
            Matrix & v,
            Matrix & w)
```

Devulve el producto escalar

Parameters

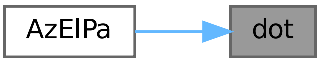
v	Matrix con tamaño 1 x 3
w	Matrix con tamaño 1 x 3

Returns

producto escalar de v·w

Definition at line 342 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.5 extract_column()

```
Matrix & extract_column (
            Matrix & v,
            int i)
```

Extrae la columna i-1 de v y lo devuelve

Parameters

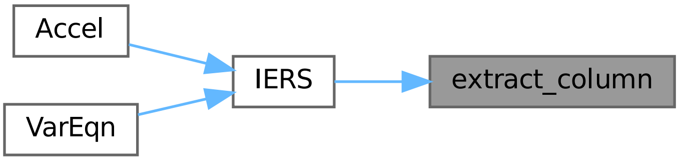
v	Matrix
i	es la columna tiene que ser >=1 && <=v.n_column

Returns

Matrix con la columna i-1 de v

Definition at line 401 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.6 extract_row()

```
Matrix & extract_row (
            Matrix & v,
            int i)
```

Extrae la fila i-1 de v y lo devuelve

Parameters

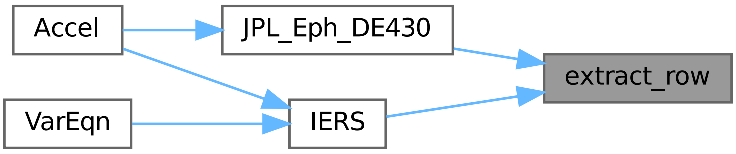
v	Matrix
i	es la fila tiene que ser >=1 && <=v.n_row

Returns

Matrix con la fila i-1 de v

Definition at line 389 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.7 extract_vector()

```
Matrix & extract_vector (
            Matrix & v,
            int start,
            int end)
```

Extrae del vector v desde la posicion start hasta end, incluidos

Parameters

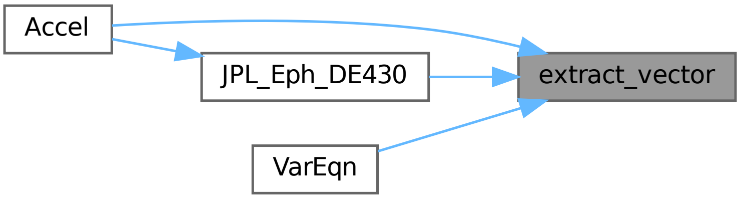
v	Matrix 1 x n
start	inicio del vector resultado
end	fin del vector resultado

Returns

Matrix 1 x (end - start + 1)

Definition at line 364 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.8 eye()

```
Matrix & eye (
             const int size)
```

Crea una Matrix identidad con tamaño size x size

Parameters

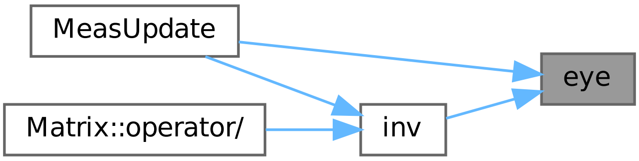
size	dimension de la matriz

Returns

una Matrix tamaño size x size

Definition at line 233 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.9 inv()

```
Matrix & inv (
             Matrix & m)
```

Crea una Matrix inversa de m, sin modificar m

Parameters

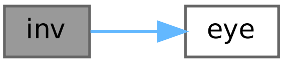
m	Matrix que tiene que ser cuadrada, es decir, con el mismo numero de columnas que filas

Returns

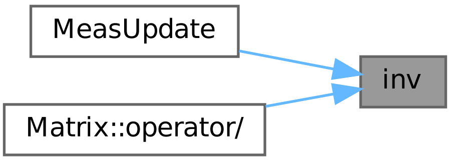
una Matrix inversa de m

Definition at line 267 of file matrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.109.2.10 norm()

```
double norm (
            Matrix & m)
```

Devulve la norma 2 de una Matrix que simula un vector

Parameters

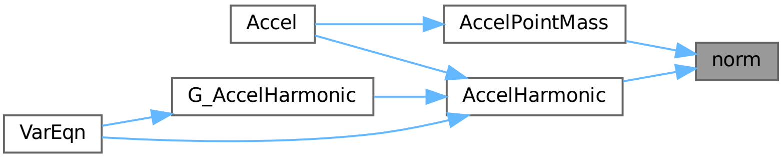
m	Matrix 1 x n_column

Returns

la norma 2 de m

Definition at line 333 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.11 operator<<()

```
ostream & operator<< (
             ostream & o,
             Matrix & m)
```

Definition at line 212 of file matrix.cpp.

5.109.2.12 swap_row()

```
void swap_row (
             Matrix & m,
             int i,
             int index)
```

Definition at line 258 of file matrix.cpp.

5.109.2.13 transpose()

```
Matrix & transpose (
             Matrix & m)
```

Crea una Matrix traspuesta de m,sin modificar m

Parameters

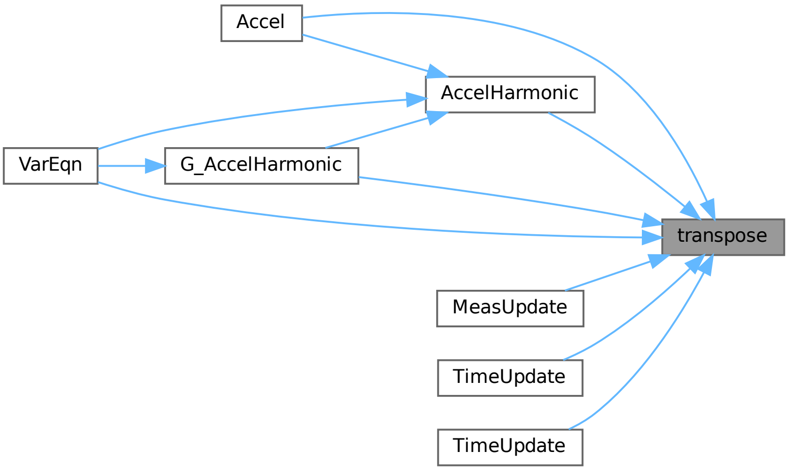
m	Matrix

Returns

una Matrix traspuesta de m

Definition at line 246 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.14 union_vector()

```
Matrix & union_vector (
             Matrix & v,
             Matrix & w)
```

Devuelve la matriz resultado de realizar la union entre v y w

Parameters

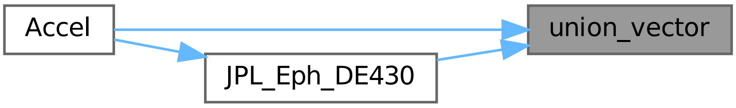
v	Matrix con tamaño 1 x n ,n pertenece a N
w	Matrix con tamaño 1 x n ,n pertenece a N

Returns

producto escalar de v x w

Definition at line 375 of file matrix.cpp.

Here is the caller graph for this function:



5.109.2.15 zeros() [1/2]

```
Matrix & zeros (
             const int n)
```

Crea una Matrix con todas sus componentes a 0

Parameters

n	numero de columnas que tiene la matriz

Returns

una Matrix tamaño 1 x n

Definition at line 323 of file matrix.cpp.

5.109.2.16 zeros() [2/2]

```
Matrix & zeros (
             const int n_row,
             const int n_column)
```

Crea una Matrix con todas sus componentes a 0

Parameters

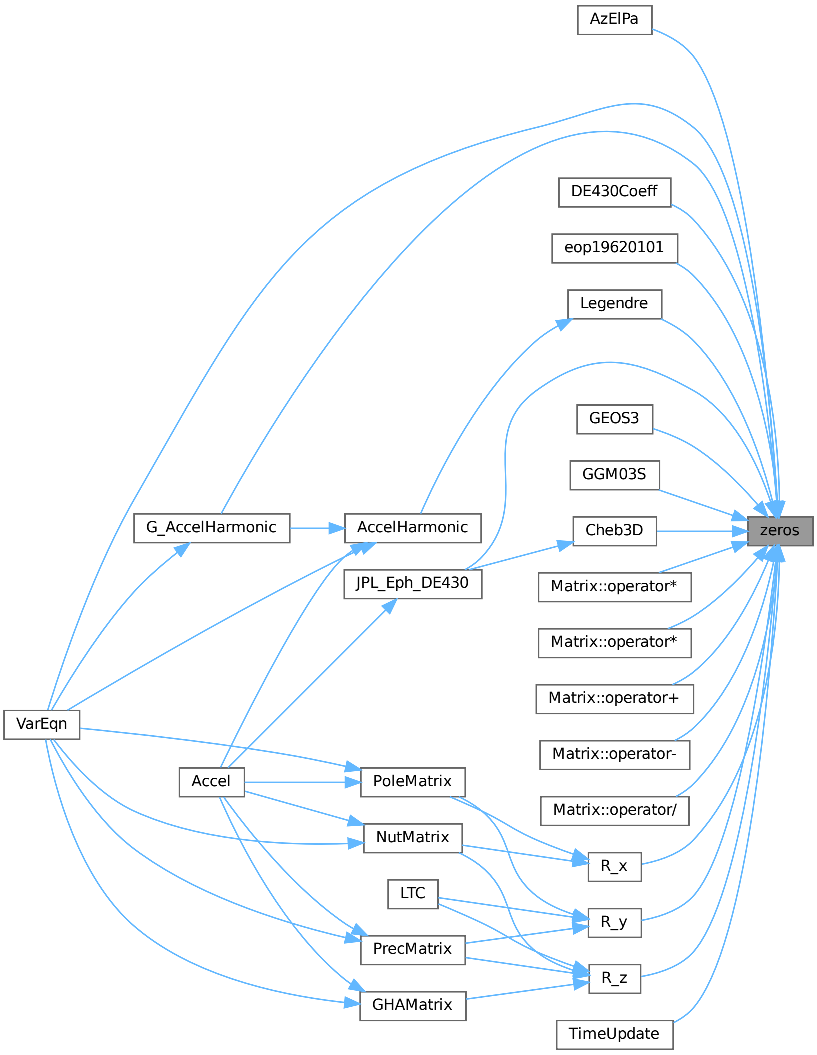
n_row	numero de filas que tiene la matriz
n_column	numero de columnas que tiene la matriz

Returns

una Matrix tamaño n_row x n_column

Definition at line 222 of file matrix.cpp.

Here is the caller graph for this function:



5.110 matrix.cpp

Go to the documentation of this file.

```
00001 #include "../include/matrix.h"
00002
00009 //----------------------------------
00010 Matrix::Matrix() {
00011     this->n_row = 0;
00012     this->n_column = 0;
00013     this->data = NULL;
00014
00015 }
```

```
//----------------------------------
Matrix::Matrix(const int n_size) {
    if (n_size <= 0) {
        cout « "Vector create: error in n_row/n_column\n";
        exit(EXIT_FAILURE);
    }

    this->n_row = 1;
    this->n_column = n_size;
    this->data = (double **) malloc(n_row*sizeof(double *));

    if (this->data == NULL) {
        cout « "Vector create: error in data\n";
        exit(EXIT_FAILURE);
    }
    this->data[0] = (double *) calloc(n_size,sizeof(double));

}
//--------------------------------------
Matrix::Matrix(const int n_row, const int n_column) {
    if (n_row <= 0 || n_column <= 0) {
        cout « "Matrix create: error in n_row/n_column\n";
        exit(EXIT_FAILURE);
    }

    this->n_row = n_row;
    this->n_column = n_column;
    this->data = (double **) malloc(n_row*sizeof(double *));

    if (this->data == NULL) {
        cout « "Matrix create: error in data\n";
        exit(EXIT_FAILURE);
    }

    for(int i = 0; i < n_row; i++) {
        this->data[i] = (double *) malloc(n_column*sizeof(double));
    }
}
//----------------------------------
double& Matrix::operator () (const int n) {
    if (n <= 0 || n > this->n_column* this->n_row) {
        cout « "Vector get: error in get:"«n«" size: " «this->n_column*this->n_row«" row/column\n";
        exit(EXIT_FAILURE);
    }

    return this->data[(n - 1)/this->n_column][(n-1)%this->n_column];
}
//----------------------------------
double& Matrix::operator () (const int row, const int column) {
    if (row <= 0 || row > this->n_row || column <= 0 || column > this->n_column) {
        cout « "Matrix get: error in " «row«" row/ "«column«" column\n";
        cout « "Matrix " «this->n_row«" n_row/ "«this->n_column«" n_column\n";
        exit(EXIT_FAILURE);
    }

    return this->data[row - 1][column - 1];
}
//----------------------------------
Matrix& Matrix::operator + (Matrix &m) {
    if (this->n_row != m.n_row || this->n_column != m.n_column) {
        cout « "Matrix sum: error in n_row/n_column\n";
        exit(EXIT_FAILURE);
    }

    Matrix *m_aux = new Matrix(this->n_row, this->n_column);

    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            (*m_aux)(i,j) = (*this)(i,j) + m(i,j);
        }
    }

    return *m_aux;
}
//----------------------------------
Matrix& Matrix::operator - (Matrix &m) {
    if (this->n_row != m.n_row || this->n_column != m.n_column) {
        cout « "Matrix sub: error in n_row/n_column\n";
        exit(EXIT_FAILURE);
    }

    Matrix *m_aux = new Matrix(this->n_row, this->n_column);

    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            (*m_aux)(i,j) = (*this)(i,j) - m(i,j);
        }
    }

    return *m_aux;
}
//-----------------------------------
Matrix& Matrix::operator * (Matrix &m){
    if (this->n_column != m.n_row) {
        cout « "Matrix muliplication: error in n->n_column, m.n_row\n";
        exit(EXIT_FAILURE);
    }

    Matrix &m_aux=zeros(this->n_row, m.n_column);

    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= m.n_column; j++) {
                    m_aux(i,j)=0;
            for(int k = 1; k <= this->n_column; k++) {
                m_aux(i,j) += (*this)(i,k) * m(k,j);
            }
        }
    }
    return m_aux;
}
//-----------------------------------
Matrix& Matrix::operator / (Matrix &m){
    if (this->n_column != m.n_row) {
        cout « "Matrix sub: error in n->n_column, m.n_row\n";
        exit(EXIT_FAILURE);
    }

    Matrix *m_aux= new Matrix(this->n_row,m.n_column);
    *m_aux=(*this)*inv(m);
    return *m_aux;
}
//-----------------------------------
Matrix& Matrix::operator = (Matrix &m){

    Matrix *m_aux = new Matrix(m.n_row, m.n_column);

    for (int i = 1; i <= m.n_row; i++) {
        for (int j = 1; j <= m.n_column; j++){
                    (*m_aux)(i,j)=m(i,j);
        }
    }

    this->n_row = m.n_row;
    this->n_column = m.n_column;
    this->data = (double **) malloc(m.n_row*sizeof(double *));

    if (this->data == NULL) {
        cout « "Matrix create: error in data\n";
        exit(EXIT_FAILURE);
    }

    for(int i = 0; i < m.n_row; i++) {
        this->data[i] = (double *) malloc(m.n_column*sizeof(double));
    }
    for (int i = 1; i <= this->n_row; i++) {
        for (int j = 1; j <= this->n_column; j++){
                    (*this)(i,j)=(*m_aux)(i,j);
        }
    }
    return *this;
}
//-----------------------------------
Matrix& Matrix::operator + (double d){

    Matrix &m_aux=zeros(this->n_row, this->n_column);

    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            m_aux(i,j) = (*this)(i,j) + d;
        }
    }

    return m_aux;
}
Matrix& Matrix::operator - (double d){

    Matrix &m_aux=zeros(this->n_row, this->n_column);

    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            (m_aux)(i,j) =(*this)(i,j)- d;
        }
    }
    return m_aux;
```

```
}
//-----------------------------------
Matrix& Matrix::operator * (double d){
    Matrix &m_aux=zeros(this->n_row, this->n_column);
    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            (m_aux)(i,j) =(*this)(i,j)* d;
        }
    }
    return m_aux;
}
//----------------------------------
Matrix& Matrix::operator / (double d){
    Matrix &m_aux=zeros(this->n_row, this->n_column);
    for(int i = 1; i <= this->n_row; i++) {
        for(int j = 1; j <= this->n_column; j++) {
            (m_aux)(i,j) =(*this)(i,j)/ d;
        }
    }
    return m_aux;
}
//-----------------------------------
ostream& operator « (ostream &o, Matrix &m) {
    for (int i = 1; i <= m.n_row; i++) {
        for (int j = 1; j <= m.n_column; j++)
            printf("%5.20lf ", m(i,j));
        o « "\n";
    }

    return o;
}
//-----------------------------------
Matrix& zeros(const int n_row, const int n_column) {
    Matrix *m_aux = new Matrix(n_row, n_column);

    for(int i = 1; i <= n_row; i++) {
        for(int j = 1; j <= n_column; j++) {
            (*m_aux)(i,j) = 0;
        }
    }
    return (*m_aux);
}
//-----------------------------------
Matrix& eye(const int size){
    Matrix *m_aux = new Matrix(size,size);

    for(int i = 1; i <= size; i++) {
        for(int j = 1; j <= size; j++) {
            (*m_aux)(i,j) = 0;
        }
        (*m_aux)(i,i) = 1;
    }

    return (*m_aux);
}
//-----------------------------------
Matrix& transpose(Matrix &m) {
    Matrix *m_aux = new Matrix(m.n_column,m.n_row);

    for(int i = 1; i <= m.n_row; i++) {
        for(int j = 1; j <= m.n_column; j++) {
            (*m_aux)(j,i) = m(i,j);
        }
    }

    return (*m_aux);
}
//-----------------------------------
void swap_row(Matrix &m,int i,int index){
    double aux;
    for(int k=1;k<=m.n_column;k++){
        aux=m(i,k);
        m(i,k)=m(index,k);
        m(index,k)=aux;
    }
}
//-----------------------------------
Matrix& inv(Matrix &m) {
    if (m.n_column != m.n_row) {
        cout « "Matrix sub: error in m.n_column, m.n_row\n";
        exit(EXIT_FAILURE);
    }
    Matrix *m_aux=new Matrix(m.n_row,m.n_column);
    Matrix *m_aux1=new Matrix(m.n_row,m.n_column);
    double ratio,aux;
    *m_aux=m;
    *m_aux1=eye(m.n_row);
    int index;
    for(int i=1;i<=m.n_column;i++){
        index=i;
        aux=fabs((*m_aux)(i,i));
        for(int j=i+1;j<=m.n_column;j++){
            if(aux<fabs((*m_aux)(j,i))){
                aux=fabs((*m_aux)(j,i));
                index=j;
            }
        }
        swap_row(*m_aux,i,index);
        swap_row(*m_aux1,i,index);
        if((*m_aux)(i,i)==0){
            cout « "Error singular Matrix\n";
            exit(EXIT_FAILURE);
        }
        for(int j=i+1;j<=m.n_column;j++){
            if((*m_aux)(j,i)!=0){
                ratio=(*m_aux)(j,i)/(*m_aux)(i,i);
                if(ratio!=0){
                    for(int k=1;k<m.n_column+1;k++){
                        (*m_aux)(j,k)-=ratio*(*m_aux)(i,k);
                        (*m_aux1)(j,k)-=ratio*(*m_aux1)(i,k);
                    }
                }
            }
        }
    }
    for(int i=m.n_row;i>=1;i--){
        ratio=1/(*m_aux)(i,i);
            for(int k=1;k<=m.n_column;k++){
                        (*m_aux)(i,k)*=ratio;
                        (*m_aux1)(i,k)*=ratio;
            }
            for(int j=i-1;j>=1;j--){
                ratio=(*m_aux)(j,i);
                for(int k=1;k<=m.n_column;k++){
                        (*m_aux)(j,k)-=ratio*(*m_aux)(i,k);
                        (*m_aux1)(j,k)-=ratio*(*m_aux1)(i,k);
                }
            }
        }
        free(m_aux);
    return *m_aux1;
}
//-----------------------------------
Matrix& zeros(const int n) {
    Matrix *m_aux = new Matrix(n);

    for(int i = 1; i <= n; i++) {
            (*m_aux)(1,i) = 0;
    }

    return (*m_aux);
}
//-----------------------------------
double norm(Matrix &m) {
    double r=0;

    for(int i = 1; i <= m.n_column*m.n_row; i++) {
            r+=m(i)*m(i);
    }
    return sqrt(r);
}
//-----------------------------------
double dot(Matrix &v,Matrix &w){
    if(v.n_column!=w.n_column){
        cout « "Vector dot: error in v.n_column, w.n_column\n";
        exit(EXIT_FAILURE);}
    double result=0;
    for(int i=1;i<=v.n_column*v.n_row;i++)
        result += v(i)*w(i);
    return result;
}

//-----------------------------------
Matrix& cross(Matrix &v,Matrix &w){
    if(v.n_column!=w.n_column){
        cout « "Vector cross: error in v.n_column, w.n_column\n";
        exit(EXIT_FAILURE);}
    Matrix *m_aux = new Matrix(v.n_column);
    (*m_aux)(1) = v(2)*w(3)-w(2)*v(3);
    (*m_aux)(2) = v(3)*w(1)-w(3)*v(1);
    (*m_aux)(3) = v(1)*w(2)-w(1)*v(2);
    return (*m_aux);
}
//-----------------------------------
```

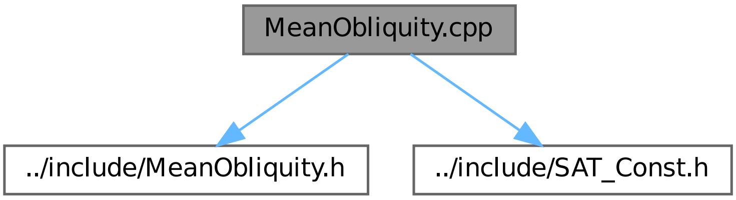
```
    Matrix& extract_vector(Matrix &v,int start,int end){

    Matrix *m_aux = new Matrix(end-start+1);
    int x=1;
    for (int i=start; i<=end;i++){
        (*m_aux)(x)=v(i);
        x++;
        }
    return *m_aux;
    }
//-----------------------------------
    Matrix& union_vector(Matrix &v,Matrix &w){
        int x=1,length=v.n_column*v.n_row+w.n_column*w.n_row;
        Matrix *v_aux=new Matrix(length);
        for(int i=1; i<=v.n_column*v.n_row;i++){
            (*v_aux)(x)=v(i);
            x++;
        }
        for(int i=1; i<=w.n_column*w.n_row;i++){
            (*v_aux)(x)=w(i);
            x++;
        }
        return (*v_aux);
    }
//-----------------------------------
    Matrix& extract_row(Matrix &v,int j){
        if(v.n_row<j || 1>j){
            cout « "Matrix extract_row: error in"« v.n_row «" "«j«"\n";
            exit(EXIT_FAILURE);}

            Matrix *m_aux = new Matrix(v.n_column);
            for (int i=1;i<=v.n_column;i++){
                (*m_aux)(i)=v(j,i);
            }
            return (*m_aux);
    }
//-----------------------------------
    Matrix& extract_column(Matrix &v,int j){
        if(v.n_column<j || j<1){
            cout « "Matrix extract_column: error in "« j «" "«v.n_column«"\n";
            exit(EXIT_FAILURE);}
            Matrix *m_aux = new Matrix(v.n_row);
            for (int i=1;i<=v.n_row;i++){
                (*m_aux)(i)=v(i,j);
            }
            return (*m_aux);
    }
//-----------------------------------
    Matrix& assign_row(Matrix &v,Matrix &w,int j){
        if(v.n_row<j || j<1 || v.n_row!=w.n_column){
            cout « "Matrix assign_row: error in v.n_row<j\n";
            exit(EXIT_FAILURE);}
            Matrix *m_aux=new Matrix(v.n_row,v.n_column);
            (*m_aux) = v;
            for (int i=1;i<=m_aux->n_column;i++){
                (*m_aux)(j,i)=w(i);
            }
            return (*m_aux);
    }
//-----------------------------------
    Matrix& assign_column(Matrix &v,Matrix &w,int j){
        if(v.n_column<j || j<1 || v.n_row!=w.n_column){
            cout « "Matrix assign_column: error in v.n_column<j\n";
            exit(EXIT_FAILURE);}
            Matrix *m_aux=new Matrix(v.n_row,v.n_column);
            (*m_aux) = v;
            for (int i=1;i<=m_aux->n_row;i++){
                (*m_aux)(i,j)=w(i);
            }
            return (*m_aux);

    }
//-----------------------------------
```

5.111 MeanObliquity.cpp File Reference

El archivo contiene las implementaciones de MeanObliquity.h.

```
#include "../include/MeanObliquity.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for MeanObliquity.cpp:



Functions

- double MeanObliquity (double Mjd_TT)

5.111.1 Detailed Description

El archivo contiene las implementaciones de MeanObliquity.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file MeanObliquity.cpp.

5.111.2 Function Documentation

5.111.2.1 MeanObliquity()

```
double MeanObliquity (
             double Mjd_TT)
```

Computes the mean obliquity of the ecliptic

Parameters

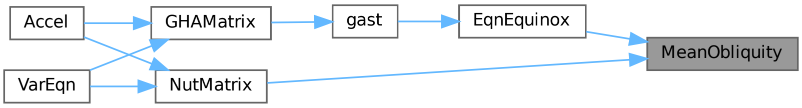
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

obliquity of the ecliptic [rad]

Definition at line 10 of file MeanObliquity.cpp.

Here is the caller graph for this function:



5.112 MeanObliquity.cpp

Go to the documentation of this file.

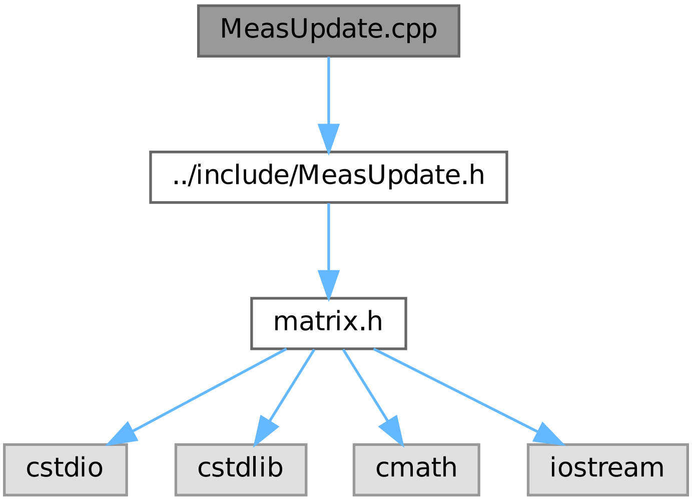
```
00001 #include "../include/MeanObliquity.h"
00002 #include "../include/SAT_Const.h"
00003
00010     double MeanObliquity (double Mjd_TT){
00011
00012
00013         double T = (Mjd_TT-MJD_J2000)/36525;
00014
00015         return Rad *( 84381.448/3600-(46.8150+(0.00059-0.001813*T)*T)*T/3600 );
00016     }
```

5.113 MeasUpdate.cpp File Reference

El archivo contiene las implementaciones de MeasUpdate.h.

```
#include "../include/MeasUpdate.h"
```

Include dependency graph for MeasUpdate.cpp:



Functions

- tuple< Matrix &, Matrix &, Matrix & > MeasUpdate (Matrix x, double z, double g, double s, Matrix G, Matrix P, int n)

5.113.1 Detailed Description

El archivo contiene las implementaciones de MeasUpdate.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file MeasUpdate.cpp.

5.113.2 Function Documentation

5.113.2.1 MeasUpdate()

```
tuple< Matrix &, Matrix &, Matrix & > MeasUpdate (
            Matrix x,
            double z,
            double g,
            double s,
            Matrix G,
            Matrix P,
            int n)
```

Parameters

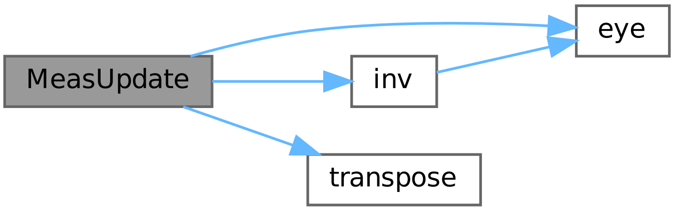
x	Matrix n∗1
z	double
g	double
s	double
G	Matrix 1∗n
P	Matrix n∗n
n	int

Returns

tupla de 3 Matrix

Definition at line 8 of file MeasUpdate.cpp.

Here is the call graph for this function:



5.114 MeasUpdate.cpp

Go to the documentation of this file.

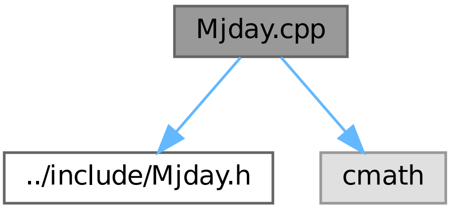
```
00001 #include "../include/MeasUpdate.h"
00008     tuple<Matrix&,Matrix&,Matrix&> MeasUpdate(Matrix x, double z,double g,double s,Matrix G,Matrix P,
      int n){
00009
00010             if(x.n_row<x.n_column){
00011                 x=transpose(x);
00012             }
00013         Matrix Inv_W(1);Inv_W(1) = s*s;
00014
00015         Matrix &K = P*transpose(G)*inv(Inv_W+G*P*transpose(G));
00016
00017         Matrix &nx = x + K*(z-g);
00018
00019         Matrix &nP = (eye(n)-K*G)*P;
00020
00021         return tie(K, nx, nP);
00022
00023     }
```

5.115 Mjday.cpp File Reference

El archivo contiene las implementaciones de Mjday.h.

```
#include "../include/Mjday.h"
#include <cmath>
```

Include dependency graph for Mjday.cpp:



Functions

- double Mjday (int yr, int mon, int day, int hr, int min, int sec)

5.115.1 Detailed Description

El archivo contiene las implementaciones de Mjday.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Mjday.cpp.

5.115.2 Function Documentation

5.115.2.1 Mjday()

```
double Mjday (
             int yr,
             int mon,
             int day,
             int hr = 0,
             int min = 0,
             int sec = 0)
```

Parameters

year	- year
mon	- month

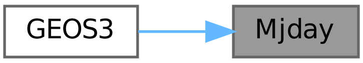
day	- day
hr	- universal time hour
min	- universal time min
sec	- universal time sec

Returns

Modified julian date

Definition at line 9 of file Mjday.cpp.

Here is the caller graph for this function:



5.116 Mjday.cpp

Go to the documentation of this file.

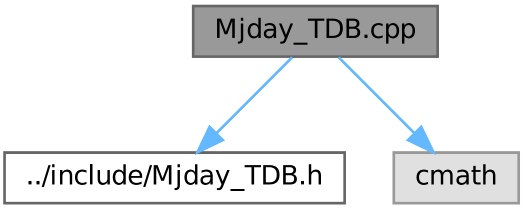
```
00001 #include "../include/Mjday.h"
00002 #include <cmath>
00009     double Mjday(int yr, int mon,int day,int hr,int min,int sec){
00010
00011 double jd = 367.0 * yr- floor( (7 * (yr + floor( (mon + 9) / 12.0) ) ) * 0.25 )+ floor( 275 * mon /
      9.0 ) + day + 1721013.5 + ( (sec/60.0 + min ) / 60.0 + hr ) / 24.0;
00012
00013 return jd-2400000.5;
00014
00015     }
```

5.117 Mjday_TDB.cpp File Reference

El archivo contiene las implementaciones de Mjday_TDB.h.

```
#include "../include/Mjday_TDB.h"
#include <cmath>
```

Include dependency graph for Mjday_TDB.cpp:



Functions

- double Mjday_TDB (double Mjd_TT)

5.117.1 Detailed Description

El archivo contiene las implementaciones de Mjday_TDB.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Mjday_TDB.cpp.

5.117.2 Function Documentation

5.117.2.1 Mjday_TDB()

```
double Mjday_TDB (
            double Mjd_TT)
```

Parameters

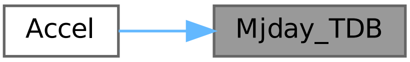
Mjd_TT	- Modified julian date (TT)

Returns

Modified julian date (TDB)

Definition at line 10 of file Mjday_TDB.cpp.

Here is the caller graph for this function:



5.118 Mjday_TDB.cpp

Go to the documentation of this file.

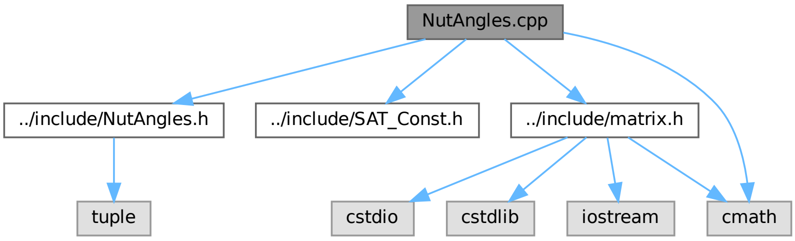
```
00001 #include "../include/Mjday_TDB.h"
00002 #include <cmath>
00003
00010     double Mjday_TDB(double Mjd_TT){
00011         double T_TT = (Mjd_TT - 51544.5)/36525;
00012
00013 return Mjd_TT + ( 0.001658*sin(628.3076*T_TT + 6.2401)+ 0.000022*sin(575.3385*T_TT+4.2970)+
      0.000014*sin(1256.6152*T_TT + 6.1969)+
00014     0.000005*sin(606.9777*T_TT+4.0212)+ 0.000005*sin(52.9691*T_TT+0.4444)+
      0.000002*sin(21.3299*T_TT+5.5431) +   0.000010*sin(628.3076*T_TT+4.2490) )/86400;
00015     }
```

5.119 NutAngles.cpp File Reference

El archivo contiene las implementaciones de NutAngles.h.

```
#include "../include/NutAngles.h"
#include "../include/SAT_Const.h"
#include "../include/matrix.h"
#include <cmath>
```

Include dependency graph for NutAngles.cpp:



Functions

- tuple< double, double > NutAngles (double Mjd_TT)

5.119.1 Detailed Description

El archivo contiene las implementaciones de NutAngles.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file NutAngles.cpp.

5.119.2 Function Documentation

5.119.2.1 NutAngles()

```
tuple< double, double > NutAngles (
            double Mjd_TT)
```

Nutation in longitude and obliquity

Parameters

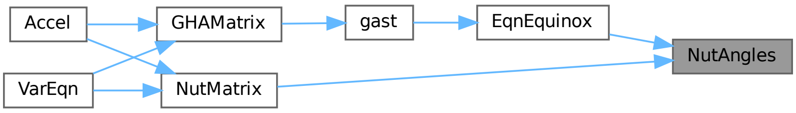
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

tupla < dpsi,deps> Nutation Angles

Definition at line 11 of file NutAngles.cpp.

Here is the caller graph for this function:



5.120 NutAngles.cpp

Go to the documentation of this file.

```
#include "../include/NutAngles.h"
#include "../include/SAT_Const.h"
#include "../include/matrix.h"
#include <cmath>
    tuple<double,double> NutAngles (double Mjd_TT){
        double T  = (Mjd_TT-MJD_J2000)/36525;
        double T2 = T*T;
        double T3 = T2*T;
        double rev = 360*3600; // arcsec/revolution

        int N_coeff = 106;
        Matrix C(106,9);
        double values[106][9]={
         // l  l' F  D Om    dpsi     *T     deps     *T

{       0, 0, 0, 0, 1,-1719960,-1742,  920250,   89},   //   1
{       0, 0, 0, 0, 2,   20620,    2,   -8950,    5},   //   2
{      -2, 0, 2, 0, 1,     460,    0,    -240,    0},   //   3
{       2, 0,-2, 0, 0,     110,    0,       0,    0},   //   4
{      -2, 0, 2, 0, 2,     -30,    0,      10,    0},   //   5
{       1,-1, 0,-1, 0,     -30,    0,       0,    0},   //   6
{       0,-2, 2,-2, 1,     -20,    0,      10,    0},   //   7
{       2, 0,-2, 0, 1,      10,    0,       0,    0},   //   8
{       0, 0, 2,-2, 2, -131870,  -16,   57360,  -31},   //   9
{       0, 1, 0, 0, 0,   14260,  -34,     540,   -1},   //  10
{       0, 1, 2,-2, 2,   -5170,   12,    2240,   -6},   //  11
{       0,-1, 2,-2, 2,    2170,   -5,    -950,    3},   //  12
{       0, 0, 2,-2, 1,    1290,    1,    -700,    0},   //  13
{       2, 0, 0,-2, 0,     480,    0,      10,    0},   //  14
{       0, 0, 2,-2, 0,    -220,    0,       0,    0},   //  15
{       0, 2, 0, 0, 0,     170,   -1,       0,    0},   //  16
{       0, 1, 0, 0, 1,    -150,    0,      90,    0},   //  17
{       0, 2, 2,-2, 2,    -160,    1,      70,    0},   //  18
{       0,-1, 0, 0, 1,    -120,    0,      60,    0},   //  19
{      -2, 0, 0, 2, 1,     -60,    0,      30,    0},   //  20
{       0,-1, 2,-2, 1,     -50,    0,      30,    0},   //  21
{       2, 0, 0,-2, 1,      40,    0,     -20,    0},   //  22
{       0, 1, 2,-2, 1,      40,    0,     -20,    0},   //  23
{       1, 0, 0,-1, 0,     -40,    0,       0,    0},   //  24
{       2, 1, 0,-2, 0,      10,    0,       0,    0},   //  25
{       0, 0,-2, 2, 1,      10,    0,       0,    0},   //  26
{       0, 1,-2, 2, 0,     -10,    0,       0,    0},   //  27
{       0, 1, 0, 0, 2,      10,    0,       0,    0},   //  28
{      -1, 0, 0, 1, 1,      10,    0,       0,    0},   //  29
{       0, 1, 2,-2, 0,     -10,    0,       0,    0},   //  30
{       0, 0, 2, 0, 2,  -22740,   -2,    9770,   -5},   //  31
{       1, 0, 0, 0, 0,    7120,    1,     -70,    0},   //  32
{       0, 0, 2, 0, 1,   -3860,   -4,    2000,    0},   //  33
{       1, 0, 2, 0, 2,   -3010,    0,    1290,   -1},   //  34
{       1, 0, 0,-2, 0,   -1580,    0,     -10,    0},   //  35
{      -1, 0, 2, 0, 2,    1230,    0,    -530,    0},   //  36
{       0, 0, 0, 2, 0,     630,    0,     -20,    0},   //  37
{       1, 0, 0, 0, 1,     630,    1,    -330,    0},   //  38
{      -1, 0, 0, 0, 1,    -580,   -1,     320,    0},   //  39
{      -1, 0, 2, 2, 2,    -590,    0,     260,    0},   //  40
{       1, 0, 2, 0, 1,    -510,    0,     270,    0},   //  41
{       0, 0, 2, 2, 2,    -380,    0,     160,    0},   //  42
{       2, 0, 0, 0, 0,     290,    0,     -10,    0},   //  43
{       1, 0, 2,-2, 2,     290,    0,    -120,    0},   //  44
{       2, 0, 2, 0, 2,    -310,    0,     130,    0},   //  45
{       0, 0, 2, 0, 0,     260,    0,     -10,    0},   //  46
{      -1, 0, 2, 0, 1,     210,    0,    -100,    0},   //  47
{      -1, 0, 0, 2, 1,     160,    0,     -80,    0},   //  48
{       1, 0, 0,-2, 1,    -130,    0,      70,    0},   //  49
{      -1, 0, 2, 2, 1,    -100,    0,      50,    0},   //  50
{       1, 1, 0,-2, 0,     -70,    0,       0,    0},   //  51
{       0, 1, 2, 0, 2,      70,    0,     -30,    0},   //  52
{       0,-1, 2, 0, 2,     -70,    0,      30,    0},   //  53
{       1, 0, 2, 2, 2,     -80,    0,      30,    0},   //  54
{       1, 0, 0, 2, 0,      60,    0,       0,    0},   //  55
{       2, 0, 2,-2, 2,      60,    0,     -30,    0},   //  56
{       0, 0, 0, 2, 1,     -60,    0,      30,    0},   //  57
{       0, 0, 2, 2, 1,     -70,    0,      30,    0},   //  58
{       1, 0, 2,-2, 1,      60,    0,     -30,    0},   //  59
{       0, 0, 0,-2, 1,     -50,    0,      30,    0},   //  60
{       1,-1, 0, 0, 0,      50,    0,       0,    0},   //  61
{       2, 0, 2, 0, 1,     -50,    0,      30,    0},   //  62
{       0, 1, 0,-2, 0,     -40,    0,       0,    0},   //  63
{       1, 0,-2, 0, 0,      40,    0,       0,    0},   //  64
{       0, 0, 0, 1, 0,     -40,    0,       0,    0},   //  65
{       1, 1, 0, 0, 0,     -30,    0,       0,    0},   //  66
{       1, 0, 2, 0, 0,      30,    0,       0,    0},   //  67
{       1,-1, 2, 0, 2,     -30,    0,      10,     0},   //  68
{      -1,-1, 2, 2, 2,     -30,    0,      10,     0},   //  69
{      -2, 0, 0, 0, 1,     -20,    0,      10,     0},   //  70
{       3, 0, 2, 0, 2,     -30,    0,      10,     0},   //  71
{       0,-1, 2, 2, 2,     -30,    0,      10,     0},   //  72
{       1, 1, 2, 0, 2,      20,    0,     -10,     0},   //  73
{      -1, 0, 2,-2, 1,     -20,    0,      10,     0},   //  74
{       2, 0, 0, 0, 1,      20,    0,     -10,     0},   //  75
{       1, 0, 0, 0, 2,     -20,    0,      10,     0},   //  76
{       3, 0, 0, 0, 0,      20,    0,       0,     0},   //  77
{       0, 0, 2, 1, 2,      20,    0,     -10,     0},   //  78
{      -1, 0, 0, 0, 2,      10,    0,     -10,     0},   //  79
{       1, 0, 0,-4, 0,     -10,    0,       0,     0},   //  80
{      -2, 0, 2, 2, 2,      10,    0,     -10,     0},   //  81
{      -1, 0, 2, 4, 2,     -20,    0,      10,     0},   //  82
{       2, 0, 0,-4, 0,     -10,    0,       0,     0},   //  83
{       1, 1, 2,-2, 2,      10,    0,     -10,     0},   //  84
{       1, 0, 2, 2, 1,     -10,    0,      10,     0},   //  85
{      -2, 0, 2, 4, 2,     -10,    0,      10,     0},   //  86
{      -1, 0, 4, 0, 2,      10,    0,       0,     0},   //  87
{       1,-1, 0,-2, 0,      10,    0,       0,     0},   //  88
{       2, 0, 2,-2, 1,      10,    0,     -10,     0},   //  89
{       2, 0, 2, 2, 2,     -10,    0,       0,     0},   //  90
{       1, 0, 0, 2, 1,     -10,    0,       0,     0},   //  91
{       0, 0, 4,-2, 2,      10,    0,       0,     0},   //  92
{       3, 0, 2,-2, 2,      10,    0,       0,     0},   //  93
{       1, 0, 2,-2, 0,     -10,    0,       0,     0},   //  94
{       0, 1, 2, 0, 1,      10,    0,       0,     0},   //  95
{      -1,-1, 0, 2, 1,      10,    0,       0,     0},   //  96
{       0, 0,-2, 0, 1,     -10,    0,       0,     0},   //  97
{       0, 0, 2,-1, 2,     -10,    0,       0,     0},   //  98
{       0, 1, 0, 2, 0,     -10,    0,       0,     0},   //  99
{       1, 0,-2,-2, 0,     -10,    0,       0,     0},   // 100
{       0,-1, 2, 0, 1,     -10,    0,       0,     0},   // 101
{       1, 1, 0,-2, 1,     -10,    0,       0,     0},   // 102
{       1, 0,-2, 2, 0,     -10,    0,       0,     0},   // 103
{       2, 0, 0, 2, 0,      10,    0,       0,     0},   // 104
{       0, 0, 2, 4, 2,     -10,    0,       0,     0},   // 105
{       0, 1, 0, 1, 0,      10,    0,       0,     0}   // 106
         };
         for (int i = 0; i < C.n_row; i++) {
            for (int j = 0; j < C.n_column; j++) {
                C.data[i][j] = values[i][j];
            }
        }

        // Mean arguments of luni-solar motion

        //   l   mean anomaly of the Moon
        //   l’  mean anomaly of the Sun
        //   F   mean argument of latitude
        //   D   mean longitude elongation of the Moon from the Sun
        //   Om  mean longitude of the ascending node

        double l  = fmod (  485866.733 + (1325.0*rev +  715922.633)*T + 31.310*T2 + 0.064*T3, rev );
        double lp = fmod ( 1287099.804 + (  99.0*rev + 1292581.224)*T -  0.577*T2 - 0.012*T3, rev );
        double F  = fmod (  335778.877 + (1342.0*rev +  295263.137)*T - 13.257*T2 + 0.011*T3, rev );
        double D  = fmod ( 1072261.307 + (1236.0*rev + 1105601.328)*T -  6.891*T2 + 0.019*T3, rev );
        double Om = fmod (  450160.280 - (   5.0*rev +  482890.539)*T +  7.455*T2 + 0.008*T3, rev );

        // Nutation in longitude and obliquity [rad]

        double dpsi = 0;
        double deps = 0;
        double arg;
        for (int i=1;i<=N_coeff;i++){
          arg  =  ( C(i,1)*l+C(i,2)*lp+C(i,3)*F+C(i,4)*D+C(i,5)*Om )/Arcs;
          dpsi = dpsi + ( C(i,6)+C(i,7)*T ) * sin(arg);
          deps = deps + ( C(i,8)+C(i,9)*T ) * cos(arg);
        }
        dpsi = 1.0e-5 * dpsi/Arcs;
        deps = 1.0e-5 * deps/Arcs;
        return tie(dpsi,deps);
    }
```

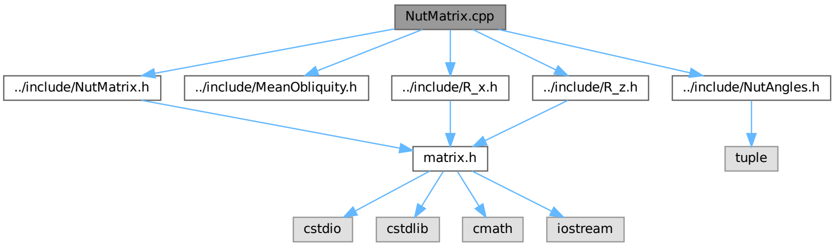
5.121 NutMatrix.cpp File Reference

El archivo contiene las implementaciones de NutMatrix.h.

```
#include "../include/NutMatrix.h"
#include "../include/MeanObliquity.h"
```

```
#include "../include/NutAngles.h"
#include "../include/R_x.h"
#include "../include/R_z.h"
```

Include dependency graph for NutMatrix.cpp:



Functions

- Matrix & NutMatrix (double Mjd_TT)

5.121.1 Detailed Description

El archivo contiene las implementaciones de NutMatrix.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file NutMatrix.cpp.

5.121.2 Function Documentation

5.121.2.1 NutMatrix()

```
Matrix & NutMatrix (
            double Mjd_TT)
```

Transformation from mean to true equator and equinox

Parameters

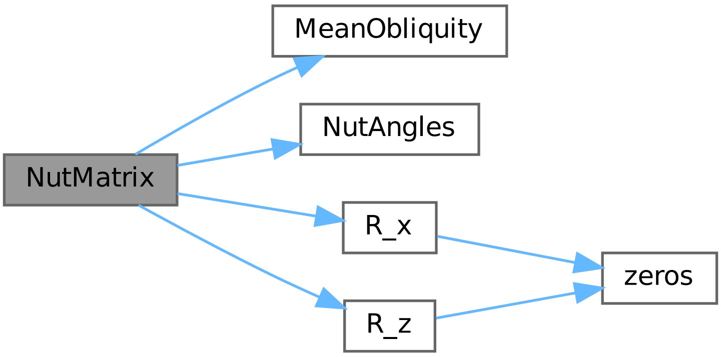
Mjd_TT	Modified Julian Date (Terrestrial Time)

Returns

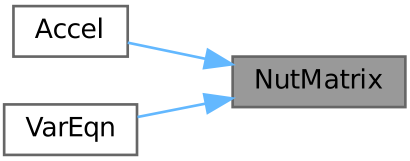
NutMat Nutation matrix

Definition at line 13 of file NutMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.122 NutMatrix.cpp

Go to the documentation of this file.

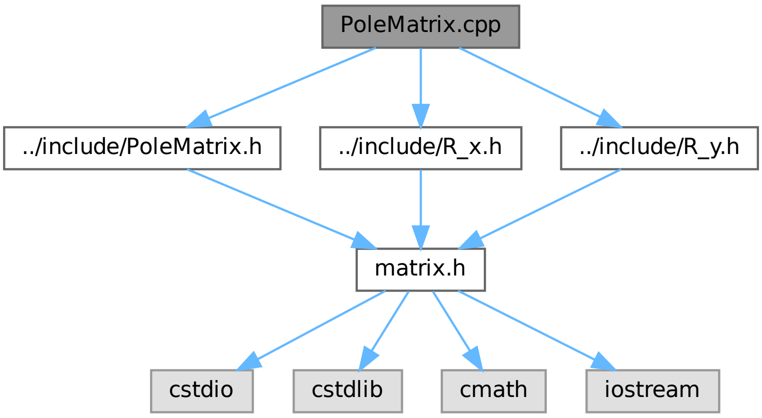
```
00001 #include "../include/NutMatrix.h"
00002 #include "../include/MeanObliquity.h"
00003 #include "../include/NutAngles.h"
00004 #include "../include/R_x.h"
00005 #include "../include/R_z.h"
00006
00013     Matrix& NutMatrix (double Mjd_TT){
00014
00015 double eps = MeanObliquity (Mjd_TT);
00016
00017 auto [dpsi, deps] = NutAngles (Mjd_TT);
00018
00019 return R_x(-eps-deps)*R_z(-dpsi)*R_x(+eps);
00020 }
```

5.123 PoleMatrix.cpp File Reference

El archivo contiene las implementaciones de PoleMatrix.h.

```
#include "../include/PoleMatrix.h"
#include "../include/R_x.h"
#include "../include/R_y.h"
```

Include dependency graph for PoleMatrix.cpp:



Functions

- Matrix & PoleMatrix (double xp, double yp)

5.123.1 Detailed Description

El archivo contiene las implementaciones de PoleMatrix.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file PoleMatrix.cpp.

5.123.2 Function Documentation

5.123.2.1 PoleMatrix()

```
Matrix & PoleMatrix (
            double xp,
            double yp)
```

Transformation from pseudo Earth-fixed to Earth-fixed coordinates for a given date

Parameters

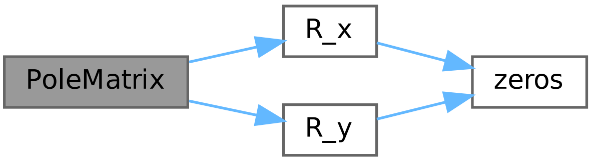
Pole	coordinte(xp,yp)

Returns

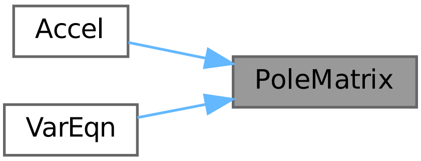
PoleMat Pole matrix

Definition at line 11 of file PoleMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.124 PoleMatrix.cpp

Go to the documentation of this file.

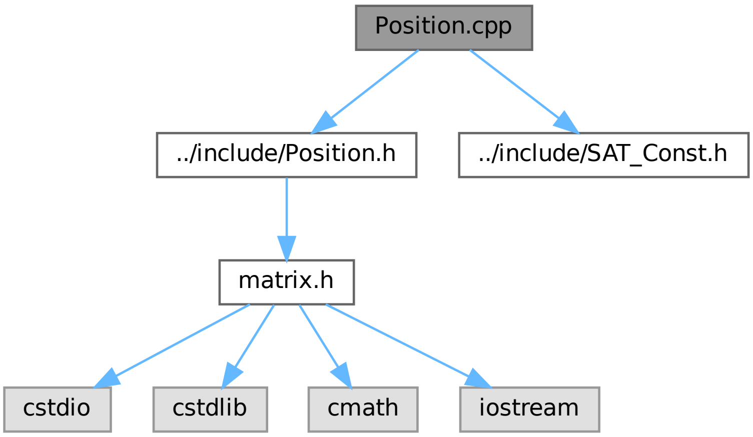
```
00001 #include "../include/PoleMatrix.h"
00002 #include "../include/R_x.h"
00003 #include "../include/R_y.h"
00004
00011     Matrix& PoleMatrix (double xp,double yp){
00012 return R_y(-xp) * R_x(-yp);
00013 }
```

5.125 Position.cpp File Reference

El archivo contiene las implementaciones de Position.h.

```
#include "../include/Position.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for Position.cpp:



Functions

- Matrix & Position (double lon, double lat, double h)

5.125.1 Detailed Description

El archivo contiene las implementaciones de Position.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file Position.cpp.

5.125.2 Function Documentation

5.125.2.1 Position()

```
Matrix & Position (
            double lon,
            double lat,
            double h)
```

Parameters

lon	longitude [rad]
lat	latitude [rad]
h	altitude [m]

Returns

Position vector (r [m]) from geodetic coordinates (Longitude [rad],latitude [rad], altitude [m])

Definition at line 10 of file Position.cpp.

5.126 Position.cpp

Go to the documentation of this file.

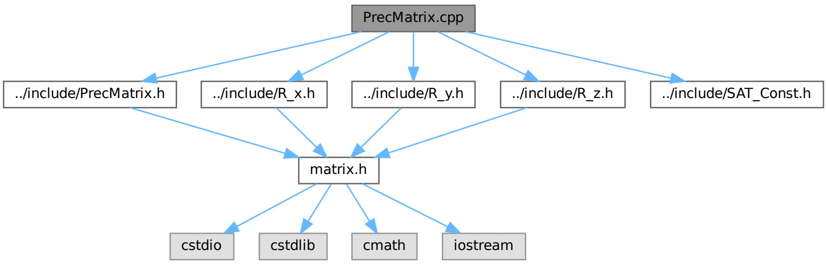
```
00001 #include "../include/Position.h"
00002 #include "../include/SAT_Const.h"
00003
00010     Matrix& Position(double lon,double lat,double h){
00011         double R_equ = R_Earth;
00012         double f     = f_Earth;
00013
00014         double e2     = f*(2.0-f);
00015         double CosLat = cos(lat);
00016         double SinLat = sin(lat);
00017
00018         double N = R_equ / sqrt(1.0-e2*SinLat*SinLat);
00019
00020         Matrix *r=new Matrix(3);
00021
00022         (*r)(1)  =  (         N+h)*CosLat*cos(lon);
00023         (*r)(2)  =  (         N+h)*CosLat*sin(lon);
00024         (*r)(3)  =  ((1.0-e2)*N+h)*SinLat;
00025
00026         return *r;
00027     }
```

5.127 PrecMatrix.cpp File Reference

El archivo contiene las implementaciones de PrecMatrix.h.

```
#include "../include/PrecMatrix.h"
#include "../include/R_x.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
#include "../include/SAT_Const.h"
```

Include dependency graph for PrecMatrix.cpp:



Functions

- Matrix & PrecMatrix (double Mjd_1, double Mjd_2)

5.127.1 Detailed Description

El archivo contiene las implementaciones de PrecMatrix.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file PrecMatrix.cpp.

5.127.2 Function Documentation

5.127.2.1 PrecMatrix()

```
Matrix & PrecMatrix (
            double Mjd_1,
            double Mjd_2)
```

Precession transformation of equatorial coordinates

Parameters

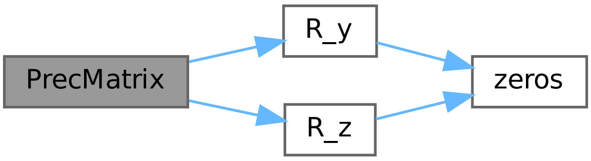
Mjd←↩ _1	Epoch given (Modified Julian Date TT)
MjD←↩ _2	Epoch to precess to (Modified Julian Date TT)

Returns

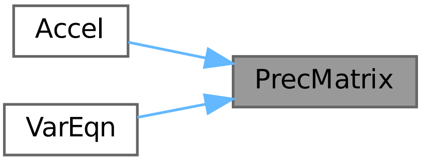
PrecMat Precession transformation matrix

Definition at line 13 of file PrecMatrix.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.128 PrecMatrix.cpp

Go to the documentation of this file.

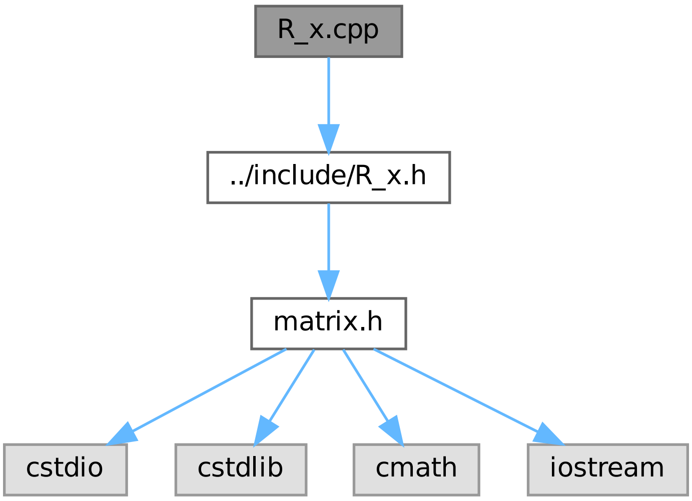
```
#include "../include/PrecMatrix.h"
#include "../include/R_x.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
#include "../include/SAT_Const.h"

    Matrix& PrecMatrix (double Mjd_1, double Mjd_2){
        double zeta,z,theta,dT,T;
        T  = (Mjd_1-MJD_J2000)/36525;
        dT = (Mjd_2-Mjd_1)/36525;


        zeta  =  ( (2306.2181+(1.39656-0.000139*T)*T)+ ((0.30188-0.000344*T)+0.017998*dT)*dT )*dT/Arcs;
        z     =  zeta + ( (0.79280+0.000411*T)+0.000205*dT)*dT*dT/Arcs;
        theta =  ( (2004.3109-(0.85330+0.000217*T)*T)-((0.42665+0.000217*T)+0.041833*dT)*dT )*dT/Arcs;

        return R_z(-z) * R_y(theta) * R_z(-zeta);
}
```

5.129 R_x.cpp File Reference

El archivo contiene las implementaciones de R_x.h.

```
#include "../include/R_x.h"
```

Include dependency graph for R_x.cpp:



Functions

- Matrix & R_x (double angle)

5.129.1 Detailed Description

El archivo contiene las implementaciones de R_x.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_x.cpp.

5.129.2 Function Documentation

5.129.2.1 R_x()

```
Matrix & R_x (
            double angle)
```

Parameters

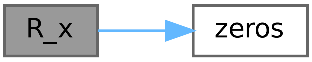
angle	angulo de rotacion

Returns

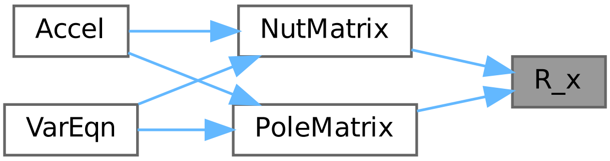
Matrix resultado

Definition at line 9 of file R_x.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.130 R_x.cpp

Go to the documentation of this file.

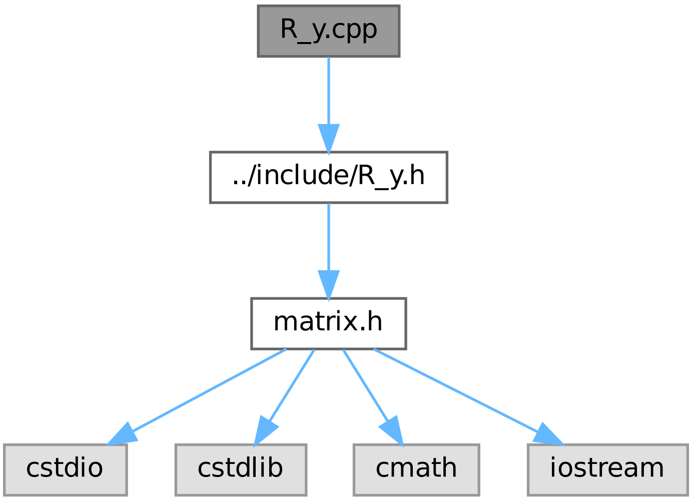
```
00001 #include "../include/R_x.h"
00002
00009     Matrix& R_x(double angle){
00010         double C = cos(angle);
00011         double S = sin(angle);
00012         Matrix *rotmat=&zeros(3,3);
00013
00014         (*rotmat)(1,1) = 1.0;   (*rotmat)(1,2) =    0.0;   (*rotmat)(1,3) = 0.0;
00015         (*rotmat)(2,1) = 0.0;   (*rotmat)(2,2) =      C;   (*rotmat)(2,3) =   S;
00016         (*rotmat)(3,1) = 0.0;   (*rotmat)(3,2) = -1.0*S;   (*rotmat)(3,3) =   C;
00017         return *rotmat;
00018     }
```

5.131 R_y.cpp File Reference

El archivo contiene las implementaciones de R_y.h.

```
#include "../include/R_y.h"
```

Include dependency graph for R_y.cpp:



Functions

- Matrix & R_y (double angle)

5.131.1 Detailed Description

El archivo contiene las implementaciones de R_y.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_y.cpp.

5.131.2 Function Documentation

5.131.2.1 R_y()

```
Matrix & R_y (
            double angle)
```

Parameters

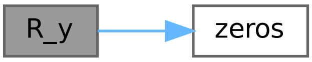
angle	angulo de rotacion

Returns

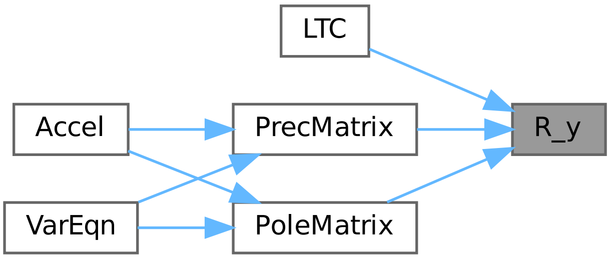
Matrix resultado

Definition at line 9 of file R_y.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.132 R_y.cpp

Go to the documentation of this file.

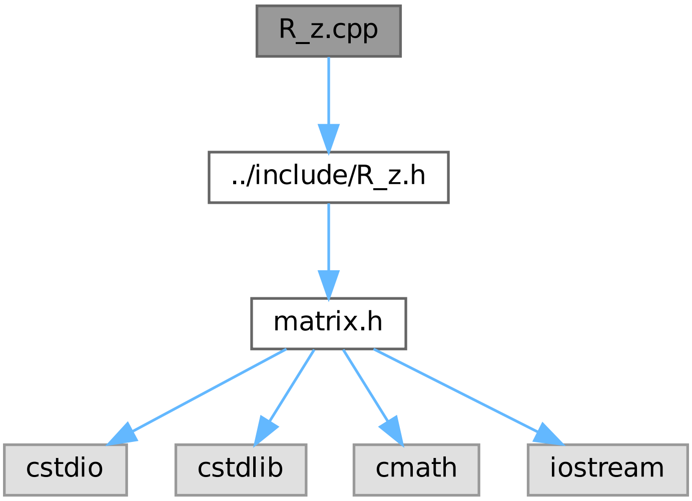
```
#include "../include/R_y.h"

    Matrix& R_y(double angle){
        double C = cos(angle);
        double S = sin(angle);
        Matrix *rotmat=&zeros(3,3);

        (*rotmat)(1,1) =    C;   (*rotmat)(1,2) = 0.0;   (*rotmat)(1,3) = -1.0*S;
        (*rotmat)(2,1) = 0.0;   (*rotmat)(2,2) = 1.0;   (*rotmat)(2,3) =    0.0;
        (*rotmat)(3,1) =    S;   (*rotmat)(3,2) = 0.0;   (*rotmat)(3,3) =      C;
        return *rotmat;
    }
```

5.133 R_z.cpp File Reference

El archivo contiene las implementaciones de R_z.h.

```
#include "../include/R_z.h"
```

Include dependency graph for R_z.cpp:



Functions

- Matrix & R_z (double angle)

5.133.1 Detailed Description

El archivo contiene las implementaciones de R_z.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file R_z.cpp.

5.133.2 Function Documentation

5.133.2.1 R_z()

```
Matrix & R_z (
            double angle)
```

Parameters

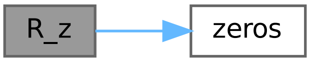
angle	angulo de rotacion

Returns

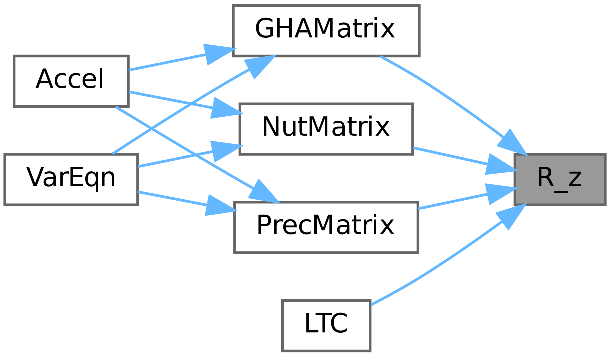
Matrix resultado

Definition at line 9 of file R_z.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.134 R_z.cpp

Go to the documentation of this file.

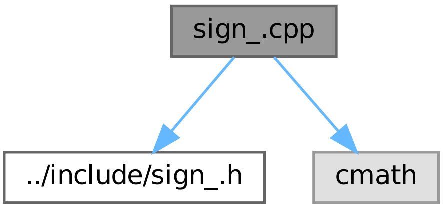
```
#include "../include/R_z.h"

    Matrix& R_z(double angle){
        double C = cos(angle);
        double S = sin(angle);
        Matrix *rotmat=&zeros(3,3);

        (*rotmat)(1,1) =      C;  (*rotmat)(1,2) =   S;  (*rotmat)(1,3) = 0.0;
        (*rotmat)(2,1) = -1.0*S;  (*rotmat)(2,2) =   C;  (*rotmat)(2,3) = 0.0;
        (*rotmat)(3,1) =    0.0;  (*rotmat)(3,2) = 0.0;  (*rotmat)(3,3) = 1.0;
        return *rotmat;
    }
```

5.135 sign_.cpp File Reference

El archivo contiene las implementaciones de sign_.h.

```
#include "../include/sign_.h"
#include <cmath>
```

Include dependency graph for sign_.cpp:



Functions

- double sign_ (double a, double b)

5.135.1 Detailed Description

El archivo contiene las implementaciones de sign_.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file sign_.cpp.

5.135.2 Function Documentation

5.135.2.1 sign_()

```
double sign_ (
            double a,
            double b)
```

Parameters

a	double
b	double

Returns

absolute value of a with sign of b

Definition at line 10 of file sign_.cpp.

5.136 sign_.cpp

Go to the documentation of this file.

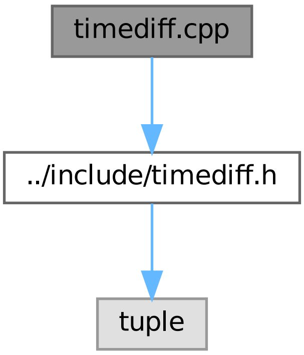
```
#include "../include/sign_.h"
#include <cmath>

    double sign_(double a, double b){
    if (b>=0.0){
        return abs(a);}
    else{
        return -abs(a);}
    }
```

5.137 timediff.cpp File Reference

El archivo contiene las implementaciones de timediff.h.

```
#include "../include/timediff.h"
```

Include dependency graph for timediff.cpp:



Functions

- tuple< double, double, double, double, double > timediff (double UT1_UTC, double TAI_UTC)

5.137.1 Detailed Description

El archivo contiene las implementaciones de timediff.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file timediff.cpp.

5.137.2 Function Documentation

5.137.2.1 timediff()

```
tuple< double, double, double, double, double > timediff (
              double UT1_UTC,
              double TAI_UTC)
```

Time differences [s]

Parameters

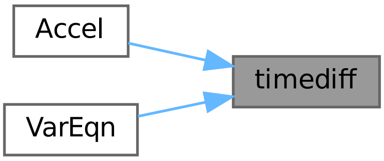
UT1_UTC	double
TAI_UTC	double

Returns

tupla <UT1_TAI, UTC_GPS, UT1_GPS, TT_UTC, GPS_UTC>

Definition at line 9 of file timediff.cpp.

Here is the caller graph for this function:



5.138 timediff.cpp

Go to the documentation of this file.

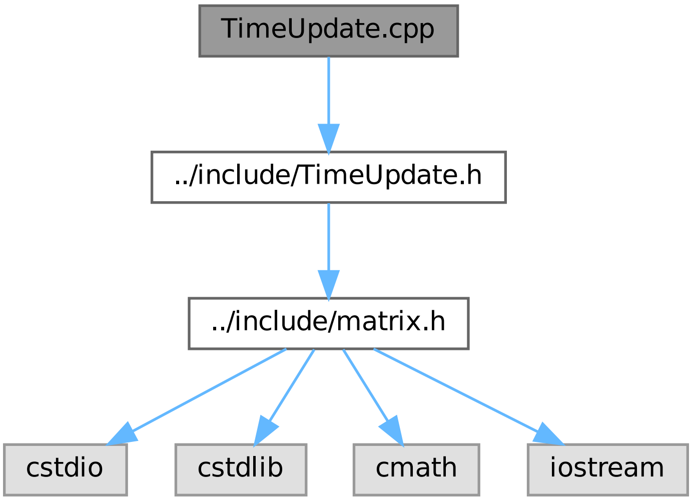
```
#include "../include/timediff.h"

    tuple<double,double,double,double,double>  timediff(double UT1_UTC,double TAI_UTC){

        double TT_TAI  = +32.184;          // TT-TAI time difference [s]

        double GPS_TAI = -19.0;            // GPS-TAI time difference [s]
/*
        double TT_GPS  =  TT_TAI-GPS_TAI;  // TT-GPS time difference [s]

        double TAI_GPS = -GPS_TAI;         // TAI-GPS time difference [s]
*/
        double UT1_TAI = UT1_UTC-TAI_UTC;  // UT1-TAI time difference [s]

        double UTC_TAI = -TAI_UTC;         // UTC-TAI time difference [s]

        double UTC_GPS = UTC_TAI-GPS_TAI;  // UTC_GPS time difference [s]

        double UT1_GPS = UT1_TAI-GPS_TAI;  // UT1-GPS time difference [s]

        double TT_UTC  = TT_TAI-UTC_TAI;   //  TT-UTC time difference [s]

        double GPS_UTC = GPS_TAI-UTC_TAI;  // GPS-UTC time difference [s]

        return tie(UT1_TAI, UTC_GPS, UT1_GPS, TT_UTC, GPS_UTC);

    }
```

5.139 TimeUpdate.cpp File Reference

El archivo contiene las implementaciones de TimeUpdate.h.

```
#include "../include/TimeUpdate.h"
```

Include dependency graph for TimeUpdate.cpp:



Functions

- Matrix & TimeUpdate (Matrix P, Matrix Phi, Matrix Qdt)
- Matrix & TimeUpdate (Matrix P, Matrix Phi)

5.139.1 Detailed Description

El archivo contiene las implementaciones de TimeUpdate.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file TimeUpdate.cpp.

5.139.2 Function Documentation

5.139.2.1 TimeUpdate() [1/2]

```
Matrix & TimeUpdate (
            Matrix P,
            Matrix Phi)
```

Parameters

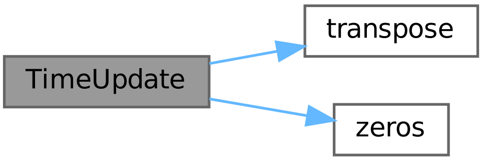
P	Matrix
Phi	Matrix

Returns

Matrix

Definition at line 15 of file TimeUpdate.cpp.

Here is the call graph for this function:



5.139.2.2 TimeUpdate() [2/2]

```
Matrix & TimeUpdate (
            Matrix P,
            Matrix Phi,
            Matrix Qdt)
```

Parameters

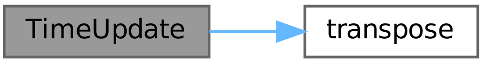
P	Matrix
Phi	Matrix
Qdt	Matrix

Returns

Matrix

Definition at line 9 of file TimeUpdate.cpp.

Here is the call graph for this function:



5.140 TimeUpdate.cpp

Go to the documentation of this file.

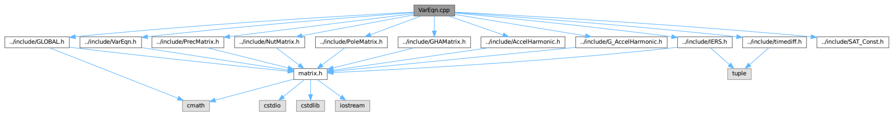
```
#include "../include/TimeUpdate.h"

    Matrix& TimeUpdate(Matrix P,Matrix Phi,Matrix Qdt){

        return Phi*P*transpose(Phi)+ Qdt;

    }

    Matrix& TimeUpdate(Matrix P,Matrix Phi){
        Matrix Qdt=zeros(P.n_column,P.n_row);
        return Phi*P*transpose(Phi)+ Qdt;

    }
```

5.141 VarEqn.cpp File Reference

El archivo contiene las implementaciones de VarEqn.h.

```
#include "../include/VarEqn.h"
#include "../include/IERS.h"
#include "../include/timediff.h"
#include "../include/GLOBAL.h"
#include "../include/SAT_Const.h"
#include "../include/PrecMatrix.h"
#include "../include/NutMatrix.h"
#include "../include/PoleMatrix.h"
#include "../include/GHAMatrix.h"
#include "../include/AccelHarmonic.h"
#include "../include/G_AccelHarmonic.h"
```

Include dependency graph for VarEqn.cpp:



Functions

- Matrix & VarEqn (double x, Matrix yPhi)

5.141.1 Detailed Description

El archivo contiene las implementaciones de VarEqn.h.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file VarEqn.cpp.

5.141.2 Function Documentation

5.141.2.1 VarEqn()

```
Matrix & VarEqn (
            double x,
            Matrix yPhi)
```

Computes the variational equations, i.e. the derivative of the state vector and the state transition matrix

Parameters

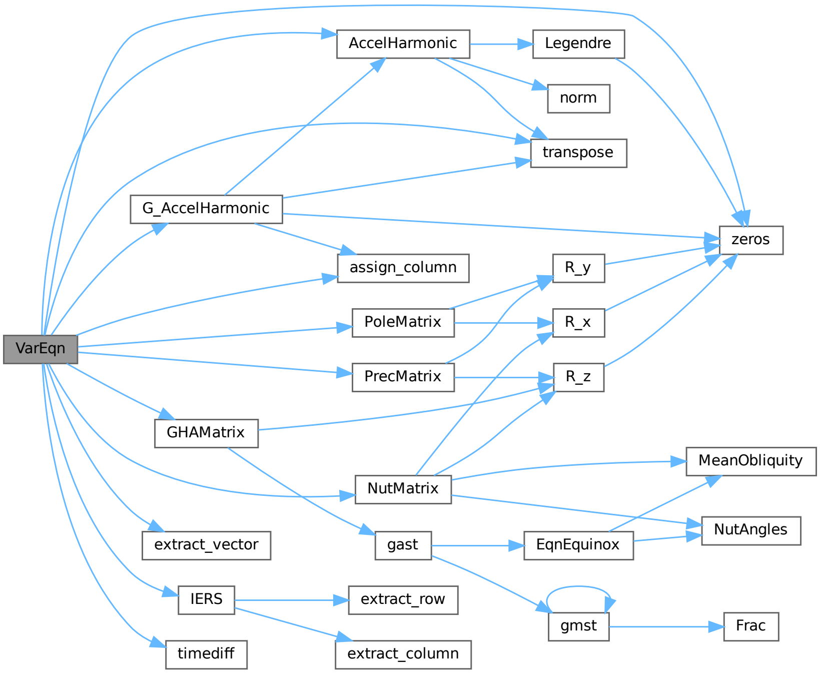
x	Time since epoch in [s]
yPhi	(6+36)-dim vector comprising the state vector (y) and the state transition matrix (Phi) in column wise storage order

Returns

yPhip Derivative of yPhi

Definition at line 18 of file VarEqn.cpp.

Here is the call graph for this function:



5.142 VarEqn.cpp

Go to the documentation of this file.

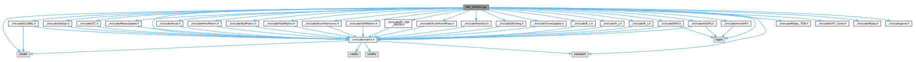
```
#include "../include/VarEqn.h"
#include "../include/IERS.h"
#include "../include/timediff.h"
#include "../include/GLOBAL.h"
#include "../include/SAT_Const.h"
#include "../include/PrecMatrix.h"
#include "../include/NutMatrix.h"
#include "../include/PoleMatrix.h"
#include "../include/GHAMatrix.h"
#include "../include/AccelHarmonic.h"
#include "../include/G_AccelHarmonic.h"
Matrix& VarEqn(double x, Matrix yPhi){
    if(yPhi.n_row<yPhi.n_column){
        yPhi=transpose(yPhi);
    }
    auto [x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC] = IERS(eopdata,AuxParam.Mjd_UTC,’l’);
    auto [UT1_TAI,UTC_GPS,UT1_GPS,TT_UTC,GPS_UTC] = timediff(UT1_UTC,TAI_UTC);
    double Mjd_UT1 = AuxParam.Mjd_TT + (UT1_UTC-TT_UTC)/86400;

// Transformation matrix
    Matrix P = PrecMatrix(MJD_J2000,AuxParam.Mjd_TT + x/86400);
    Matrix N = NutMatrix(AuxParam.Mjd_TT + x/86400);
    Matrix T = N * P;
    Matrix E = PoleMatrix(x_pole,y_pole) * GHAMatrix(Mjd_UT1) * T;

// State vector components
    Matrix r = extract_vector(yPhi,1,3);
    Matrix v = extract_vector(yPhi,4,6);
    Matrix Phi = zeros(6,6);
// State transition matrix
    for (int j=1;j<=6;j++){
        Phi=assign_column(Phi,extract_vector(yPhi,6*j+1,6*j+6),j);
    }

// Acceleration and gradient
    Matrix a = AccelHarmonic ( r, E, AuxParam.n, AuxParam.m );
    Matrix G = G_AccelHarmonic ( r, E, AuxParam.n, AuxParam.m );

// Time derivative of state transition matrix
    Matrix &yPhip = zeros(42,1);
    Matrix dfdy = zeros(6,6);

    for (int i=1;i<=3;i++){
        for (int j=1;j<=3;j++){
            dfdy(i,j) = 0.0;                 // dv/dr(i,j)
            dfdy(i+3,j) = G(i,j);            // da/dr(i,j)
            if ( i==j ){
                dfdy(i,j+3) = 1;}
                else{
                dfdy(i,j+3) = 0;             // dv/dv(i,j)
            }
            dfdy(i+3,j+3) = 0.0;             // da/dv(i,j)
        }
    }

    Matrix Phip = dfdy*Phi;

// Derivative of combined state vector and state transition matrix
    for (int i=1;i<=3;i++){
    yPhip(i)   = v(i);                 // dr/dt(i)
    yPhip(i+3) = a(i);                 // dv/dt(i)
}

for (int i=1;i<=6;i++){
    for (int j=1;j<=6;j++){
        yPhip(6*j+i) = Phip(i,j);     // dPhi/dt(i,j)
    }
}
return yPhip;
}
```

5.143 EKF_GEOS3.cpp File Reference

Es el archivo principal del programa.

```
#include <cmath>
#include <tuple>
#include <iostream>
#include "../include/matrix.h"
#include "../include/Accel.h"
#include "../include/PrecMatrix.h"
#include "../include/NutMatrix.h"
#include "../include/IERS.h"
#include "../include/timediff.h"
#include "../include/PoleMatrix.h"
#include "../include/AccelHarmonic.h"
#include "../include/GHAMatrix.h"
#include "../include/JPL_Eph_DE430.h"
#include "../include/GLOBAL.h"
#include "../include/AccelPointMass.h"
#include "../include/Mjday_TDB.h"
#include "../include/SAT_Const.h"
#include "../include/Position.h"
#include "../include/Mjday.h"
#include "../include/DEInteg.h"
#include "../include/TimeUpdate.h"
#include "../include/AzElPa.h"
#include "../include/R_x.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
#include "../include/gmst.h"
#include "../include/VarEqn.h"
#include "../include/LTC.h"
#include "../include/MeasUpdate.h"
```

Include dependency graph for EKF_GEOS3.cpp:



Functions

- int main ()

5.143.1 Detailed Description

Es el archivo principal del programa.

Author

Pedro Zhuzhan

Bug No known bugs

Definition in file EKF_GEOS3.cpp.

5.143.2 Function Documentation

5.143.2.1 main()

```
int main ()
```

Definition at line 36 of file EKF_GEOS3.cpp.

5.144 EKF_GEOS3.cpp

Go to the documentation of this file.

```
00001 #include <cmath>
00002 #include<tuple>
00003 #include <iostream>
00004 #include"../include/matrix.h"
00005 #include "../include/Accel.h"
00006 #include "../include/PrecMatrix.h"
00007 #include "../include/NutMatrix.h"
00008 #include "../include/IERS.h"
00009 #include "../include/timediff.h"
00010 #include "../include/PoleMatrix.h"
00011 #include "../include/AccelHarmonic.h"
00012 #include "../include/GHAMatrix.h"
00013 #include "../include/JPL_Eph_DE430.h"
00014 #include "../include/GLOBAL.h"
00015 #include "../include/AccelPointMass.h"
00016 #include "../include/Mjday_TDB.h"
00017 #include "../include/SAT_Const.h"
00018 #include "../include/Position.h"
00019 #include "../include/Mjday.h"
00020 #include "../include/DEInteg.h"
00021 #include "../include/TimeUpdate.h"
00022 #include "../include/AzElPa.h"
00023 #include "../include/R_x.h"
00024 #include "../include/R_y.h"
00025 #include "../include/R_z.h"
00026 #include "../include/gmst.h"
00027 #include "../include/VarEqn.h"
00028 #include "../include/LTC.h"
00029 #include "../include/MeasUpdate.h"
00036         int main(){
00037
00038     AuxParamLoad();
00039     eop19620101();
00040     GGM03S();
00041     DE430Coeff();
00042     GEOS3();
00043
00044     int i=0,j,ii,nobs = 46;
00045             double sigma_range,sigma_az,sigma_el,lat,lon,alt,Mjd1,Mjd2,Mjd3,Mjd0,Mjd_UTC=obs(9,1),
00046             n_eqn,theta,t_old,Mjd_TT,Dist,Mjd_UT1,UT1_TAI,UTC_GPS,UT1_GPS,TT_UTC,GPS_UTC,
00047             x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC,Azim, Elev,t;
00048
00049             Matrix Rs,Y0_apr,P,LT,yPhi,Phi,Y_true=zeros(6),Y0=zeros(6),U,Y_old,dDdY,dDds,r,s,dEdY,
00050             K, Y, dAds, dEds,dAdY;
00051             sigma_range = 92.5;
00052 sigma_az = 0.0224*Rad;
00053 sigma_el = 0.0139*Rad;
00054
00055
00056 lat = Rad*21.5748;
00057 lon = Rad*(-158.2706);
00058 alt = 300.20;
00059 Rs = transpose(Position(lon, lat, alt));
00060
00061 Mjd1 = obs(1,1);
00062 Mjd2 = obs(9,1);
00063 Mjd3 = obs(18,1);
00064 Matrix r2(3),v2(3);
00065 r2(1)=6221397.62857869;
00066 r2(2)=2867713.77965738;
00067 r2(3)=3006155.98509949;
00068
00069 v2(1)= 4645.04725161806;
00070 v2(2)=-2752.21591588204;
00071 v2(3)=-7507.99940987031;
```

```
//auto [r2,v2] = anglesg(obs(1,2),obs(9,2),obs(18,2),obs(1,3),obs(9,3),obs(18,3),Mjd1,Mjd2,Mjd3,Rs,Rs,Rs);



Y0_apr = union_vector(r2,v2);

Mjd0 = Mjday(1995,1,29,02,38,0);

AuxParam.Mjd_UTC = Mjd_UTC;

Mjd_UTC = obs(9,1);
n_eqn  = 6;

Y = DEInteg(Accel,0,-(obs(9,1)-Mjd0)*86400.0,1e-13,1e-6,6,Y0_apr);
P = zeros(6,6);
for (i=1;i<=3;i++){
    P(i,i)=1e8;}
for (i=4;i<=6;i++){
    P(i,i)=1e3;}

LT = LTC(lon,lat);

yPhi = zeros(42,1);
Phi  = zeros(6,6);


t = 0;

for (i=1;i<=nobs;i++){
    t_old = t;
    Y_old = Y;

    Mjd_UTC = obs(i,1);
    t       = (Mjd_UTC-Mjd0)*86400.0;

    tie (x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC) = IERS(eopdata,Mjd_UTC,'l');
    tie (UT1_TAI,UTC_GPS,UT1_GPS,TT_UTC,GPS_UTC) = timediff(UT1_UTC,TAI_UTC);
    Mjd_TT = Mjd_UTC + TT_UTC/86400;
    Mjd_UT1 = Mjd_TT + (UT1_UTC-TT_UTC)/86400.0;
    AuxParam.Mjd_UTC = Mjd_UTC;
    AuxParam.Mjd_TT = Mjd_TT;
    for ( ii=1;ii<=6;ii++){
        yPhi(ii) = Y_old(ii);
        for (j=1;j<=6;j++)  {
            if (ii==j) {
                yPhi(6*j+ii) = 1;
            }
            else{
                yPhi(6*j+ii) = 0;

            }
        }
    }
    yPhi = DEInteg (VarEqn,0,t-t_old,1e-13,1e-6,42,yPhi);

    for (j=1;j<=6;j++){
        Phi = assign_column(Phi,extract_vector(yPhi,6*j+1,6*j+6),j);

    }
    Y = DEInteg (Accel,0,t-t_old,1e-13,1e-6,6,Y_old);
    theta = gmst(Mjd_UT1);
    U = R_z(theta);
    r = extract_vector(Y,1,3);
    r=transpose(r);
    s = LT*(U*r-Rs);

    P = TimeUpdate(P, Phi);


    tie( Azim, Elev, dAds, dEds)= AzElPa(s);
    dAdY = union_vector(dAds*LT*U,zeros(1,3));


     tie( K, Y, P) = MeasUpdate ( Y, obs(i,2), Azim, sigma_az, dAdY, P, 6 );

    r = extract_vector(Y,1,3);
    r=transpose(r);
    s = LT*(U*r-Rs);
    tie( Azim, Elev, dAds, dEds)= AzElPa(s);


    dEdY = union_vector(dEds*LT*U,zeros(1,3));


     tie( K, Y, P) = MeasUpdate ( Y, obs(i,3), Elev, sigma_el, dEdY, P, 6 );

    r = extract_vector(Y,1,3);
    r=transpose(r);
    s = LT*(U*r-Rs);

    Dist = norm(s);
    dDds = transpose(s/Dist);
    dDdY = union_vector(dDds*LT*U,zeros(1,3));


     tie(K, Y,P) = MeasUpdate ( Y, obs(i,4), Dist, sigma_range, dDdY, P, 6 );
}

tie(x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC) = IERS(eopdata,obs(46,1),'l');
tie(UT1_TAI,UTC_GPS,UT1_GPS,TT_UTC,GPS_UTC) = timediff(UT1_UTC,TAI_UTC);
Mjd_TT = Mjd_UTC + TT_UTC/86400;
AuxParam.Mjd_UTC = Mjd_UTC;
AuxParam.Mjd_TT = Mjd_TT;

Y0 = DEInteg (Accel,0,-(obs(46,1)-obs(1,1))*86400.0,1e-13,1e-6,6,Y);

 double aux[]= {5753.173e3, 2673.361e3, 3440.304e3, 4.324207e3, -1.924299e3, -5.728216e3};
 Y_true(6);
 for(i=0;i<6;i++){
     Y_true(i+1)=aux[i];
 }

cout«"\nError of Position Estimation\n";
cout«Y0(1)-Y_true(1)«" [m]\n";
cout«Y0(2)-Y_true(2)«" [m]\n";
cout«Y0(3)-Y_true(3)«" [m]\n";
cout«"\nError of Velocity Estimation\n";
cout«Y0(4)-Y_true(4)«" [m/s]\n";
cout«Y0(5)-Y_true(5)«" [m/s]\n";
cout«Y0(6)-Y_true(6)«" [m/s]\n";
return 0;
    }
```

5.145 tests.cpp

```
#include "../include/matrix.h"
#include "../include/R_x.h"
#include "../include/R_y.h"
#include "../include/R_z.h"
#include "../include/AccelPointMass.h"
#include "../include/Cheb3D.h"
#include "../include/EccAnom.h"
#include "../include/Frac.h"
#include "../include/MeanObliquity.h"
#include "../include/Mjday.h"
#include "../include/Mjday_TDB.h"
#include "../include/Position.h"
#include "../include/sign_.h"
#include "../include/timediff.h"
#include "../include/AzElPa.h"
#include "../include/IERS.h"
#include "../include/Legendre.h"
#include "../include/NutAngles.h"
#include "../include/TimeUpdate.h"
#include "../include/GLOBAL.h"
#include "../include/AccelHarmonic.h"
#include "../include/EqnEquinox.h"
#include "../include/JPL_Eph_DE430.h"
#include "../include/LTC.h"
#include "../include/NutMatrix.h"
#include "../include/PoleMatrix.h"
#include "../include/PrecMatrix.h"
#include "../include/gmst.h"
#include "../include/gast.h"
#include "../include/MeasUpdate.h"
#include "../include/G_AccelHarmonic.h"
#include "../include/GHAMatrix.h"
#include "../include/Accel.h"
#include "../include/VarEqn.h"
#include "../include/DEInteg.h"
#include <cstdio>
#include <cmath>
#include <tuple>

using namespace std;
int tests_run = 0;

#define FAIL() printf("\nfailure in %s() line %d\n", __func__, __LINE__)
#define _assert(test) do { if (!(test)) { FAIL(); return 1; } } while(0)
```

```
#define _verify(test) do { int r=test(); tests_run++; if(r) return r; } while(0)

int m_equals(Matrix A, Matrix B, double p) {
    if (A.n_row != B.n_row || A.n_column != B.n_column)
        return 0;
    else
        for(int i = 1; i <= A.n_row; i++)
            for(int j = 1; j <= A.n_column; j++)
                if(fabs(A(i,j)-B(i,j)) > p) {
                    printf("%2.20lf %2.20lf\n",A(i,j),B(i,j));
                    return 0;
                }

    return 1;
}
int m_sum_01() {
    int f = 3;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 0; A(1,2) =  2; A(1,3) = 8; A(1,4) = 0;
    A(2,1) = 1; A(2,2) = -1; A(2,3) = 0; A(2,4) = 0;
    A(3,1) = 0; A(3,2) =  1; A(3,3) = 0; A(3,4) = 5;

    Matrix B(f, c);
    B(1,1) = 2; B(1,2) =  0; B(1,3) = 0; B(1,4) = 0;
    B(2,1) = 7; B(2,2) = -2; B(2,3) = 1; B(2,4) = 0;
    B(3,1) = 0; B(3,2) = -3; B(3,3) = 0; B(3,4) = 2;

    Matrix C(f, c);
    C(1,1) = 2; C(1,2) =  2; C(1,3) = 8; C(1,4) = 0;
    C(2,1) = 8; C(2,2) = -3; C(2,3) = 1; C(2,4) = 0;
    C(3,1) = 0; C(3,2) = -2; C(3,3) = 0; C(3,4) = 7;

    Matrix R = A + B;

    _assert(m_equals(C, R, 1e-11));

    return 0;
}

int m_sub_01() {
    int f = 3;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 0; A(1,2) = 2; A(1,3) = 8; A(1,4) = 0;
    A(2,1) = 1; A(2,2) = -1; A(2,3) = 0; A(2,4) = 0;
    A(3,1) = 0; A(3,2) = 1; A(3,3) = 0; A(3,4) = 5;

    Matrix B(f, c);
    B(1,1) = 2; B(1,2) = 0; B(1,3) = 0; B(1,4) = 0;
    B(2,1) = 7; B(2,2) = -2; B(2,3) = 1; B(2,4) = 0;
    B(3,1) = 0; B(3,2) = -3; B(3,3) = 0; B(3,4) = 2;

    Matrix C(f, c);
    C(1,1) = -2; C(1,2) = 2; C(1,3) = 8; C(1,4) = 0;
    C(2,1) = -6; C(2,2) = 1; C(2,3) = -1; C(2,4) = 0;
    C(3,1) = 0; C(3,2) = 4; C(3,3) = 0; C(3,4) = 3;

    Matrix R = A - B;

    _assert(m_equals(C, R, 1e-11));

    return 0;
}

int m_mul_01() {
    int f = 4;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 5; A(1,2) = 8; A(1,3) = 2; A(1,4) = 4;
    A(2,1) = 2; A(2,2) = 2; A(2,3) = 2; A(2,4) = 2;
    A(3,1) = 2; A(3,2) = 2; A(3,3) = 1; A(3,4) = 3;
    A(4,1) = 2; A(4,2) = 2; A(4,3) = 2; A(4,4) = 1;

    Matrix B(f, c);

    B(1,1) = 4; B(1,2) = 2; B(1,3) = 9; B(1,4) = 1;
    B(2,1) = 2; B(2,2) = 9; B(2,3) = 2; B(2,4) = 7;
    B(3,1) = 2; B(3,2) = 2; B(3,3) = 4; B(3,4) = 9;
    B(4,1) = 3; B(4,2) = 4; B(4,3) = 2; B(4,4) = 5;

    Matrix C(f, c);

    C(1,1) = 52 ; C(1,2) = 102  ; C(1,3) = 77   ; C(1,4) = 99   ;
    C(2,1) = 22 ; C(2,2) = 34   ; C(2,3) = 34   ; C(2,4) = 44   ;
    C(3,1) = 23 ; C(3,2) = 36   ; C(3,3) = 32   ; C(3,4) = 40   ;
    C(4,1) = 19 ; C(4,2) = 30   ; C(4,3) = 32   ; C(4,4) = 39   ;

    Matrix R = A * B;
    _assert(m_equals(R, C, 1e-11));

    return 0;
}
int m_div_01() {
    int f = 4;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;

    Matrix B(f, c);

    B(1,1) = 4; B(1,2) = 2; B(1,3) = 9; B(1,4) = 1;
    B(2,1) = 2; B(2,2) = 9; B(2,3) = 2; B(2,4) = 7;
    B(3,1) = 2; B(3,2) = 2; B(3,3) = 4; B(3,4) = 9;
    B(4,1) = 3; B(4,2) = 4; B(4,3) = 2; B(4,4) = 5;

    Matrix C(f, c);

    C(1,1) = 337./963   ; C(1,2) = 350./963     ; C(1,3) = 677./963 ; C(1,4) = -271./321;
    C(2,1) = 23./963    ; C(2,2) = -179./963    ; C(2,3) = 592./963 ; C(2,4) = 112./321 ;
    C(3,1) = 58./963        ; C(3,2) = -577./963; C(3,3) = -475./963    ; C(3,4) = 743./321 ;
    C(4,1) = 430./963   ; C(4,2) = 338./963     ; C(4,3) = 98./963  ; C(4,4) = -181./321    ;

    Matrix R = A / B;
    _assert(m_equals(R, C, 1e-11));

    return 0;
}

int m_sum_d_01() {
    int f = 4;
    int c = 4;
    double num=2;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;


    Matrix B(f, c);
    B(1,1) = 1+num; B(1,2) = 2+num; B(1,3) = 5+num; B(1,4) = 5+num;
    B(2,1) = 2+num; B(2,2) = 1+num; B(2,3) = 3+num; B(2,4) = 6+num;
    B(3,1) = 5+num; B(3,2) = 3+num; B(3,3) = 2+num; B(3,4) = 3+num;
    B(4,1) = 1+num; B(4,2) = 2+num; B(4,3) = 4+num; B(4,4) = 1+num;


    Matrix R=A+num;

    _assert(m_equals(B, R, 1e-11));

    return 0;
}

int m_sub_d_01() {
    int f = 4;
    int c = 4;
    double num=2;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;


    Matrix B(f, c);
    B(1,1) = 1-num; B(1,2) = 2-num; B(1,3) = 5-num; B(1,4) = 5-num;
    B(2,1) = 2-num; B(2,2) = 1-num; B(2,3) = 3-num; B(2,4) = 6-num;
    B(3,1) = 5-num; B(3,2) = 3-num; B(3,3) = 2-num; B(3,4) = 3-num;
    B(4,1) = 1-num; B(4,2) = 2-num; B(4,3) = 4-num; B(4,4) = 1-num;


    Matrix R=A-num;

    _assert(m_equals(B, R, 1e-11));

    return 0;
}

int m_mul_d_01() {
    int f = 4;
    int c = 4;
    double num=2;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;


    Matrix B(f, c);
    B(1,1) = 1.*num; B(1,2) = 2.*num; B(1,3) = 5.*num; B(1,4) = 5.*num;
    B(2,1) = 2.*num; B(2,2) = 1.*num; B(2,3) = 3.*num; B(2,4) = 6.*num;
    B(3,1) = 5.*num; B(3,2) = 3.*num; B(3,3) = 2.*num; B(3,4) = 3.*num;
    B(4,1) = 1.*num; B(4,2) = 2.*num; B(4,3) = 4.*num; B(4,4) = 1.*num;

    Matrix R=A*num;

    _assert(m_equals(R, B, 1e-11));

    return 0;
}
int m_div_d_01() {
    int f = 4;
    int c = 4;
    double num=2;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;


    Matrix B(f, c);
    B(1,1) = 1./num; B(1,2) = 2./num; B(1,3) = 5./num; B(1,4) = 5./num;
    B(2,1) = 2./num; B(2,2) = 1./num; B(2,3) = 3./num; B(2,4) = 6./num;
    B(3,1) = 5./num; B(3,2) = 3./num; B(3,3) = 2./num; B(3,4) = 3./num;
    B(4,1) = 1./num; B(4,2) = 2./num; B(4,3) = 4./num; B(4,4) = 1./num;

    Matrix R=A/num;

    _assert(m_equals(R, B, 1e-11));

    return 0;
}
int m_asig_01() {
    int f = 4;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;

    Matrix B=A;

    _assert(m_equals(A, B, 1e-11));

    return 0;
}
int m_zeros_01() {
    int f = 3;
    int c = 4;

    Matrix A(f, c);
    A(1,1) = 0; A(1,2) = 0; A(1,3) = 0; A(1,4) = 0;
    A(2,1) = 0; A(2,2) = 0; A(2,3) = 0; A(2,4) = 0;
    A(3,1) = 0; A(3,2) = 0; A(3,3) = 0; A(3,4) = 0;

    Matrix B = zeros(3, 4);

    _assert(m_equals(A, B, 1e-11));

    return 0;
}

int m_eye_01() {
    int f = 3;


    Matrix A(f, f);
    A(1,1) = 1; A(1,2) = 0; A(1,3) = 0;
    A(2,1) = 0; A(2,2) = 1; A(2,3) = 0;
    A(3,1) = 0; A(3,2) = 0; A(3,3) = 1;


    Matrix B = eye(f);

    _assert(m_equals(A, B, 1e-11));


    return 0;
}

int m_transpose_01() {
    int f = 3;
    int c = 3;


    Matrix A(f, c);
    A(1,1) = 1; A(1,2) = 4; A(1,3) = 9;
    A(2,1) = 2; A(2,2) = 3; A(2,3) = 8;
    A(3,1) = 5; A(3,2) = 6; A(3,3) = 7;


    Matrix B(f,c);

    B(1,1) = 1; B(1,2) = 2; B(1,3) = 5;
    B(2,1) = 4; B(2,2) = 3; B(2,3) = 6;
    B(3,1) = 9; B(3,2) = 8; B(3,3) = 7;

    Matrix R=transpose(A);

    _assert(m_equals(R, B, 1e-11));

    return 0;
}
int m_inv_01() {
    int f = 4;


    Matrix A(f, f);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3; A(2,4) = 6;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2; A(3,4) = 3;
    A(4,1) = 1; A(4,2) = 2; A(4,3) = 4; A(4,4) = 1;

    Matrix B(f, f);

    B(1,1)=-47./42; B(1,2) = 5./6; B(1,3) = -1./14 ; B(1,4) =17./21;
    B(2,1)=41./21; B(2,2) = -5./3; B(2,3) = 4./7 ; B(2,4) =-31./21;
    B(3,1) = -17./21 ; B(3,2) = 2./3 ; B(3,3) = -2./7; B(3,4) =19./21;
    B(4,1) = 19./42 ; B(4,2) = -1./6; B(4,3) = 1./14 ; B(4,4) = -10./21;

    Matrix R=inv(A);
    _assert(m_equals(R, B, 1e-11));
    return 0;
}
int m_norm_01() {
    int f = 4;


    Matrix A(f);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 5; A(1,4) = 5;

    double B=7.4161984870956629487113974408007;

    double R=norm(A);
    _assert(fabs(R-B)<1e-11);
    return 0;
}

int m_dot_01() {
    int f = 3;
    Matrix A(f);
    A(1,1) = 1; A(1,2) = 2; A(1,3) = 3;

    Matrix B(f);
    B(1,1)= 1; B(1,2) = 2; B(1,3) = 3 ;


    double R=14;

    double C=dot(A,B);

    _assert(fabs(R-C)<1e-11);

    return 0;
}
int m_cross_01() {
    int f = 3;
    Matrix A(f);
    Matrix B(f);
    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    B(1,1)= 3; B(1,2) = 5; B(1,3) = 6 ;

    Matrix R(f);
    R(1,1)= 6; R(1,2) = -12; R(1,3) = 7 ;

    Matrix C=cross(A,B);

    _assert(m_equals(R, C, 1e-11));

    return 0;
}
int m_extract_vector_01() {
    int f = 5;
    Matrix A(f);

    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0; A(1,4) = 5; A(1,5) = 1;

    Matrix B=extract_vector(A,1,3);

    Matrix R(3);
    R(1,1)= 2; R(1,2) = 1; R(1,3) = 0 ;


    _assert(m_equals(R, B, 1e-11));

    return 0;
}

int m_union_vector_01() {
    int f = 3;
    Matrix A(f);
    Matrix B(f);
    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    B(1,1)= 3; B(1,2) = 1; B(1,3) = 6 ;

    Matrix R(6);
    R(1,1) = 2; R(1,2) = 1; R(1,3) = 0;
    R(1,4)= 3; R(1,5) = 1;R(1,6) = 6 ;
    Matrix C=union_vector(A,B);

    _assert(m_equals(R, C, 1e-11));

    return 0;
}

int m_extract_row_01() {
    int f = 3;
    Matrix A(f,f);

    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2;

    Matrix B=extract_row(A,f);

    Matrix R(f);
    R(1,1)= 5; R(1,2) = 3; R(1,3) = 2 ;

    _assert(m_equals(R, B, 1e-11));

    return 0;

}

int m_extract_column_01() {
    int f = 3;
    Matrix A(f,f);

    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2;

    Matrix B=extract_column(A,f);

    Matrix R(f);
    R(1,1)= 0; R(1,2) = 3; R(1,3) = 2 ;


    _assert(m_equals(R, B, 1e-11));

    return 0;
}

int m_assign_row_01() {
    int f = 3;
    Matrix A(f,f);

    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2;

    Matrix B(f);

    B(1,1)= 3; B(1,2) = 5; B(1,3) = 6 ;


    Matrix C=assign_row(A,B,f);

    Matrix R(f,f);
    R(1,1) = 2; R(1,2) = 1; R(1,3) = 0;
    R(2,1) = 2; R(2,2) = 1; R(2,3) = 3;
    R(3,1) = 3; R(3,2) = 5; R(3,3) = 6;

    _assert(m_equals(R, C, 1e-11));

    return 0;
}

int m_assign_column_01() {
    int f = 3;
    Matrix A(f,f);

    A(1,1) = 2; A(1,2) = 1; A(1,3) = 0;
    A(2,1) = 2; A(2,2) = 1; A(2,3) = 3;
    A(3,1) = 5; A(3,2) = 3; A(3,3) = 2;

    Matrix B(f);

    B(1,1)= 3; B(1,2) = 5; B(1,3) = 6 ;

    Matrix C=assign_column(A,B,f);

    Matrix R(f,f);
    R(1,1) = 2; R(1,2) = 1; R(1,3) = 3;
    R(2,1) = 2; R(2,2) = 1; R(2,3) = 5;
    R(3,1) = 5; R(3,2) = 3; R(3,3) = 6;


    _assert(m_equals(R, C, 1e-11));

    return 0;
}

int m_R_x_01() {
    Matrix A=R_x(10);


    Matrix R(3,3);
    R(1,1) = 1; R(1,2) = 0; R(1,3) = 0;
    R(2,1) = 0; R(2,2) = -0.839071529076452; R(2,3) = -0.54402111088937;
    R(3,1) = 0; R(3,2) = 0.54402111088937; R(3,3) = -0.839071529076452;

    _assert(m_equals(R, A, 1e-11));

    return 0;
}

int m_R_y_01() {

    Matrix A=R_y(10);

    Matrix R(3,3);
    R(1,1) = -0.839071529076452 ; R(1,2) = 0; R(1,3) = 0.54402111088937;
    R(2,1) = 0                  ; R(2,2) = 1; R(2,3) = 0;
    R(3,1) = -0.54402111088937  ; R(3,2) = 0; R(3,3) = -0.839071529076452;


    _assert(m_equals(R, A, 1e-11));

    return 0;
}

int m_R_z_01() {

    Matrix A=R_z(10);


    Matrix R(3,3);
    R(1,1) = -0.839071529076452; R(1,2) = -0.54402111088937; R(1,3) = 0;
    R(2,1) = 0.54402111088937; R(2,2) = -0.839071529076452; R(2,3) = 0;
    R(3,1) = 0; R(3,2) = 0; R(3,3) = 1;


    _assert(m_equals(R, A, 1e-11));

    return 0;
}

int m_AccelPointMass_01() {

    Matrix A(3);

    A(1,1) = 1; A(1,2) = 1; A(1,3) = 1;

    Matrix B(3);

    B(1,1)= 2; B(1,2) = 3; B(1,3) = 4 ;

    Matrix C=AccelPointMass(A,B,10);

    Matrix R(3);
    R(1,1) = 0.0628351366133708; R(1,2) = 0.189703148460208; R(1,3) = 0.316571160307045;

    _assert(m_equals(R, A, 1e-11));

    return 0;
}
int m_Cheb3D_01() {
    double f = 3;

    Matrix A(f);

    A(1,1) = 1; A(1,2) = 2; A(1,3) = 3;

    Matrix B(f);

    B(1,1)= 1; B(1,2) = 2; B(1,3) = 3;

    Matrix C(f);

    C(1,1)= 5; C(1,2) = 2; C(1,3) = 3;

    Matrix D=Cheb3D(1,3,0.5,1,A,B,C);

    Matrix R(f);
    R(1,1) = 6; R(1,2) = 6; R(1,3) = 10;

    _assert(m_equals(R, D, 1e-11));

    return 0;
}
int m_EccAnom_01() {

    double R = 2.38006127313934;
    double D=EccAnom(1,2);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}
int m_Frac_01() {

    double R = 0.3801;
    double D=Frac(2.3801);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}

int m_MeanObliquity_01() {

    double R = 0.409412815476201;
    double D=MeanObliquity(41);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}


int m_Mjday_01() {

    double R = 60800;
    double D= Mjday(2025,5,5);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}
int m_Mjday_TDB_01() {

    double R = 2025.0000000092;
    double D= Mjday_TDB(2025);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}
int m_Position_01() {

    Matrix R(3);
    R(1) = 2.627855739427486e+06; R(2) = -5.741969545549633e+06; R(3) = 8.941173180321892e+05;

    Matrix D= Position(2,3,4);
    _assert(m_equals(R, D, 1e-8));

    return 0;
}
int m_sign__01() {

    double R = -4;
    double D= sign_(4,-3);

    _assert(fabs(R-D)< 1e-11);

    return 0;
}
int m_timediff_01() {

    double R0 = -6;
    double R1 = 9;
    double R2 = 13;
    double R3 = 42.184;
    double R4 = -9;
    auto D= timediff(4,10);
    _assert(fabs(get<0>(D)-R0)< 1e-11);
    _assert(fabs(get<1>(D)-R1)< 1e-11);
    _assert(fabs(get<2>(D)-R2)< 1e-11);
    _assert(fabs(get<3>(D)-R3)< 1e-11);
    _assert(fabs(get<4>(D)-R4)< 1e-11);

    return 0;
}
int m_AzElPa_01() {
    Matrix A(3);

    A(1) = 1; A(2) = 2; A(3) = 3;

    double R0=0.463647609000806;

    double R1=0.930274014115472;

    Matrix R2(3);

    R2(1) = 0.4; R2(2) = -0.2; R2(3) = 0;
    Matrix R3(3);

    R3(1) = -0.095831484749991; R3(2) = -0.191662969499982; R3(3) = 0.159719141249985;

    auto [Az, El, dAds, dEds]= AzElPa(A);
    _assert(fabs(Az-R0)< 1e-11);
    _assert(fabs(El-R1)< 1e-11);
    _assert(m_equals(dAds,R2,1e-11));
    _assert(m_equals(dEds,R3,1e-11));


    return 0;
}

int m_IERS_01() {


    eop19620101(21413);

    double R0 = -5.59518621231704e-07;
    double R1 = 2.33458634442529e-06;
    double R2 =  0.3260677;
    double R3 = 0.0027213;
    double R4 = -1.16864337831454e-07;
    double R5 = -2.48709418409192e-08;
    double R6 = -8.19335121075116e-10;
    double R7 =  -1.53201123230613e-09;
    double R8 = 29;


    auto [x_pole,y_pole,UT1_UTC,LOD,dpsi,deps,dx_pole,dy_pole,TAI_UTC]= IERS(eopdata,49746,'l');
    _assert(fabs(x_pole-R0)< 1e-11);
    _assert(fabs(y_pole-R1)< 1e-11);
    _assert(fabs(UT1_UTC-R2)< 1e-11);
    _assert(fabs(LOD-R3)< 1e-11);
    _assert(fabs(dpsi-R4)< 1e-11);
    _assert(fabs(deps-R5)< 1e-11);
    _assert(fabs(dx_pole-R6)< 1e-11);
    _assert(fabs(dy_pole-R7)< 1e-11);
    _assert(fabs(TAI_UTC-R8)< 1e-11);


    return 0;
}

int m_Legendre_01() {


    Matrix R0 (4,4);
    R0(1,1) = 1; R0(1,2) = 0; R0(1,3) = 0; R0(1,4) = 0;
    R0(2,1) =  1.4574704987823; R0(2,2) =  0.935831045210238; R0(2,3) = 0; R0(2,4) = 0;
    R0(3,1) =  1.25691645573063 ; R0(3,2) = 1.76084689542256  ; R0(3,3) = 0.565313394670859 ; R0(3,4) = 0;
    R0(4,1) = 0.601515831515714; R0(4,2) = 2.22381140389174 ; R0(4,3) = 1.25857019087392 ; R0(4,4) = 0.329913047636197;
    Matrix R1 (4,4);
    R1(1,1) = 0; R1(1,2) = 0; R1(1,3) = 0; R1(1,4) = 0;
    R1(2,1) = 0.935831045210238 ; R1(2,2) = -1.4574704987823 ; R1(2,3) = 0; R1(2,4) = 0;
    R1(3,1) =  3.0498762872218; R1(3,2) = -1.61172976752398 ; R1(3,3) = -1.76084689542256  ; R1(3,4) = 0;
    R1(4,1) = 5.44720322371707  ; R1(4,2) =  0.516567339757783 ; R1(4,3) =  -3.11209524837966 ; R1(4,4) = -1.54142738655916;

    auto [pnm, dpnm]= Legendre(3,3,1);
    _assert(m_equals(pnm,R0, 1e-11));
    _assert(m_equals(dpnm,R1,1e-11));


    return 0;
}
int m_NutAngles_01() {


    double R0 = 2.72256565175042e-05;
    double R1 =  3.87947551912632e-05;

    auto [dpsi, deps]= NutAngles(3);
    _assert(fabs(dpsi-R0)< 1e-11);
    _assert(fabs(deps-R1)< 1e-11);


    return 0;
}

int m_TimeUpdate_01() {

    Matrix A(3, 3);
    A(1,1) = 1; A(1,2) = 4; A(1,3) = 9;
    A(2,1) = 2; A(2,2) = 3; A(2,3) = 8;
    A(3,1) = 5; A(3,2) = 6; A(3,3) = 7;


    Matrix B(3,3);

    B(1,1) = 1; B(1,2) = 2; B(1,3) = 5;
    B(2,1) = 4; B(2,2) = 3; B(2,3) = 6;
    B(3,1) = 9; B(3,2) = 8; B(3,3) = 7;

    Matrix C(3, 3);

    C(1,1) = 52 ; C(1,2) = 102  ; C(1,3) = 77   ;
    C(2,1) = 22 ; C(2,2) = 34   ; C(2,3) = 34   ;
    C(3,1) = 23 ; C(3,2) = 36   ; C(3,3) = 32   ;


    Matrix D(3, 3);

    D(1,1) = 462    ; D(1,2) = 702  ; D(1,3) = 1087 ;
    D(2,1) = 694    ; D(2,2) = 989  ; D(2,3) = 1596 ;
    D(3,1) = 1257   ; D(3,2) = 1746 ; D(3,3) = 2746 ;

    Matrix R = TimeUpdate(A,B,C);

    _assert(m_equals(R,D,1e-11));

    return 0;
}
int m_AccelHarmonic_01() {

    Matrix R0(3);
    R0(1)=2.42488766379455e+34;
    R0(2)=2.65762182552943e+34;
    R0(3)=2.65762182552943e+34;

    Matrix A(3);
    A(1)=1.0;
    A(2)=2.0;
    A(3)=3.0;
    A=transpose(A);
    Matrix B(3,3);
    B(1,1)= 1.0; B(1,2) = 1.0; B(1,3) = 1.0 ;
    B(2,1)= 1.0; B(2,2) = 2.0; B(2,3) = 2.0 ;
    B(3,1)= 1.0; B(3,2) = 3.0; B(3,3) = 3.0 ;

    Matrix R = AccelHarmonic(A,B,5,5);
    _assert(m_equals(R,R0,R0(1)*1e-11));

    return 0;
}

int m_EqnEquinox_01() {

    double R0=2.6045897022442e-05;
    double R = EqnEquinox(5);

    _assert(fabs(R-R0)< 1e-11);

    return 0;
}

int m_JPL_Eph_DE430_01() {

    Matrix R0(3);
    R0(1)=147208460159.245;
    R0(2)=  54592844683.9181;
    R0(3)= 15319523517.8098;
    Matrix R1(3);
    R1(1)= 72752904522.483;
    R1(2)=2340227175.73022;
    R1(3)=1670913926.26657;
    Matrix R2(3);
    R2(1)=-108493583087.765;
    R2(2)=-97599455066.7732;
    R2(3)=-42280555048.3341;
    Matrix R3(3);
    R3(1)=-131548434829.954;
    R3(2)= 156673495960.063;
    R3(3)= 75876841465.0958;
    Matrix R4(3);
    R4(1)=125152801267.633;
    R4(2)= 801496792415.556;
    R4(3)=343590087271.679;
    Matrix R5(3);
    R5(1)= 1532648188929.54;
    R5(2)=-30170201818.4012;
    R5(3)=-71835402852.1091;
    Matrix R6(3);
    R6(1)= 1707620424797.68;
    R6(2)= 2344787198692.63;
    R6(3)=1003869836966.23;
    Matrix R7(3);
    R7(1)= 4578089056071.27;
    R7(2)= 104288160816.022;
    R7(3)= -66258775563.9107;
    Matrix R8(3);
    R8(1)=2885822907234;
    R8(2)= -3876433792795.09;
    R8(3)=-2034695099132.45;
    Matrix R9(3);
    R9(1)=-295931131.772483;
    R9(2)= 224622149.622798;
    R9(3)= 120216992.495936;
    Matrix R10(3);
    R10(1)= 107770931597.498;
    R10(2)=96865992179.6201;
    R10(3)=41989334136.8052;
    auto [r_Mercury,r_Venus,r_Earth,r_Mars,r_Jupiter,r_Saturn,r_Uranus,r_Neptune,r_Pluto,r_Moon,r_Sun]
= JPL_Eph_DE430(60800);
```

```
    _assert(m_equals(r_Mercury,R0, abs(R0(1)*1e-11)));
    _assert(m_equals(r_Venus,R1,abs(R1(1)*1e-11)));
    _assert(m_equals(r_Earth,R2, abs(R2(1)*1e-11)));
    _assert(m_equals(r_Mars,R3,abs(R3(1)*1e-11)));
    _assert(m_equals(r_Jupiter,R4, abs(R4(1)*1e-11)));
    _assert(m_equals(r_Saturn,R5,abs(R5(1)*1e-11)));
    _assert(m_equals(r_Uranus,R6, abs(R6(1)*1e-11)));
    _assert(m_equals(r_Neptune,R7,abs(R7(1)*1e-11)));
    _assert(m_equals(r_Pluto,R8, abs(R8(1)*1e-11)));
    _assert(m_equals(r_Moon,R9,abs(R9(1)*1e-11)));
    _assert(m_equals(r_Sun,R10, abs(R10(1)*1e-11)));

    return 0;
}


int m_LTC_01() {

    Matrix A=LTC(10,10);


    Matrix R(3,3);
    R(1,1) = 0.54402111088937; R(1,2) = -0.839071529076452; R(1,3) = 0;
    R(2,1) = -0.456472625363814; R(2,2) = -0.295958969093304; R(2,3) = -0.839071529076452;
    R(3,1) = 0.704041030906696; R(3,2) = 0.456472625363814; R(3,3) = -0.54402111088937;


    _assert(m_equals(R, A, 1e-11));

    return 0;
}


int m_NutMatrix_01() {

    Matrix A=NutMatrix(10);


    Matrix R(3,3);
    R(1,1) = 0.999999999492159; R(1,2) = -2.92358797494727e-05 ; R(1,3) = -1.26864277798066e-05;
    R(2,1) = 2.9235393806329e-05 ; R(2,2) = 0.999999998839099; R(2,3) = -3.83026695232602e-05;
    R(3,1) = 1.26875475773192e-05; R(3,2) =  3.83022986110704e-05 ; R(3,3) =  0.99999999918598;


    _assert(m_equals(R, A, 1e-11));

    return 0;
}
int m_PoleMatrix_01() {

    Matrix A=PoleMatrix(10,10);


    Matrix R(3,3);
    R(1,1) = -0.839071529076452; R(1,2) = 0.295958969093304; R(1,3) = 0.456472625363814;
    R(2,1) = 0; R(2,2) = -0.839071529076452; R(2,3) = 0.54402111088937;
    R(3,1) = 0.54402111088937; R(3,2) = 0.456472625363814; R(3,3) = 0.704041030906696;


    _assert(m_equals(R, A, 1e-11));

    return 0;
   }

int m_PrecMatrix_01() {

    Matrix A=PrecMatrix(100,1);


    Matrix R(3,3);
    R(1,1) = 0.999999997819034; R(1,2) =  6.05590736738844e-05; R(1,3) = 2.63539319986234e-05;
    R(2,1) =  -6.05590736738844e-05; R(2,2) = 0.999999998166299 ; R(2,3) = -7.97984483806257e-10;
    R(3,1) = -2.63539319986234e-05; R(3,2) =   -7.97985227435292e-10; R(3,3) =  0.999999999652735;


    _assert(m_equals(R, A, 1e-11));

    return 0;
   }
   int m_gmst_01() {

    double A=gmst(10);


    double R=1.14523606099042;


    _assert(fabs(R-A)< 1e-11);

    return 0;
   }

   int m_gast_01() {

    double A=gast(10);


    double R=1.14526529687017;


    _assert(fabs(R-A)< 1e-11);

    return 0;
   }

int m_MeasUpdate_01() {

    Matrix A(3);
    A(1)=1;
    A(2)=2;
    A(3)=3;
    Matrix B=transpose(A);

    Matrix C(3,3);
    C(1,1) = 1; C(1,2) = 2; C(1,3) = 3;
    C(2,1) = 6; C(2,2) = 2; C(2,3) = 3;
    C(3,1) = 8; C(3,2) = 2; C(3,3) = 3;

    auto [K, x, P]=MeasUpdate(B,2,3,4,A,C,3);


    Matrix R0(3);
    R0(1)=0.106870229007634;
    R0(2)=  0.145038167938931;
    R0(3)= 0.16030534351145;
    R0=transpose(R0);
    Matrix R1(3);
    R1(1)= 0.893129770992366;
    R1(2)= 1.85496183206107;
    R1(3)=2.83969465648855;
    R1=transpose(R1);

    Matrix R2(3,3);
    R2(1,1) = -2.95419847328244  ; R2(1,2) = 0.717557251908397; R2(1,3) = 1.0763358778626;
    R2(2,1) = 0.633587786259541; R2(2,2) = 0.259541984732824; R2(2,3) = 0.389312977099237;
    R2(3,1) = 2.06870229007634; R2(3,2) = 0.0763358778625954 ; R2(3,3) = 0.114503816793893;
    _assert(m_equals(R0, K, 1e-11));
    _assert(m_equals(R1, x, 1e-11));
    _assert(m_equals(R2, P, 1e-11));

    return 0;
   }
   int m_G_AccelHarmonic_01() {

    Matrix R2(3,3);
    R2(1,1) = -2.0122905124052e+34   ; R2(1,2) =-3.29511632095482e+34 ; R2(1,3) =
-3.29511632095482e+34;
    R2(2,1) = -3.11072371483497e+34  ; R2(2,2) = -3.38401367341354e+34 ; R2(2,3) =
-3.38401367341354e+34;
    R2(3,1) =  -3.11072371483497e+34 ; R2(3,2) =  -3.38401367341354e+34 ; R2(3,3) =
-3.38401367341354e+34;

    Matrix A(3);
    A(1)=1.0;
    A(2)=2.0;
    A(3)=3.0;
    A=transpose(A);
    Matrix B(3,3);
    B(1,1)= 1.0; B(1,2) = 1.0; B(1,3) = 1.0 ;
    B(2,1)= 1.0; B(2,2) = 2.0; B(2,3) = 2.0 ;
    B(3,1)= 1.0; B(3,2) = 3.0; B(3,3) = 3.0 ;

    Matrix R = G_AccelHarmonic(A,B,5,5);
    _assert(m_equals(R,R2,fabs(R2(1)*1e-11)));

    return 0;
}
int m_GHAMatrix_01() {

    Matrix R(3,3);
    R(1,1) = 0.412804512414729; R(1,2) = 0.910819649837463 ; R(1,3) = 0;
    R(2,1) = -0.910819649837463 ; R(2,2) = 0.412804512414729; R(2,3) = 0;
    R(3,1) = 0; R(3,2) = 0; R(3,3) = 1;


    Matrix A = GHAMatrix(10);

    _assert(m_equals(R,A,1e-11));

    return 0;
}

int m_Accel_01() {

    Matrix R(6);
    R(1)=1.0;
    R(2)=2.0;
    R(3)=3.0;
    R(4)=-9.52489066332755e+131;
    R(5)=-1.68703107956274e+132;
    R(6)=-4.07471909292663e+132;

    Matrix A(6);
    A(1)=1.0;
    A(2)=2.0;
    A(3)=3.0;
    A(4)=1.0;
    A(5)=2.0;
    A(6)=3.0;
    A=transpose(A);
    Matrix B = Accel(10,A);


    _assert(m_equals(R,B,abs(R(6)*1e-11)));

    return 0;
}
int m_VarEqn_01() {
Matrix A(42);
    A(1)=7101800.90695315;
    A(2)=1293997.58115302;
    A(3)=10114.014948955;
    A(4)= 573.068082065557;
    A(5)= -3085.15736953138;
    A(6)=      -6736.03068347156;
    A(7)=        1.0000293469741;
    A(8)=    8.22733917593032e-06;
    A(9)=   2.17104932968693e-07;
    A(10)=    1.08925458231315e-05;
    A(11)=   3.04673932160225e-06;
    A(12)=    6.63504292706821e-08;
    A(13)=    8.22733944423959e-06;
    A(14)=      0.999986101965304;
    A(15)=   3.99927483270551e-08;
    A(16)=    3.04673960163327e-06;
    A(17)=   -5.1596062466179e-06;
    A(18)=    1.22075292404534e-08;
    A(19)=   2.17105640392839e-07;
    A(20)=     3.9992870847826e-08;
    A(21)=       0.999984551298692;
    A(22)=    6.63510875632706e-08;
    A(23)=    1.22076480274715e-08;
    A(24)=  -5.73276287738792e-06;
    A(25)=      5.38976081674752;
    A(26)=    1.47507305174403e-05;
    A(27)=   3.21241787851554e-07;
    A(28)=       1.00002936035846;
    A(29)=   8.19365458482084e-06;
    A(30)=    1.40504658112974e-07;
    A(31)=    1.47507306419397e-05;
    A(32)=        5.38968310056198;
    A(33)=    5.90697768748029e-08;
    A(34)=   8.19365482653896e-06;
    A(35)=          0.9999860891763;
    A(36)=    2.58022974647481e-08;
    A(37)=     3.21242427100724e-07;
    A(38)=  5.90698876854246e-08;
    A(39)=         5.38968032557769;
    A(40)=      1.4050537070756e-07;
    A(41)=    2.58024285760964e-08;
    A(42)=        0.999984550703337;
    Matrix R(42);
    R(1) = 573.068082065557;
    R(2) = -3085.15736953138;
    R(3) = -6736.03068347156;
    R(4) = -7.53489822593659;
    R(5) = -1.37294429126638;
    R(6) = -0.0107597986473575;
    R(7) = 1.08925458231315e-05;
    R(8) = 3.04673932160225e-06;
    R(9) = 6.63504292706821e-08;
    R(10) = 2.02239897508587e-06;
    R(11) = 5.61811901849645e-07;
    R(12) = 4.39846387071934e-09;
    R(13) = 3.04673960163327e-06;
    R(14) = -5.1596062466179e-06;
    R(15) = 1.22075292404534e-08;
    R(16) = 5.61812134084449e-07;
    R(17) = -9.58613689243416e-07;
    R(18) = 8.05616500343474e-10;
    R(19) = 6.63510875632706e-08;
    R(20) = 1.22076480274715e-08;
    R(21) = -5.73276287738792e-06;
    R(22) = 4.39895597958216e-09;
    R(23) = 8.0570607835305e-10;
    R(24) = -1.06368693580442e-06;
    R(25) = 1.00002936035846;
    R(26) = 8.19365458482084e-06;
    R(27) = 1.40504658112974e-07;
    R(28) = 1.08999102436198e-05;
    R(29) = 3.02797128053784e-06;
    R(30) = 2.37068516291712e-08;
    R(31) = 8.19365482653896e-06;
    R(32) = 0.9999860891763;
    R(33) = 2.58022974647481e-08;
    R(34) = 3.02797160153579e-06;
    R(35) = -5.16671243316801e-06;
    R(36) = 4.34211426867344e-09;
    R(37) = 1.4050537070756e-07;
    R(38) = 2.58024285760964e-08;
    R(39) = 0.999984550703337;
    R(40) = 2.37075280907946e-08;
    R(41) = 4.34223837651307e-09;
    R(42) = -5.73302112206999e-06;
    R=transpose(R);
    Matrix B = VarEqn( 5.38970808087706,A);

    _assert(m_equals(R,B,abs(R(1)*1e-11)));

    return 0;
}
int m_DEInteg_01() {

    Matrix R(6);
    R(1)=5542555.89427452;
    R(2)=3213514.83814162;
    R(3)= 3990892.92789074;
    R(4)=5394.06894044389;
    R(5)=-2365.2129057402;
    R(6)=-7061.8448137347;

    Matrix A(6);
    A(1)=   6221397.62857869;
    A(2)=   2867713.77965738;
    A(3)=   3006155.98509949;
    A(4)=   4645.04725161806;
    A(5)=  -2752.21591588204;
    A(6)=  -7507.99940987031;

    A=transpose(A);
    Matrix B = DEInteg(Accel,0,-134.999991953373,1e-13,1e-6,6,A);


    _assert(m_equals(R,B,abs(R(5)*1e-11)));

    return 0;
}

int all_tests()
{
    _verify(m_sum_01);
    _verify(m_sub_01);
    _verify(m_mul_01);
    _verify(m_div_01);
    _verify(m_sum_d_01);
    _verify(m_sub_d_01);
    _verify(m_mul_d_01);
    _verify(m_div_d_01);
    _verify(m_asig_01);
    _verify(m_zeros_01);
    _verify(m_eye_01);
    _verify(m_transpose_01);
    _verify(m_inv_01);
    _verify(m_norm_01);
    _verify(m_dot_01);
    _verify(m_cross_01);
    _verify(m_extract_vector_01);
    _verify(m_union_vector_01);
    _verify(m_extract_row_01);
    _verify(m_extract_column_01);
    _verify(m_assign_row_01);
    _verify(m_assign_column_01);
    _verify(m_R_x_01);
    _verify(m_R_y_01);
    _verify(m_R_z_01);
    _verify(m_Cheb3D_01);
    _verify(m_EccAnom_01);
    _verify(m_Frac_01);
    _verify(m_MeanObliquity_01);
    _verify(m_Mjday_01);
    _verify(m_Mjday_TDB_01);
    _verify(m_Position_01);
    _verify(m_sign__01);
    _verify(m_timediff_01);
    _verify(m_AzElPa_01);
    _verify(m_IERS_01);
    _verify(m_Legendre_01);
    _verify(m_NutAngles_01);
    _verify(m_TimeUpdate_01);
    _verify(m_AccelHarmonic_01);
    _verify(m_EqnEquinox_01);
    _verify(m_JPL_Eph_DE430_01);
    _verify(m_LTC_01);
    _verify(m_NutMatrix_01);
    _verify(m_PoleMatrix_01);
    _verify(m_PrecMatrix_01);
    _verify(m_gmst_01);
    _verify(m_gast_01);
    _verify(m_MeasUpdate_01);
    _verify(m_G_AccelHarmonic_01);
    _verify(m_GHAMatrix_01);
    _verify(m_Accel_01);
    _verify(m_VarEqn_01);
    _verify(m_DEInteg_01);


    return 0;
}


int main()
{
    AuxParamLoad();
    eop19620101();
    GGM03S();
    DE430Coeff();
    GEOS3();
    int result = all_tests();

    if (result == 0)
        printf("PASSED\n");

    printf("Tests run: %d\n", tests_run);

    return (result != 0);
}
```

Index