

# GTEC

version 0.1

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	igrf Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	5
3.1.2	Constructor & Destructor Documentation . . . . .	5
3.1.2.1	igrf(std::string fname) . . . . .	5
3.1.3	Member Function Documentation . . . . .	6
3.1.3.1	getMODIP(triple &pos, int &t) . . . . .	6
3.2	internalTime Class Reference . . . . .	6
3.2.1	Detailed Description . . . . .	7
3.2.2	Constructor & Destructor Documentation . . . . .	7
3.2.2.1	internalTime(int Y, int M, int D, int h, int m, int s) . . . . .	7
3.2.2.2	internalTime() . . . . .	7
3.2.3	Member Function Documentation . . . . .	8
3.2.3.1	parse(std::string strtime) . . . . .	8
3.2.3.2	toUNIXTime() . . . . .	8
3.2.4	Member Data Documentation . . . . .	8
3.2.4.1	year . . . . .	8
3.3	navigation Class Reference . . . . .	8

3.3.1	Detailed Description	9
3.3.2	Constructor & Destructor Documentation	9
3.3.2.1	navigation(std::vector< std::string > fnames)	9
3.3.2.2	navigation()	10
3.3.3	Member Function Documentation	10
3.3.3.1	applyRotations(float &Lk, float &ik, float &uk, float &rk, triple &pos)	10
3.3.3.2	computeIPP(const triple &marker, const triple &sat, const double &rh, triple &IPP, double &coschi)	10
3.3.3.3	eccAnomaly(float M, float e)	11
3.3.3.4	ecefToEllipsoidal(const triple &ecef, triple &ellipsoid)	11
3.3.3.5	getPositionGE(ephemerisGE &initial, int t, triple &pos)	11
3.3.3.6	getPositionR(ephemerisR &initialConditions, int h, triple &pos)	12
3.3.3.7	read()	12
3.3.3.8	satElevAzim(triple &markerECEF, triple &sat, triple &markerEllip, double &elevation, double &azimuth)	12
3.4	ObsData Class Reference	14
3.4.1	Detailed Description	16
3.4.2	Constructor & Destructor Documentation	16
3.4.2.1	ObsData(std::vector< std::string > fvec, std::string sysString)	16
3.4.3	Member Function Documentation	17
3.4.3.1	buildB()	17
3.4.3.2	cleanUp()	17
3.4.3.3	dumpRawMatrix(const double *mat, int &dim1, int &dim2)	17
3.4.3.4	lagrangeInterpolation(float *target, float *s, float *e, int deg)	17
3.4.3.5	pre_process(int minArcLen, int intrpollIntrvl, int deg)	18
3.4.3.6	read()	18
3.4.3.7	setArcStartEnd()	19
3.4.3.8	setSysFlags(std::string sysString)	19
3.4.4	Member Data Documentation	19
3.4.4.1	arcs	19
3.4.4.2	arcs2	19

3.4.4.3	<a href="#">arcs3</a>	19
3.4.4.4	<a href="#">B</a>	19
3.4.4.5	<a href="#">BDU_ucTEC</a>	20
3.4.4.6	<a href="#">GAL_ucTEC</a>	20
3.4.4.7	<a href="#">GLO_ucTEC</a>	20
3.4.4.8	<a href="#">GPS_ucTEC</a>	20
3.4.4.9	<a href="#">hasTOFO</a>	20
3.4.4.10	<a href="#">numArcs</a>	20
3.4.4.11	<a href="#">S</a>	20
3.4.4.12	<a href="#">S_arcnum</a>	21
3.4.4.13	<a href="#">S_prn</a>	21
3.5	<a href="#">ptr_pair Class Reference</a>	21
3.5.1	<a href="#">Detailed Description</a>	21
3.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	22
3.5.2.1	<a href="#">ptr_pair()</a>	22
3.5.2.2	<a href="#">ptr_pair(float *s, float *e)</a>	22
3.6	<a href="#">triple Class Reference</a>	23
3.6.1	<a href="#">Detailed Description</a>	23
3.6.2	<a href="#">Member Function Documentation</a>	23
3.6.2.1	<a href="#">dump(std::ostream &amp;s)</a>	23
<b>4</b>	<b><a href="#">File Documentation</a></b>	<b>25</b>
4.1	<a href="#">igrf.hpp File Reference</a>	25
4.1.1	<a href="#">Detailed Description</a>	25
4.2	<a href="#">internalTime.hpp File Reference</a>	26
4.2.1	<a href="#">Detailed Description</a>	26
4.3	<a href="#">navigation.hpp File Reference</a>	27
4.3.1	<a href="#">Detailed Description</a>	27
4.4	<a href="#">ObsData.hpp File Reference</a>	27
4.4.1	<a href="#">Detailed Description</a>	28
4.5	<a href="#">ptr_pair.hpp File Reference</a>	28
4.5.1	<a href="#">Detailed Description</a>	29
4.6	<a href="#">triple.hpp File Reference</a>	29
4.6.1	<a href="#">Detailed Description</a>	30
	<b><a href="#">Index</a></b>	<b>31</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">igrf</a> . . . . .	5
<a href="#">internalTime</a> . . . . .	6
<a href="#">navigation</a> . . . . .	8
<a href="#">ObsData</a> . . . . .	14
<a href="#">ptr_pair</a> . . . . .	21
<a href="#">triple</a> . . . . .	23





## Chapter 2

# File Index

### 2.1 File List

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

<b>constants.hpp</b>	??
<b>ephemerisGE.hpp</b>	??
<b>ephemerisR.hpp</b>	??
<a href="#">igrf.hpp</a>	
This class implements IGRF model	<a href="#">25</a>
<b>inout.hpp</b>	??
<b>int_pair.hpp</b>	??
<a href="#">internalTime.hpp</a>	
Class defining internal time format	<a href="#">26</a>
<a href="#">navigation.hpp</a>	
This is class navigation data	<a href="#">27</a>
<a href="#">ObsData.hpp</a>	
Class defining observation data	<a href="#">27</a>
<a href="#">ptr_pair.hpp</a>	
Class defining pointer pairs	<a href="#">28</a>
<a href="#">triple.hpp</a>	
This class defines a 3-D Coordinate	<a href="#">29</a>



## Chapter 3

# Class Documentation

### 3.1 igrf Class Reference

```
#include <igrf.hpp>
```

#### Public Member Functions

- `igrf` (`std::string fname`)  
*Constructor with Input file.*
- `double getMODIP` (`triple &pos`, `int &t`)  
*Function to compute MODIP.*

#### 3.1.1 Detailed Description

##### Author

Muhammad Owais

##### Date

14/01/17

#### 3.1.2 Constructor & Destructor Documentation

##### 3.1.2.1 `igrf::igrf ( std::string fname )`

Constructor with Input file.

Constructs `igrf` object by reading input IGRF coefficients file.

##### Parameters

<code>fname</code>	IGRF coefficients file name.
--------------------	------------------------------

Here is the call graph for this function:



### 3.1.3 Member Function Documentation

#### 3.1.3.1 `double igrf::getMODIP ( triple & pos, int & t )`

Function to compute MODIP.

This function compute MODIP (Modified Dip) given ellipsoidal coordinates of the point and time (in unit of years).

##### Parameters

<i>pos</i>	ellipsoidal coordinates of the point as <a href="#">triple</a> object.
<i>t</i>	time (in unit of years).

##### Returns

Returns computed MODIP.

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

- [igrf.hpp](#)
- [igrf.cpp](#)

## 3.2 internalTime Class Reference

```
#include <internalTime.hpp>
```

### Public Member Functions

- [internalTime](#) (int Y, int M, int D, int h, int m, int s)  
*Constructor with explicit values.*
- void [parse](#) (std::string strtime)  
*Member function parse.*
- void [toUNIXTime](#) ()  
*Member Function, providing UNIX time.*
- [internalTime](#) ()

## Public Attributes

- int [year](#)
- int [month](#)  
*Stores Month as Integer.*
- int [day](#)  
*Stores day as Integer.*
- int [hour](#)  
*Stores hour as Integer.*
- int [minute](#)  
*Stores minute as Integer.*
- int [second](#)  
*Stores second as Integer.*
- int [UNIX](#)  
*Stores Converted UNIX Time as Integer.*

### 3.2.1 Detailed Description

#### Author

Muhammad Owais

#### Date

04/12/16

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 internalTime::internalTime ( int *Y*, int *M*, int *D*, int *h*, int *m*, int *s* )

Constructor with explicit values.

Constructs [internalTime](#) object explicitly taking date/time values as parameters. Requires 6 integers (YY↔YY,MM,DD,hh,mm,ss).

#### Parameters

<i>Y</i>	year(YYYY), given as integer
<i>M</i>	Month(MM), given as integer
<i>D</i>	Day(DD), given as integer
<i>h</i>	Hour(hh), given as integer
<i>m</i>	Minute(mm), given as integer
<i>s</i>	Second(ss), given as integer

#### 3.2.2.2 internalTime::internalTime ( )

Default Constructor.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 void internalTime::parse ( std::string *strtime* )

Member function parse.

Member function parse sets internal values by parsing a given string representing date/time values.

Parameters

<i>strtime</i>	string representing time.
----------------	---------------------------

#### 3.2.3.2 void internalTime::toUNIXTime ( )

Member Function, providing UNIX time.

Member function, converting stored time to UNIX time.

### 3.2.4 Member Data Documentation

#### 3.2.4.1 int internalTime::year

Stores Year as Integer

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

- [internalTime.hpp](#)
- [internalTime.cpp](#)

## 3.3 navigation Class Reference

```
#include <navigation.hpp>
```

### Public Member Functions

- void [read](#) ()  
*Member function read.*
- [navigation](#) (std::vector< std::string > fnames)  
*Constructor with Input files.*
- void [getPositionR](#) (ephemerisR &initialConditions, int h, [triple](#) &pos)  
*Function to compute GLONASS satellite positions.*
- void [getPositionGE](#) (ephemerisGE &initial, int t, [triple](#) &pos)  
*Function to compute GPS/Galileo/BeiDou satellite positions.*
- void [ecefToEllipsoidal](#) (const [triple](#) &ecef, [triple](#) &ellipsoid)  
*Function to convert ECEF to ellipsoidal coordinates.*
- void [satElevAzim](#) ([triple](#) &markerECEF, [triple](#) &sat, [triple](#) &markerEllip, double &elevation, double &azimuth)  
*Function to compute satellite elevation and azimuth.*
- int [computeIPP](#) (const [triple](#) &marker, const [triple](#) &sat, const double &rh, [triple](#) &IPP, double &coschi)  
*Function to compute IPP and zenith angle over IPP.*

## Public Attributes

- `std::vector< std::string > fileNames`  
*list of file names to read from*
- `float version`  
*Stores RINEX version.*
- `int leapSeconds`  
*Stores leapSeconds from Navigation files.*
- `std::vector< std::vector< ephemerisGE > > ephemeris\_G`  
*Vector to store objects of type ephemerisGE for GPS.*
- `std::vector< std::vector< ephemerisGE > > ephemeris\_E`  
*Vector to store objects of type ephemerisGE for Galileo.*
- `std::vector< std::vector< ephemerisR > > ephemeris\_R`  
*Vector to store objects of type ephemerisR for GLONASS.*
- `std::vector< std::vector< ephemerisGE > > ephemeris\_C`  
*Vector to store objects of type ephemerisGE for BeiDou.*

## Private Member Functions

- `navigation ()`
- `float eccAnomaly (float M, float e)`  
*Function to compute eccentricity anomaly Ek.*
- `void applyRotations (float &Lk, float &ik, float &uk, float &rk, triple &pos)`  
*This Function apply rotations around uk, ik and Lk.*

### 3.3.1 Detailed Description

#### Author

Muhammad Owais

#### Date

05/12/16

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 `navigation::navigation ( std::vector< std::string > fnames )`

Constructor with Input files.

Constructs navigation object by reading input navigation files defined by fnames.

#### Parameters

<i>fnames</i>	Vector of Navigation file names.
---------------	----------------------------------

### 3.3.2.2 navigation::navigation ( ) [private]

Default Constructor. Hidden, cannot be used.

### 3.3.3 Member Function Documentation

#### 3.3.3.1 void navigation::applyRotations ( float & Lk, float & ik, float & uk, float & rk, triple & pos ) [private]

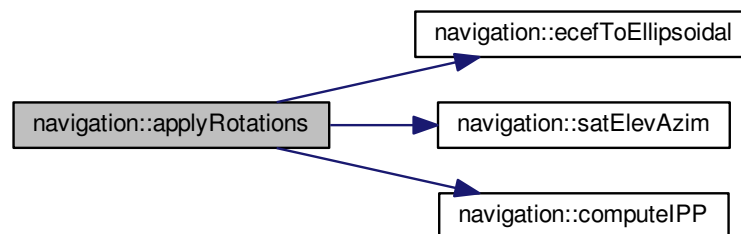
This Function apply rotations around uk, ik and Lk.

This Function apply rotations around uk, ik and Lk, Rotation ==  $\begin{bmatrix} Xk \\ rk \\ Yk \\ 0 \\ Zk \\ 0 \end{bmatrix}$  Where R1 and R3 are the rotation matrices defined at: [http://www.navipedia.net/index.php/Transformation\\_between\\_Terrestrial\\_Frames](http://www.navipedia.net/index.php/Transformation_between_Terrestrial_Frames) By Hernández-Pajares, Technical University of Catalonia, Spain.

#### Parameters

<i>Lk</i>	Longitude of the ascending node LAMBDaK.
<i>ik</i>	Inclination of the orbital plane.
<i>uk</i>	Argument of latitude.
<i>rk</i>	Radial distance rk.
<i>pos</i>	triple object returned with computed coordinates.

Here is the call graph for this function:



#### 3.3.3.2 int navigation::computeIPP ( const triple & marker, const triple & sat, const double & rh, triple & IPP, double & coschi )

Function to compute IPP and zenith angle over IPP.

This function computes IPP (Ionospheric Pierce Point) in ECEF cartesian coordinates and zenith angle over IPP using sphere-line equation. Reference height of ionosphere, marker (receiver station), and satellite position are given as inputs. An integer status is returned describing solution type.



## Parameters

<i>marker</i>	ECEF cartesian coordinates for marker (receiver station) as a <a href="#">triple</a> object.
<i>sat</i>	ECEF cartesian coordinates for satellite as a <a href="#">triple</a> object.
<i>rh</i>	Ionosphere reference height in Kilometers.
<i>IPP</i>	Output ECEF cartesian coordinates for IPP as a <a href="#">triple</a> object.
<i>coschi</i>	Output zenith angle over IPP.

3.3.3.3 float navigation::eccAnomaly ( float *M*, float *e* ) [private]

Function to compute eccentricity anomaly  $E_k$ .

This Function computes eccentricity anomaly  $E_k$  by Solving (iteratively) the Kepler equation for the eccentricity anomaly, using Newton–Raphson method, Equation  $\rightarrow M_k = E_k - (e * \sin(E_k))$

## Parameters

<i>M</i>	mean anomaly for reference time tk.
<i>e</i>	eccentricity.

3.3.3.4 void navigation::ecefToEllipsoidal ( const [triple](#) & *ecef*, [triple](#) & *ellipsoid* )

Function to convert ECEF to ellipsoidal coordinates.

This function converts ECEF cartesian coordinates  $(x, y, z)$  to ellipsoidal coordinates  $(\varphi, \lambda, h)$  respectively latitude, longitude, and height.

## Parameters

<i>ecef</i>	ECEF cartesian coordinates.
<i>ellipsoid</i>	Output ellipsoidal coordinates $(\varphi, \lambda, h)$ .

3.3.3.5 void navigation::getPositionGE ( [ephemerisGE](#) & *initial*, int *t*, [triple](#) & *pos* )

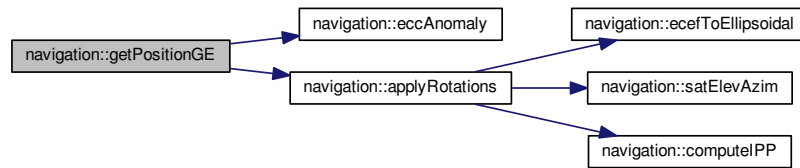
Function to compute GPS/Galileo/BeiDou satellite positions.

This function calculates GPS/Galileo/BeiDou satellite coordinates given an [ephemerisGE](#) object and time for which coordinates are required.

## Parameters

<i>initial</i>	<a href="#">ephemerisGE</a> object containing initial Keplerian elements.
<i>t</i>	Integer time for which coordinates are to be computed.
<i>pos</i>	<a href="#">triple</a> object returned with computed coordinates.

Here is the call graph for this function:



### 3.3.3.6 void navigation::getPositionR ( ephemerisR & *initialConditions*, int *h*, triple & *pos* )

Function to compute GLONASS satellite positions.

This function calculates GLONASS satellite coordinates given an ephemerisR object, and a step size.

#### Parameters

<i>initialConditions</i>	ephemerisR object containing initial conditions.
<i>h</i>	Integer step size for next coordinate.
<i>pos</i>	triple object returned with computed coordinates.

### 3.3.3.7 void navigation::read ( )

Member function read.

Member function read parses input navigation files and constructs internal navigation structure.

Here is the call graph for this function:



### 3.3.3.8 void navigation::satElevAzim ( triple & *markerECEF*, triple & *sat*, triple & *markerEllip*, double & *elevation*, double & *azimuth* )

Function to compute satellite elevation and azimuth.

This function computes satellite elevation and azimuth given marker (receiver station) position in ECEF and ellipsoidal coordinates and satellite position in ECEF coordinates. This function implements elevation/azimuth computation as described in [Transformations between ECEF and ENU coordinates](#) J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain.

## Parameters

<i>markerECEF</i>	ECEF cartesian coordinates for marker (receiver station) as a <a href="#">triple</a> object.
<i>sat</i>	ellipsoidal coordinates for satellite as a <a href="#">triple</a> object.
<i>markerEllip</i>	ellipsoidal coordinates for marker (receiver station) as a <a href="#">triple</a> object.
<i>elevation</i>	Output satellite elevation.
<i>azimuth</i>	Output satellite azimuth.

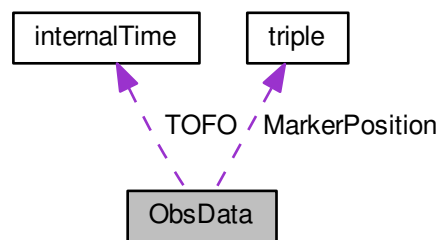
The documentation for this class was generated from the following files:

- [navigation.hpp](#)
- [navigation.cpp](#)

### 3.4 ObsData Class Reference

```
#include <ObsData.hpp>
```

Collaboration diagram for ObsData:



#### Public Member Functions

- void [read](#) ()  
*Member function read.*
- [ObsData](#) (std::vector< std::string > fvec, std::string sysString)  
*Constructor with Input files, and system string.*
- void [cleanUp](#) ()  
*Clean-up function.*
- void [pre\\_process](#) (int minArcLen, int intrpolIntrvl, int deg)  
*Function to perform preprocessing.*
- void [buildB](#) ()  
*Builds matrix B.*
- void [dumpRawMatrix](#) (const double \*mat, int &dim1, int &dim2)  
*Function to dump raw matrix.*

## Public Attributes

- `std::vector< std::string > fnames`  
*list of file names to read from*
- `triple MarkerPosition`  
*triple Object to store receiver-station position*
- `float version`  
*Stores RINEX version of observation files.*
- `int interval`  
*Interval between observations in data file.*
- `bool hasGPS`  
*Flag to indicate whether Data file contains GPS Data.*
- `bool hasGLO`  
*Flag to indicate whether Data file contains GLONASS Data.*
- `bool hasGAL`  
*Flag to indicate whether Data file contains Galileo Data.*
- `bool hasBEI`  
*Flag to indicate whether Data file contains BeiDou Data.*
- `bool readGPS`  
*Flag to indicate whether to process GPS Data.*
- `bool readGLO`  
*Flag to indicate whether to process GLONASS Data.*
- `bool readGAL`  
*Flag to indicate whether to process Galileo Data.*
- `bool readBEI`  
*Flag to indicate whether to process BeiDou Data.*
- `bool hasTOFO`  
*Time of first observation flag.*
- `std::string TOFO_system`  
*Time system of first observation from observation Header.*
- `internalTime TOFO`  
*internalTime Object to store Time of first observation*
- `std::vector< int > timeline_main`  
*Integer vector to store epochs in UNIX time.*
- `std::vector< std::vector< float > > GPS_ucTEC`  
*Vectors to store raw non-calibrated TEC for GPS Satellites.*
- `std::vector< std::vector< float > > GLO_ucTEC`  
*Vectors to store raw non-calibrated TEC for GLONASS Satellites.*
- `std::vector< std::vector< float > > GAL_ucTEC`  
*Vectors to store raw non-calibrated TEC for Galileo Satellites.*
- `std::vector< std::vector< float > > BDU_ucTEC`  
*Vectors to store raw non-calibrated TEC for BeiDou Satellites.*
- `std::vector< double > S`  
*Stores vector S (non-calibrated TEC).*
- `std::vector< int > S_arcnum`  
*Stores arc numbers for S.*
- `std::vector< int > S_prn`  
*Stores Satellite IDs for S.*
- `int size_of_S`  
*Indicates size of S.*
- `int numArcs`

- *Indicates total number of arcs.*
- double \* [B](#)  
*Stores matrix B.*
- std::vector< [ptr\\_pair](#) > [arcs](#)  
*Initial non-zero arc pointers.*
- std::vector< [ptr\\_pair](#) > [arcs2](#)  
*Arc pointers without zeros.*
- std::vector< [ptr\\_pair](#) > [arcs3](#)  
*Arc pointers without gaps.*

## Private Member Functions

- [ObsData](#) ()  
*default hidden Constructor*
- void [setSysFlags](#) (std::string sysString)  
*Sets system flags.*
- void [setArcStartEnd](#) ()  
*Sets Arc pointers using [ptr\\_pair](#) objects.*
- int [lagrangeInterpolation](#) (float \*target, float \*s, float \*e, int deg)  
*Function to perform lagrange interpolation.*

## 3.4.1 Detailed Description

### Author

Muhammad Owais

### Date

05/12/16

## 3.4.2 Constructor & Destructor Documentation

### 3.4.2.1 [ObsData::ObsData](#) ( std::vector< std::string > *fvec*, std::string *sysString* )

Constructor with Input files, and system string.

Constructs observation object by setting input observation file name vector [fnames](#) given file names and setting system flags given system string.

#### Parameters

<i>fvec</i>	Vector of observation file names.
<i>sysString</i>	string (any combination of 'G','R','E','C') defining constellations being processed.

Here is the call graph for this function:



### 3.4.3 Member Function Documentation

#### 3.4.3.1 void ObsData::buildB ( )

Builds matrix B.

This function builds and stores matrix B.

#### 3.4.3.2 void ObsData::cleanUp ( )

Clean-up function.

This function cleans up internal workspace, should be called before end of object's lifetime.

#### 3.4.3.3 void ObsData::dumpRawMatrix ( const double \* *mat*, int & *dim1*, int & *dim2* )

Function to dump raw matrix.

This Function dumps raw matrix to standard output stream, usefull in debugging purposes.

##### Parameters

<i>mat</i>	pointer to stored matrix.
<i>dim1</i>	First dimension of matrix (number of rows).
<i>dim2</i>	Second dimension of matrix (number of columns).

#### 3.4.3.4 int ObsData::lagrangeInterpolation ( float \* *target*, float \* *s*, float \* *e*, int *deg* ) [private]

Function to perform lagrange interpolation.

This function performs lagrange Interpolation needed in preprocessing phase, given a required degree for interpolation.

##### Parameters

<i>target</i>	pointer to the value being interpolated.
---------------	--

## Parameters

<i>s</i>	start pointer of the arc.
<i>e</i>	end pointer of the arc.
<i>deg</i>	degree of Interpolation.

3.4.3.5 void `ObsData::pre_process` ( int *minArcLen*, int *intrpollIntrvl*, int *deg* )

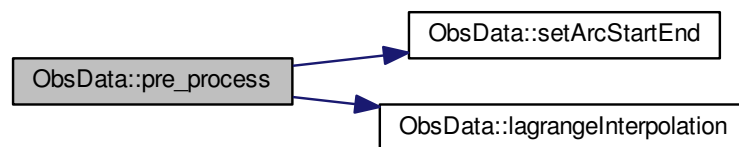
Function to perform preprocessing.

This function performs preprocessing by filling gaps using lagrange interpolation and removing phase jumps using quartiles and Inter Quartile Range.

## Parameters

<i>minArcLen</i>	minimum data duration(Seconds) to consider an arc valid.
<i>intrpollIntrvl</i>	Maximum gap duration (Seconds) to interpolate.
<i>deg</i>	Degree of Interpolation, passed to <a href="#">lagrangeInterpolation</a> .

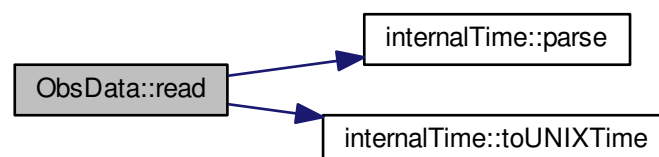
Here is the call graph for this function:

3.4.3.6 void `ObsData::read` ( )

Member function read.

Member function read parses input observation files and constructs internal observation structure.

Here is the call graph for this function:





**3.4.3.7 void ObsData::setArcStartEnd ( ) [private]**

Sets Arc pointers using [ptr\\_pair](#) objects.

This function sets Arc pointers to start/end pairs using [ptr\\_pair](#), which serve as input arcs to preprocessing phase.

**3.4.3.8 void ObsData::setSysFlags ( std::string sysString ) [private]**

Sets system flags.

This function sets system flags given sysString, to indicate which constellations are processed.

**Parameters**

<i>sysString</i>	string (any combination of 'G','R','E','C') indicating constellations being processed.
------------------	--

**3.4.4 Member Data Documentation****3.4.4.1 std::vector<ptr\_pair> ObsData::arcs**

Initial non-zero arc pinters.

[ptr\\_pair](#) Object containing Initial non-zero arcs, without preprocessing being applied.

**3.4.4.2 std::vector<ptr\_pair> ObsData::arcs2**

Arc pointers without zeros.

[ptr\\_pair](#) Object containing arcs, without leading and trailing zeros.

**3.4.4.3 std::vector<ptr\_pair> ObsData::arcs3**

Arc pointers without gaps.

[ptr\\_pair](#) Object containing arcs, with gaps removed by [lagrangeInterpolation](#) and phase jumps removed. These are the processed Arcs.

**3.4.4.4 double\* ObsData::B**

Stores matrix B.

This is stored matrix B. B is a boolean matrix relating each value in vector S to a given arc number. The  $i^{th}$  row of B has only one non-zero in the  $j^{th}$  column, relating  $i^{th}$  value in vector S to  $j^{th}$  arc number defined by [S\\_arcnum](#). Size of B is ( [size\\_of\\_S](#) x [numArcs](#) ).

#### 3.4.4.5 `std::vector< std::vector<float> > ObsData::BDU_ucTEC`

Vectors to store raw non-calibrated TEC for BeiDou Satellites.

This is a Vector of float-vectors, where first index is the Satellite prn-id and the second index is the raw non-calibrated TEC for BeiDou satellites corresponding to the epoch index in [timeline\\_main](#).

#### 3.4.4.6 `std::vector< std::vector<float> > ObsData::GAL_ucTEC`

Vectors to store raw non-calibrated TEC for Galileo Satellites.

This is a Vector of float-vectors, where first index is the Satellite prn-id and the second index is the raw non-calibrated TEC for Galileo satellites corresponding to the epoch index in [timeline\\_main](#).

#### 3.4.4.7 `std::vector< std::vector<float> > ObsData::GLO_ucTEC`

Vectors to store raw non-calibrated TEC for GLONASS Satellites.

This is a Vector of float-vectors, where first index is the Satellite prn-id and the second index is the raw non-calibrated TEC for GLONASS satellites corresponding to the epoch index in [timeline\\_main](#).

#### 3.4.4.8 `std::vector< std::vector<float> > ObsData::GPS_ucTEC`

Vectors to store raw non-calibrated TEC for GPS Satellites.

This is a Vector of float-vectors, where first index is the Satellite prn-id and the second index is the raw non-calibrated TEC for GPS satellites corresponding to the epoch index in [timeline\\_main](#).

#### 3.4.4.9 `bool ObsData::hasTOFO`

Time of first observation flag.

Flag to indicate whether Time of first observation was present in observation Header.

#### 3.4.4.10 `int ObsData::numArcs`

Indicates total number of arcs.

Indicates total number of arcs formed. Arc numbers are defined by pre\_processing phase using [pre\\_process](#).

#### 3.4.4.11 `std::vector<double> ObsData::S`

Stores vector S (non-calibrated TEC).

This vector stores all computed non-calibrated TEC values, arranged by epochs. This is the input vector given to the system solver.

#### 3.4.4.12 `std::vector<int> ObsData::S_arcnum`

Stores arc numbers for [S](#).

This vector stores for each element in [S](#) , a corresponding value indicating the its arc number. Arc numbers are defined by pre\_processing phase using [pre\\_process](#).

#### 3.4.4.13 `std::vector<int> ObsData::S_prn`

Stores Satellite IDs for [S](#).

This vector stores for each element in [S](#) , a corresponding value indicating the its Satellite ID.

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

- [ObsData.hpp](#)
- [ObsData.cpp](#)

## 3.5 ptr\_pair Class Reference

```
#include <ptr_pair.hpp>
```

### Public Member Functions

- [ptr\\_pair](#) ()  
*Default constructor.*
- [ptr\\_pair](#) (float \*s, float \*e)  
*Custom constructor.*

### Public Attributes

- float \* [start](#)  
*Start pointer.*
- float \* [end](#)  
*End pointer.*

### 3.5.1 Detailed Description

#### Author

Muhammad Owais

#### Date

05/12/16

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 `ptr_pair::ptr_pair ( )`

Default constructor.

Default constructor, creates `ptr_pair` object with NULLL start and end pointers.

#### 3.5.2.2 `ptr_pair::ptr_pair ( float * s, float * e )`

Custom constructor.

Constructor, creates `ptr_pair` object with start and end pointers set to given pointers.

## Parameters

s	Input start pointer for new <a href="#">ptr_pair</a> object.
e	Input end pointer for new <a href="#">ptr_pair</a> object.

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

- [ptr\\_pair.hpp](#)
- [ptr\\_pair.cpp](#)

## 3.6 triple Class Reference

```
#include <triple.hpp>
```

### Public Member Functions

- void [dump](#) (std::ostream &s)  
*Member function dump.*

### Public Attributes

- double [X](#)  
*Stores X Coordinate.*
- double [Y](#)  
*Stores Y Coordinate.*
- double [Z](#)  
*Stores Z Coordinate.*

### 3.6.1 Detailed Description

#### Author

Muhammad Owais

#### Date

05/12/16

### 3.6.2 Member Function Documentation

#### 3.6.2.1 void triple::dump ( std::ostream & s )

Member function dump.

Member function dump output coordinates into a given output stream.

**Parameters**

s	output stream
---	---------------

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

- [triple.hpp](#)
- triple.cpp

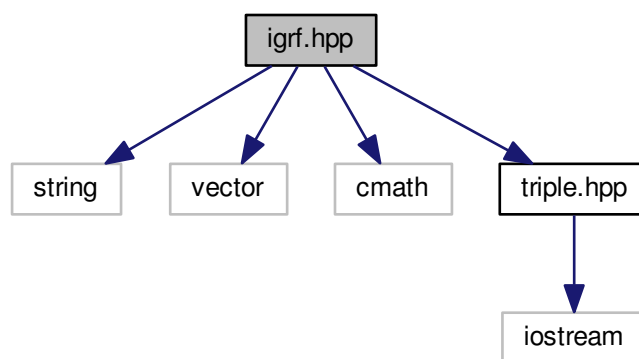
## Chapter 4

# File Documentation

### 4.1 igrf.hpp File Reference

This class implements IGRF model.

```
#include <string>
#include <vector>
#include <cmath>
#include "triple.hpp"
Include dependency graph for igrf.hpp:
```



#### Classes

- class [igrf](#)

#### 4.1.1 Detailed Description

This class implements IGRF model.

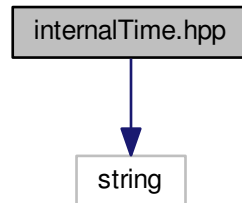
This class implements IGRF (International Geomagnetic Reference Field) model, as defined in [IGRF Web Site](#). An instance of this class could be created using a generation of IGRF coefficients, currently [IGRF-12](#) which would be valid for years 1900 to 2020.

## 4.2 internalTime.hpp File Reference

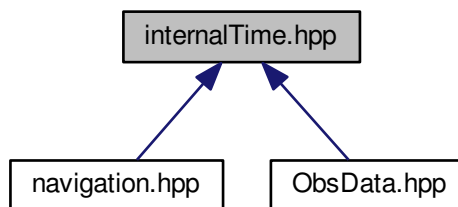
Class defining internal time format.

```
#include <string>
```

Include dependency graph for internalTime.hpp:



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



### Classes

- class [internalTime](#)

#### 4.2.1 Detailed Description

Class defining internal time format.

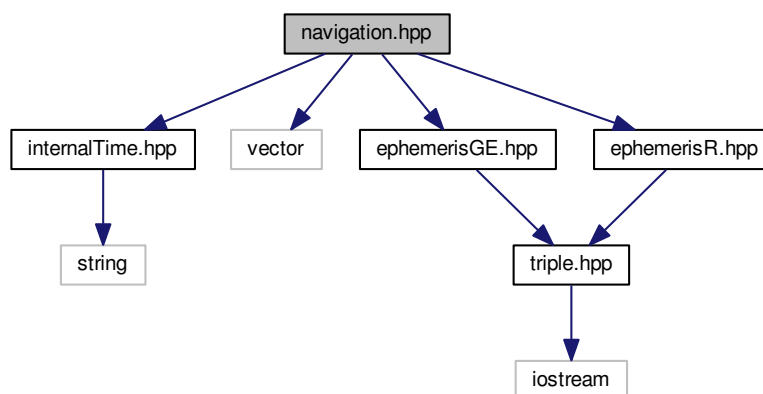
This Class Defines Internal time which is based on Unix Time. It stores the normal Date/Time as (Year,Month,Day,Hour,Minute,Second), while also providing equivalent UNIX Time. An instance of this class could be generated by explicitly providing normal Date/Time values or by providing a string which would be parse to store time in both formats.



## 4.3 navigation.hpp File Reference

This is class navigation data.

```
#include "internalTime.hpp"
#include <vector>
#include "ephemerisGE.hpp"
#include "ephemerisR.hpp"
Include dependency graph for navigation.hpp:
```



### Classes

- class [navigation](#)

#### 4.3.1 Detailed Description

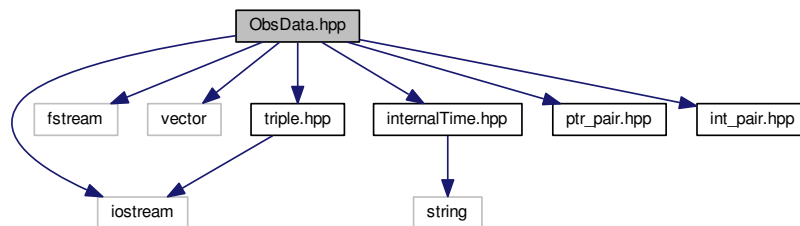
This is class navigation data.

This class defines navigation data, stored after reading RINEX navigation files, for different constellations.

## 4.4 ObsData.hpp File Reference

Class defining observation data.

```
#include <iostream>
#include <fstream>
#include <vector>
#include "internalTime.hpp"
#include "triple.hpp"
#include "ptr_pair.hpp"
#include "int_pair.hpp"
Include dependency graph for ObsData.hpp:
```



## Classes

- class [ObsData](#)

### 4.4.1 Detailed Description

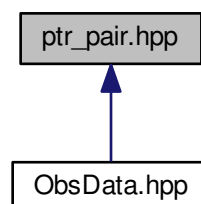
Class defining observation data.

This Class Defines observation data handling, including reading from observation files and storing in internal data structure, the raw non-calibrated TEC from phase observables. This class also includes preprocessing routines being applied to internal data structure, and allot of dump routines for debugging and plotting arc states.

## 4.5 ptr\_pair.hpp File Reference

Class defining pointer pairs.

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



## Classes

- class [ptr\\_pair](#)

### 4.5.1 Detailed Description

Class defining pointer pairs.

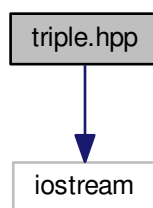
This Class Defines pointer pairs objects used in preprocessing to define arcs. Each arc could be defined as a [ptr\\_pair](#) object having a start pointer (pointer to first value in arc) and an end pointer (pointer to last value in arc).

## 4.6 triple.hpp File Reference

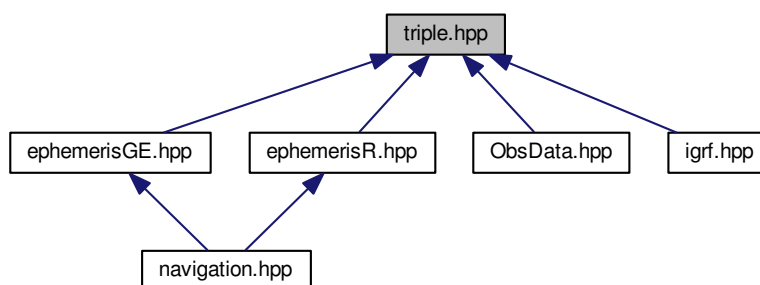
This class defines a 3-D Coordinate.

```
#include <iostream>
```

Include dependency graph for triple.hpp:



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



## Classes

- class [triple](#)

### 4.6.1 Detailed Description

This class defines a 3-D Coordinate.

# Index

applyRotations  
    navigation, 10  
arcs  
    ObsData, 19  
arcs2  
    ObsData, 19  
arcs3  
    ObsData, 19  
  
B  
    ObsData, 19  
BDU\_ucTEC  
    ObsData, 19  
buildB  
    ObsData, 17  
  
cleanUp  
    ObsData, 17  
computeIPP  
    navigation, 10  
  
dump  
    triple, 23  
dumpRawMatrix  
    ObsData, 17  
  
eccAnomaly  
    navigation, 11  
ecefToEllipsoidal  
    navigation, 11  
  
GAL\_ucTEC  
    ObsData, 20  
GLO\_ucTEC  
    ObsData, 20  
GPS\_ucTEC  
    ObsData, 20  
getMODIP  
    igrf, 6  
getPositionGE  
    navigation, 11  
getPositionR  
    navigation, 12  
  
hasTOFO  
    ObsData, 20  
  
igrf, 5  
    getMODIP, 6  
    igrf, 5  
igrf.hpp, 25

internalTime, 6  
    internalTime, 7  
    parse, 8  
    toUNIXTime, 8  
    year, 8  
internalTime.hpp, 26  
  
lagrangeInterpolation  
    ObsData, 17  
  
navigation, 8  
    applyRotations, 10  
    computeIPP, 10  
    eccAnomaly, 11  
    ecefToEllipsoidal, 11  
    getPositionGE, 11  
    getPositionR, 12  
    navigation, 9  
    read, 12  
    satElevAzim, 12  
navigation.hpp, 27  
numArcs  
    ObsData, 20  
  
ObsData, 14  
    arcs, 19  
    arcs2, 19  
    arcs3, 19  
    B, 19  
    BDU\_ucTEC, 19  
    buildB, 17  
    cleanUp, 17  
    dumpRawMatrix, 17  
    GAL\_ucTEC, 20  
    GLO\_ucTEC, 20  
    GPS\_ucTEC, 20  
    hasTOFO, 20  
    lagrangeInterpolation, 17  
    numArcs, 20  
    ObsData, 16  
    pre\_process, 18  
    read, 18  
    S, 20  
    S\_arcnum, 20  
    S\_prn, 21  
    setArcStartEnd, 19  
    setSysFlags, 19  
ObsData.hpp, 27  
  
parse

- internalTime, [8](#)
- pre\_process
  - ObsData, [18](#)
- ptr\_pair, [21](#)
  - ptr\_pair, [22](#)
- ptr\_pair.hpp, [28](#)
- read
  - navigation, [12](#)
  - ObsData, [18](#)
- S
  - ObsData, [20](#)
- S\_arcnum
  - ObsData, [20](#)
- S\_prn
  - ObsData, [21](#)
- satElevAzim
  - navigation, [12](#)
- setArcStartEnd
  - ObsData, [19](#)
- setSysFlags
  - ObsData, [19](#)
- toUNIXTime
  - internalTime, [8](#)
- triple, [23](#)
  - dump, [23](#)
- triple.hpp, [29](#)
- year
  - internalTime, [8](#)