Master in High Performance Computing

Multi Constellation GNSS based TEC Calibration

Supervisor(s):
Luigi Ciraolo SUPERVISOR,
Katy Alazo SUPERVISOR,
Ivan Girotto SUPERVISOR

Candidate:
Muhammad Owais Arain

2nd EDITION
2015–2016

TABLE OF CONTENTS

Introduction

The purpose of this work is to provide an improved version of multi-constellation GNSS TEC calibration software[29], used by Telecommunications/ICT for Development Laboratory, Applied Physics section, ICTP. This tool is based on an arc-by-arc GPS based TEC calibration algorithm[20]. The current version of the code runs only on Windows operating system and does not use standard libraries, Furthermore the code is serial and without sufficient documentation. It is very difficult for the T/ICT4D Lab to maintain the current version. A web model based on this tool is running and accessible from ICTP website[30]. Improvements in this tool would also increase the functionality of this published service for external users. Furthermore it would enable us to build on top of this framework a more flexible and scalable service. We start this work from scratch as a new development project, based on C++ (using OOP features) and with the documentation describing the calibration technique. We also make use of existing executables for benchmarking. We were able to achieve impressive performance improvements over current version by integrating standard high performance libraries[27], besides that we now have a software which is well structured, uses OOP features, is well documented, and therefore it would now be possible for T/ICT4D Lab, Applied Physics group ICTP to maintain this software.

Acknowledgements

This project as presented in this thesis and its related activities were supported by Abdus Salam International Center for Theoretical Physics (ICTP) under the framework of Training and research in Italian Laboratories (TRIL). I would like to express my sincere gratitude for this institution's support towards participation in this Master, activities presented in this thesis, and for all the opportunities for professional growth. I would like to thank my supervisors Luigi Ciraolo, Katy Alazo, and Ivan Girotto for their support and guidance throughout this activity. I would also like to thank all my Instructors who kept me encouraged and focused during the Master. Moreover I am also thankful to all my class mates, friends, and family for their support.

List of Tables

List of Figures

Chapter 1

Calibration Algorithm

1.1 TEC:

Total Electron Content (TEC) is a descriptive quantity of Earth's ionosphere, having a practical importance. TEC data is derived from carrier phase measurements of Global Navigation Satellite Systems, but is biased by satellite-receiver biases. TEC calibration is the process of determining and removing these biases from raw biased TEC.

The basic relation used to calibrate the TEC is given by:

$$S_{\Phi} = sTEC + \beta_{arc}$$

Where:

S_{Φ} is the ionospheric delay from the raw carrier phase observations,

β_{arc} is the Arc-Offset, a constant to be determined for each arc[20].

An arc here means a set of continuous observations related to a given receiver and satellite pair. β_{arc} represents the contribution of receiver and satellite biases.

The $sTEC$ and β_{arc} are unknowns to be determined by the so-called calibration or de-biasing process.

1.2 The Thin Shell Model:

A two-dimensional thin shell model at 350 km is assumed to define the mapping function between the slant and vertical TEC. The vertical ionospheric variation over the thin shell is expressed as a function of the MODIP and the Local Time of the IPPs.

1.3 The current version:

The current version of the software provides a single-station, multi-constellation solution. Constellations being processed are GPS, GLONASS, Galileo and BeiDou. Several stations could be processed for a defined period in a sequence.

The software is written in FORTRAN using the Compaq Visual FORTRAN 6.6 environment, with IMSL library, used for linear algebra operations[21].

1.4 Multi-constellation solution:

The current version allows the processing of multi-constellation (Section 1.3) solution. There are major differences in the signal characteristics among different constellations, which is relevant to TEC estimation. Nevertheless, the calibration technique could be applied to multi-constellation measurements, since the satellite and receiver biases are not explicitly separated.

1.5 Calibration Parameters

Here we provide a list of calibration parameters, which strongly govern the calibration process. Some of these parameters are fixed and some are allowed to be changed by user.

Parameter	Value	Description
href	350	Reference ionosphere height
minimumelevation	5 degrees	Minimum Satellite Elevation
maxlosstime	5 minutes	Maximum arc discontinuity
decday	12 hours	Data duration to discard from start and end
minblocktime	$2\times decday+24\,hours$	Minimum data to load for calibration
minimumarclength	2 hours	Minimum arc length
intervaltime	30	Observation interval in data files
samplingtime	10 minutes	Refreshing interval for vTECeq
Constellation Codes	GREC	Constellations to add in calibration G(GPS) R(GLONASS) E(Galileo) C(BeiDou)

Table 1.1: Calibration Parameters

1.6 Non calibrated TEC from the GNSS observables:

The non calibrated TEC is calculated using the first order approximation of the Appleton-Hartree formula for the refractive index for electromagnetic wave propagation in a cold magnetized plasma, i.e. the ionosphere (Section 2.4)

1.7 Calculation of the Geometry:

Satellite positions are calculated based on individual constellation's ICD documents(Section 2.5), followed by computation of geometrical parameters IPP and zenith angle over IPP (Section 2.6)

1.8 Preprocessing:

In preprocessing we account for arc discontinuities and phase jumps. This section is very important for a smooth calibration and is highly affected by data quality (Section 2.7).

1.9 Definition of vTEC model:

Representation of vTECeq by a 2D function and Thin Shell model. The vTECeq variation is expressed as a function of the variation of local time and MODIP (Section 2.8).

1.10 Calibration:

To perform calibration we Solve the equation system defined as:

$$S = AC + B\beta$$

where,

S is matrix of the non-calibrated TEC.

A is matrix of the data of the vTECeq variations in the 2D representation.

C is matrix of the solution of the coefficients of the function that defines vTECeq.

B is matrix of the arcs. Its values are 0 or 1 .

β is matrix of the Arc-Offset.

Here C and β are the unknowns to be determined by this calibration algorithm.

A double substitution is performed, to final expression:

$$S - AcS = (B - AcB)\beta$$

being $c = (A^T A)^{-1} A^T$

It is assumed that the functional representation of vTECeq is separated for each of the continuous interval of duration “samplingtime” (Table: 1.1), therefore it is possible to compute independently the values of c, $S - AcS$ and $B - AcB$ and accumulate in the end. We call this section as computing for blocks of A (A_Blocks).

Once full data interval is completed, the solution for β is computed (Section 1.10.1), and the calibration is formally completed:

calibrated $sTEC = S - B\beta$

calibration residuals are:

$$(S - AcS) - (B - AcB)\beta$$

1.10.1 Solution for β :

Assuming $B - AcB = B_{AcS}$ and $S - AcS = S_{AcS}$

we have:

$$B_{AcB}^T S_{AcS} = B_{AcB}^T B_{AcB} \beta$$

$$\beta = (B_{AcB}^T B_{AcB})^{-1} B_{AcB}^T S_{AcS}$$

The inverse of $B_{AcB}^T B_{AcB}$ is computed by LU factorization. We call this section Arc-Offset solution as we are computing for β (Arc-Offset).

1.10.2 vTECeq solution:

The solution for the coefficients of the function that defines the vTECeq is:

$$C = c(S - B\beta)$$

given

$$calibrated\ sTEC = S - B\beta \quad \text{computed earlier,}$$

we have

$$C = c(calibrated\ sTEC)$$

1.11 Matrix Dimensions in a single-station single-day calibration

Given 10 SV per epoch, 2 observations per minute, and "samplingtime" of 10 minutes, for one day TEC calibration:

time interval for the calibration = $24h + 12h \times 2 = 48h$ *

total quantity of epochs = $48 \times 60 \times 2 = 5760$

total quantity of observations = $5760 \times 10\,SV = 57600$

quantity of epochs in one samplingtime = $10\,minutes \times 2 = 20$

quantity of observations in one samplingtime = $20 \times 10\,SV = 200$

quantity of blocs of 10 mins (blocs of A) = $48 \times 60/10 = 288$

*This calibration technique is applied to non real time data. To process one day (24 hours), the previous and last 12 hours, at least, are discarded. This is to guaranty that full arcs are processed.

Matrix	**Description**	**in one ‘sampling time’**		**Full time interval**	
		Dim	Type	Dim	Type
S	The uncalibrated TEC observations	(200,1)		(57600,1)	real no sparse
A	the coefficients of the functional vTECeq	(200,9)	real no sparse	(57600,2592)	real sparse
c	$c=(A^T A)^{-1} A^T$	(9,200)	real no sparse	(2592, 57600)	real sparse
S_{AcS}	$S_{AcS}=S-AcS$	(200,1)		(57600,1)	real no sparse
B	The arcs, values are 0 or 1	(200, narcs)	real sparse	(57600, narcs)	real sparse
B_{AcB}	$B_{AcB}=B-AcB$	(200, narcs)		(57600, narcs)	real sparse

The solution

Matrix	Description				
C	The coefficients of the function that defines vTECeq	(9,1)		(2592,1)	
β	Solution of the Arc-Offset	(narcs,1)			

Table 1.2: Sample Matrix Sizes

Chapter 2

Implementation

2.1 Input/Output file formats:

For calibration tool we use RINEX files which are structured formats having separate header and data sections within a single file as defined by IGS formats[1]. RINEX allows a user to exchange mixed GNSS data without worrying about differences in receivers being used. As of the latest RINEX version, one can find data for GPS, GLONASS, Gallileo, Beidou (Compass), QZSS, and SBAS. Different versions for this format exist which are:

1) RINEX Version 2
    i. Version 2.10[2]
    ii. Version 2.11[3]
2) RINEX Version 3
    i. Version 3.01[4]
    ii. Version 3.02[5]
    iii. Version 3.03[6]

RINEX format provides following file types:

1) Observation Data
2) Navigation Data
3) Meteorological Data

We use RINEX 3 for all computations mentioned in this thesis¬, and the type of files relevant for this activity are RINEX observation data and

¬Although we use RINEX 3, but its not adopted fully by the community and RINEX 2 data is still useful and could be used with this tool.

RINEX navigation data. We employed custom I/O routines to read data from these files.

2.2 Internal Data Structure:

After reading data from input files (observation/navigation) data is internally stored as a hash table, for fast querying. Indexing for this data is done by means of a hashing function. Inputs to this function are epoch and satellite ID. We maintain a single time-line which provides the epoch input parameter with a given satellite ID. Figure 2.1 explains how an observable is obtained from this structure by means of hashing function.

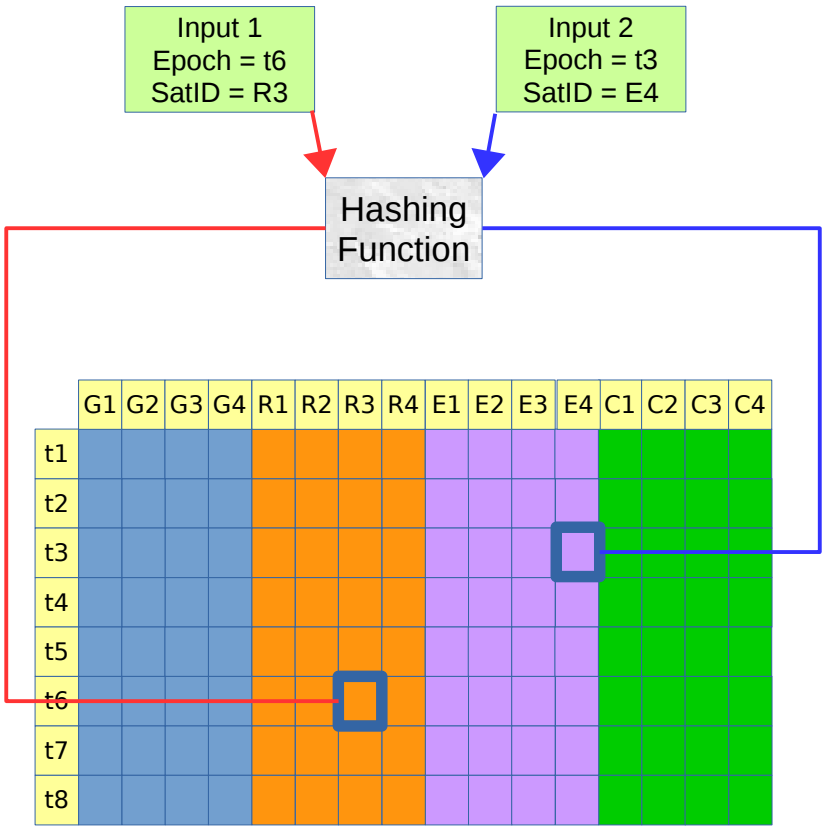

Figure 2.1: Hash Table - Internal Data Structure

2.3 Timing Systems:

A number of timing systems are involved in this calibration tool. Since we are dealing with a solution approach which combines various satellite constellations using their respective Time systems, in general for a mixed constellation solution we need to convert times expressed in multiple timing systems to a single time system (usually UTC). This makes it easy to express a time-dependent phenomenon and reporting results after calibration. We also need to opt for a time system which is easy to deal with-in a computer program. For this we use internal-time which is based on UNIX time. We convert a given epoch from its native time system to UTC and to UNIX time. Since after time conversion to UNIX time we get an integer, we can traverse better across a list of observables in a computer program. Following is a brief description of individual time systems used by this tool.

2.3.1 UT:

Universal time defined by earth's rotation, where UT0 is "raw" uncorrected UT, UT1 is corrected for polar wandering, UT2 is corrected for seasonal variations in the earth's rotation speed.

2.3.2 TAI:

Is the international atomic time with difference $TAI - UT2 = 0$ for the epoch at January 1st 1958[11].

2.3.3 UTC:

A compromise between TAI and UT1, and kept closer to UT1 to follow earth's rotation variations by adding leap-seconds[5]. Leap-seconds are refreshed periodically and provided by IERS[12].

2.3.4 GPST:

GPS time is a continuous time with no leap seconds. Start epoch for GPS is 0 hour UTC (midnight) between January 5th and 6th 1980[7]. At that epoch $TAI-UTC$ was $19\,seconds$, thence:

$$TAI=GPST+19\,seconds$$

$$UTC=GPST+19\,seconds-leap\,seconds$$

2.3.5 GLONASST:

GLONASS time is generated by the GLONASS central synchronizer and is synchronized with UTC(SU)[8], such that:

$$GLONASST=UTC(SU)+3^{h}-\tau,\ where\ |\tau|<1\,ms$$

Unlike GPS, Gallileo or BeiDou, GLONASS uses leap-seconds in the system.

2.3.6 GST:

Galileo time is a continuous time without leap seconds, and is kept synchronized with TAI. Start epoch for GST is 0 hour UTC (midnight) between August 21st and 22nd 1999[9].

2.3.7 BDT:

BeiDou time is a continuous time without leap seconds, and is kept synchronized with UTC. Start epoch for BDT is 0 hour UTC on January 1st 2006[10].

2.3.8 Internal Time:

Internal time is based on UNIX time, which is a continuous time without leap-seconds, counting number of seconds elapsed since start epoch. Start epoch for UNIX time is 0 hour UTC January 1st 1970.

e.g. November 25 2016 14:22:00 (UTC) is 1480083720 in UNIX time.

2.4 Computing raw slant TEC:

After reading Observation data we compute for each epoch and for each satellite in that epoch, raw(uncalibrated) slant TEC, from carrier-phase observables, using following relation:

$$TEC\varphi = \tau c\left(\frac{\varphi_1}{f_1} - \frac{\varphi_2}{f_2}\right)$$

$$where, \tau = \frac{1}{(40.3 \times f_{12})}$$

$$f_{12} = \frac{1}{f_2^2} - \frac{1}{f_1^2}$$

and $c = speed\ of\ light$

Here f_1 and f_2 are chosen carrier-frequencies, φ_1 and φ_2 are corresponding observables. These values after pre-processing are stored in array S (see system description).

2.5 Computing Satellite Positions:

After reading navigation files we need to compute satellite positions corresponding to every raw slant TEC computed in previous step. This is needed for computing geometry (Section 3.6) and for pre-processing (section 3.7). For GPS, Galileo, and BeiDou same algorithm can be used, whereas for GLONASS a separate algorithm is used.

2.5.1 GPS/Galileo/BeiDou Satellite Coordinates Computation:

Here we describe the parameters obtained from navigation files which would be used in this algorithm[13]. These parameters are refreshed periodically and cannot be used after prescribed time[14]…

…We dont consider BeiDou Geo-stationary satellite position calculation using this algorithm, and at the moment for this version of code we exclude these Satellites.

Parameter	Description
t_{oe}	Ephemerides reference epoch in seconds within the week
$\sqrt{a}$	Square root of semi-major axis
e	Eccentricity
M_o	Mean anomaly at reference epoch
ω	Argument of perigee
i_o	Inclination at reference epoch
Ω_o	Longitude of ascending node at the beginning of week
Δn	Mean motion difference
$\dot{i}$	Rate of inclination angle
$\dot{\Omega}$	Rate of node's right ascension
c_{uc}, c_{us}	Latitude argument correction
c_{rc}, c_{rs}	Orbital radius correction
c_{ic}, c_{is}	Inclination correction

Table 2.1: GPS,Galileo,and BeiDou broadcast ephemeris

Step 1 – Compute the time t_k from the ephemerides reference epoch t_{oe} using following relation:

$$t_k = t - t_{oe}$$

Note: t and t_{oe} are expressed in seconds in GPS week.

Check if $t_k > 302400\,Sec$, then subtract $604800\,Sec$ from t_k. If $t_k < -302400$ then add $604800\,Sec$ to t_k.

Step 2 – Compute the mean anomaly for t_k using following relation:

$$M_k = M_o + (\frac{\sqrt{\mu}}{\sqrt{a^3}} + \Delta n) t_k$$

Step 3 – Iterative Solution of Kepler equation for the eccentricity anomaly E_k :

$$M_k = E_k - e \sin(E_k)$$

Step 4 – Computing true anomaly v_k :

$$v_k = \arctan(\frac{\sqrt{1-e^2}\sin(E_k)}{\cos(E_k) - e})$$

Step 5 – Computing argument of latitude υ_k from the argument of perigee ω , true anomaly v_k , and corrections c_{uc}, c_{us} :

$$\upsilon_k = \omega + v_k + c_{uc}\cos 2(\omega + v_k) + c_{us}\sin 2(\omega + v_k)$$

Step 6 – Computing radial distance r_k using corrections c_{rc}, c_{rs} :

$$r_k = a(1 - e\cos E_k) + c_{rc}\cos 2(\omega + v_k) + c_{rs}\sin 2(\omega + v_k)$$

Step 7 – Computing inclination i_k of the orbital plane using inclination i_o at reference time t_{oe} and corrections c_{ic}, c_{is} :

$$i_k = i_o + \dot{i} t_k + c_{ic}\cos 2(\omega + v_k) + c_{is}\sin 2(\omega + v_k)$$

Step 8 – Computing longitude of the ascending node λ_k (Greenwich) . This computation uses right ascension at start of the current week Ω_o , the correction from the apparent sidereal time variation (Greenwich) between start of week and reference time $t_k = t - t_{oe}$, and the change in the longitude of the ascending node from reference time t_{oe} :

$$\lambda_k = \Omega_o + (\dot{\Omega} - \omega_E) t_k - \omega_E t_{oe}$$

where ω_E is earth's rotation rate according to WGS-84 datum[15].

Step 9 – Compute the coordinates in TRF, by applying three rotations around υ_k , i_k and λ_k :

$$\begin{bmatrix} X_k \\ Y_k \\ Z_k \end{bmatrix} = R_3(-\lambda_k) R_1(-i_k) R_3(-\upsilon_k) \begin{bmatrix} r_k \\ 0 \\ 0 \end{bmatrix}$$

where R_1 and R_3 are rotation matrices[16].

2.5.2 GLONASS Satellite Coordinates Computation:

In order to compute GLONASS satellite coordinates we shall perform numerical integration of differential equations describing motion of satellites[8]. Initial conditions broadcast in GLONASS navigation message are in PZ-90[15]. We must transform these initial conditions to an absolute (inertial) coordinate system[17]. Table 2.2 lists the parameters broadcast in GLONASS navigation message, which provides initial conditions for position and velocity. These values should be used to perform numerical integration within interval $|t - t_b| < 15\, minutes$. The accelerations due to solar and lunar gravitational perturbations are also given.

Parameter	Description
t_b	Ephemerides reference epoch
x	x Coordinate at t_b in PZ-90
y	y Coordinate at t_b in PZ-90
z	z Coordinate at t_b in PZ-90
v_x	Velocity x component at t_b in PZ-90
v_y	Velocity y component at t_b in PZ-90
v_z	Velocity z component at t_b in PZ-90
$\ddot{x}$	Moon and Sun acceleration x component at t_b
$\ddot{y}$	Moon and Sun acceleration y component at t_b
$\ddot{z}$	Moon and Sun acceleration z component at t_b

Table 2.2: GLONASS broadcast ephemeris

For numerical integration we use 4th order Runge-Kutta method. RKM uses a weighted average of the values of the derivative $f(t, y)$ taken at different points in the interval $t_n \le t \le t_{n+1}$. Relation between value of y at time $n+1$ and value of y at time n is given by:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$k_1 = hf(t_n, y_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$

$$k_4 = h f(t_n + h, y_n + k_3)$$

and h is the step size.

Following are the six orbital differential equations we need to solve for GLONASS position at a given time, as described in GLONASS ICD[8]:

$$dx/dt = v_x$$

$$dy/dt = v_y$$

$$dz/dt = v_z$$

$$dv_x/dt = -\frac{\mu}{r^3}x + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}x\left[1-\frac{5z^2}{r^2}\right] + \omega_3^2 x + 2\omega_3 v_y + \ddot{x}$$

$$dv_y/dt = -\frac{\mu}{r^3}y + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}y\left[1-\frac{5z^2}{r^2}\right] + \omega_3^2 y + 2\omega_3 v_x + \ddot{y}$$

$$dv_z/dt = -\frac{\mu}{r^3}z + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}z\left[3-\frac{5z^2}{r^2}\right] + \ddot{z}$$

where

$r = \sqrt{x^2 + y^2 + z^2}$

$\mu = 398600.44\ km^3/s^2$	Earth's universal gravitational parameter
$a_e = 6378.136\ km$	Earth's equatorial radius
$C_{20} = -1082.63 \times 10^{-6}$	zonal Geo-potential coefficient of spherical harmonic expansion
$\omega_3 = 0.7292115 \times 10^{-4} c^{-1}$	Earth's rotation rate

The values of $r, \mu, a_e, C_{20}, \omega_3$ given here are based on PZ-90 datum[15].

Considering initial satellite position (x, y, z) and the initial satellite velocity (v_x, v_y, v_z) at the reference time t_b we assume:

$$t_0 = t_b$$

$$\omega_{10} = x$$

$$\omega_{20} = y$$

$$\omega_{30} = z$$

$$\omega_{40} = v_x$$

$$\omega_{50} = v_y$$

$$\omega_{60} = v_z$$

Now for each of the six GLONASS differential equations we determine the k-parameters k_{pq}, where subscript p refers to one of the four integration parameters and subscript q refers to one of the six GLONASS differential equations:

$$k_{11} = h(\omega_{40})$$

$$k_{12} = h(\omega_{50})$$

$$k_{13} = h(\omega_{60})$$

$$k_{14} = h\left(-\frac{\mu}{r^3}\omega_{10} + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}\omega_{10}\left(1 - \frac{5}{r^2}\omega_{30}^2\right) + \omega_3^2\omega_{10} + 2\omega_3\omega_{50} + \ddot{x}\right)$$

$$k_{15} = h\left(-\frac{\mu}{r^3}\omega_{20} + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}\omega_{20}\left(1 - \frac{5}{r^2}\omega_{30}^2\right) + \omega_3^2\omega_{20} + 2\omega_3\omega_{40} + \ddot{y}\right)$$

$$k_{16} = h\left(-\frac{\mu}{r^3}\omega_{30} + \frac{3}{2}C_{20}\frac{\mu(a_e^2)}{r^5}\omega_{30}\left(3 - \frac{5}{r^2}\omega_{30}^2\right) + \ddot{z}\right)$$

$$k_{21} = h\left(\omega_{40} + \frac{1}{2}k_{14}\right)$$

$$k_{22}=h\left(\omega_{50}+\frac{1}{2}k_{15}\right)$$

$$k_{23}=h\left(\omega_{60}+\frac{1}{2}k_{16}\right)$$

$$k_{24}=h\left(-\frac{\mu}{r^3}\left(\omega_{10}+\frac{1}{2}k_{11}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{10}+\frac{1}{2}k_{11}\right)\right.$$
$$\left.\left(1-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{13}\right)^2\right)+\omega_3{}^2\left(\omega_{10}+\frac{1}{2}k_{11}\right)+2\,\omega_3\left(\omega_{50}+\frac{1}{2}k_{15}\right)+\ddot{x}\right)$$

$$k_{25}=h\left(-\frac{\mu}{r^3}\left(\omega_{20}+\frac{1}{2}k_{12}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{20}+\frac{1}{2}k_{12}\right)\right.$$
$$\left.\left(1-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{13}\right)^2\right)+\omega_3{}^2\left(\omega_{20}+\frac{1}{2}k_{12}\right)+2\,\omega_3\left(\omega_{40}+\frac{1}{2}k_{14}\right)+\ddot{y}\right)$$

$$k_{26}=h\left(-\frac{\mu}{r^3}\left(\omega_{30}+\frac{1}{2}k_{13}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{30}+\frac{1}{2}k_{13}\right)\left(3-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{13}\right)^2\right)+\ddot{z}\right)$$

$$k_{31}=h\left(\omega_{40}+\frac{1}{2}k_{24}\right)$$

$$k_{32}=h\left(\omega_{50}+\frac{1}{2}k_{25}\right)$$

$$k_{33}=h\left(\omega_{60}+\frac{1}{2}k_{26}\right)$$

$$k_{34}=h\left(-\frac{\mu}{r^3}\left(\omega_{10}+\frac{1}{2}k_{21}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{10}+\frac{1}{2}k_{21}\right)\right.$$
$$\left.\left(1-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{23}\right)^2\right)+\omega_3{}^2\left(\omega_{10}+\frac{1}{2}k_{21}\right)+2\,\omega_3\left(\omega_{50}+\frac{1}{2}k_{25}\right)+\ddot{x}\right)$$

$$k_{35}=h\left(-\frac{\mu}{r^3}\left(\omega_{20}+\frac{1}{2}k_{22}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{20}+\frac{1}{2}k_{22}\right)\right.$$
$$\left.\left(1-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{23}\right)^2\right)+\omega_3{}^2\left(\omega_{20}+\frac{1}{2}k_{22}\right)+2\,\omega_3\left(\omega_{40}+\frac{1}{2}k_{24}\right)+\ddot{y}\right)$$

$$k_{36}=h\left(-\frac{\mu}{r^3}\left(\omega_{30}+\frac{1}{2}k_{23}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{30}+\frac{1}{2}k_{23}\right)\left(3-\frac{5}{r^2}\left(\omega_{30}+\frac{1}{2}k_{23}\right)^2\right)+\ddot{z}\right)$$

$$k_{41}=h\left(\omega_{40}+k_{34}\right)$$

$$k_{42}=h\left(\omega_{50}+k_{35}\right)$$

$$k_{43}=h\left(\omega_{60}+k_{36}\right)$$

$$k_{44}=h\left(-\frac{\mu}{r^3}\left(\omega_{10}+k_{31}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{10}+k_{31}\right)\left(1-\frac{5}{r^2}\left(\omega_{30}+k_{33}\right)^2\right)+\omega_3^{\,2}\left(\omega_{10}+k_{31}\right)+2\,\omega_3\left(\omega_{50}+k_{35}\right)+\ddot{x}\right)$$

$$k_{45}=h\left(-\frac{\mu}{r^3}\left(\omega_{20}+k_{32}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{20}+k_{32}\right)\left(1-\frac{5}{r^2}\left(\omega_{30}+k_{33}\right)^2\right)+\omega_3^{\,2}\left(\omega_{20}+k_{32}\right)+2\,\omega_3\left(\omega_{40}+k_{34}\right)+\ddot{y}\right)$$

$$k_{46}=h\left(-\frac{\mu}{r^3}\left(\omega_{30}+k_{33}\right)+\frac{3}{2}C_{20}\frac{\mu\left(a_e^{\,2}\right)}{r^5}\left(\omega_{30}+k_{33}\right)\left(3-\frac{5}{r^2}\left(\omega_{30}+k_{33}\right)^2\right)+\ddot{z}\right)$$

Finally we get satellite positions and velocities at the next time step by:

$$\omega_{11}=\omega_{10}+\frac{1}{6}\left(k_{11}+2k_{21}+2k_{31}+k_{41}\right)$$

$$\omega_{21}=\omega_{20}+\frac{1}{6}\left(k_{12}+2k_{22}+2k_{32}+k_{42}\right)$$

$$\omega_{31}=\omega_{30}+\frac{1}{6}\left(k_{13}+2k_{23}+2k_{33}+k_{43}\right)$$

$$\omega_{41}=\omega_{40}+\frac{1}{6}\left(k_{14}+2k_{24}+2k_{34}+k_{44}\right)$$

$$\omega_{51} = \omega_{50} + \frac{1}{6}(k_{15} + 2k_{25} + 2k_{35} + k_{45})$$

$$\omega_{61} = \omega_{60} + \frac{1}{6}(k_{16} + 2k_{26} + 2k_{36} + k_{46})$$

2.6 Calculating Geometry:

After computing satellite positions we go on with computing two key geometrical parameters as shown in figure 2.2:

- IPP (Coordinates)
- zenith angle over IPP

These parameters are used to define vertical TEC equivalent (vTECeq) and to define block matrix A (Section 2.9).

M = Marker (Receiver)
C = Center (Earth)
S = Satellite
IPP = Ionospheric Pierce Point
z´ = Zenith angle over IPP
LoS = Line of Sight
H = Reference Ionosphere Height
R = Earth's Mean Radius

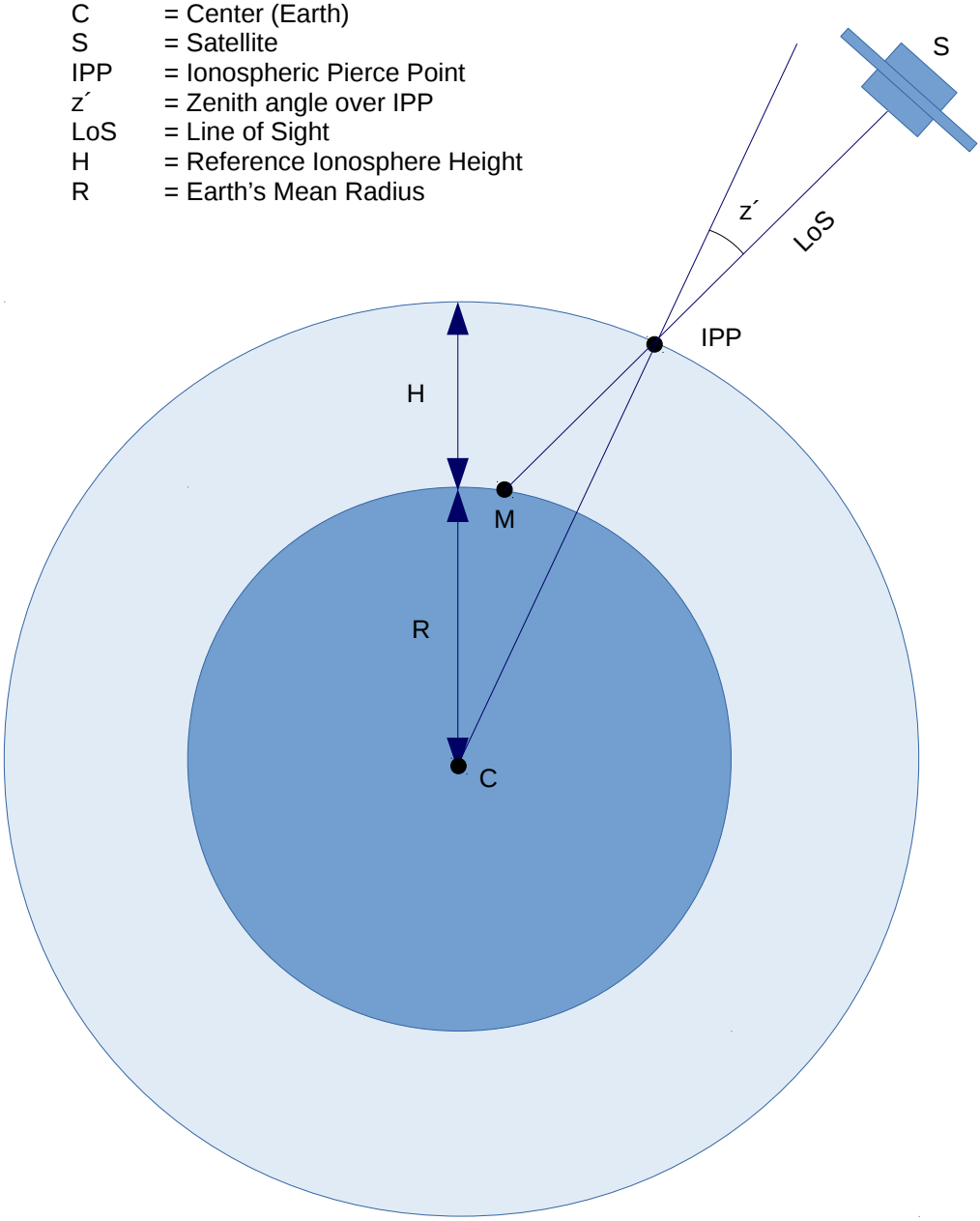

Figure 2.2: Satellite Geometry

2.7 Preprocessing:

The goal of this section is to detect and remove any irregularity in the data, before being served as input to the final system solver. We aim for this section to produce smooth continuous arcs, while keeping the number of arcs to minimum. Following are the sub-tasks being performed in this sections:

- Elevation masking
- Gap filling
- Phase jump detection
- Data arrangement

2.7.1 Elevation masking:

In this section we use computed satellite positions (section 2.5) and Marker (Receiver-Station) position to compute satellite elevation and azimuth. Based on elevation threshold we cut each arc, and only then we move to any further preprocessing step.

2.7.2 Gap filling:

In this section we find in each arc for a possible gap (missing data) and if the gap is within a threshold (defaults to 5 minutes) , then we interpolate the missing data, otherwise we break that arc at the gap and consider start of a new arc after that gap. This continues until the end of the arc being processed, after which we check resulting arc/s for minimum arc length (defaults to 2 hours). Interpolation used here is Lagrange polynomial interpolation of a given degree (minimum is 4, defaults to 6).

2.7.3 Phase jump detection and correction:

We can often find sudden jumps in phase observables which could be due to reasons other than real sudden TEC variations. In order to detect these jumps we use quartiles and IQR . A given arc is divided into n chunks

of length $l \leq 10\,minutes$, and for each chunk we compute quartiles $Q1, Q2, Q3$ and $IQR = Q3 - Q1$. A value is marked as a false jump if $value < Q1 - LB \times IQR$ or $value > Q3 + UB \times IQR$, where LB and UB are given parameters (both default to 1.5). Once detected, these values are replace by an average of values indexed at $\{j-k, \cdots, j-1, j+1, \cdots, j+k\}$, where j is the index of value being replaced, and k is a given parameter (defaults to 3).

2.7.4 Data arrangement:

Finally data is arranged according to time (epochs), the same order found in observation files. This is required so that we have input arrays ready for final solver (see system description).

2.8 vTECeq

For vTECeq we use MODIP, local time, and the zenith angle over IPP (Section 2.6). MODIP is obtained by following relation:

$$\tan(\mu) = \frac{I}{\sqrt{\cos(\varphi)}}$$

being I the true magnetic inclination, and φ the geographic latitude of the receiver-station[19].

For I we compute IGRF-12± model[18], implemented as a separate class requiring respective IGRF coefficients file to compute for I .

These values are used in the polynomial describing vTECeq:

$$vTECeq = C_0 + C_1 x + C_2 x^2 + C_3 y + C_4 x y + C_5 y^2$$

where

$$x = \lambda_{IPP} - \lambda_{station} + 2\pi \frac{\Delta t}{86400}$$

± Use of IGRF (currently 12th generation) is based on input data to be calibrated.

$$y = \mu_{IPP} - \mu_{station}$$

being λ_{IPP} the IPP longitude, $\lambda_{station}$ the station longitude, μ_{IPP} the IPP MODIP, $\mu_{station}$ the station MODIP, and $\Delta t = t_{obs} - t_{mid}$ being t_{obs} the UT observation time in minutes, t_{mid} the UT mid time of the corresponding sampling time interval(see section parameter_samplingtime).

The relation between vTECeq and sTEC is given by following relation:

$$\cos(z') = \frac{vTECeq}{sTEC}$$

being $$\cos(z') = \frac{1}{sec(z')}$$

we have $$sTEC = sec(z')\, vTECeq$$

or

$$sTEC = sec(z') \times (C_0 + C_1 x + C_2 x^2 + C_3 y + C_4 x y + C_5 y^2)$$

Here Coefficients go into Matrix C (unknowns) and the values corresponding to coefficients makeup Matrix A (Section 2.9.2).

2.9 System Description and sparsity

In this section we define the system matrices and vectors which would be operated on by main solver. These include following:

- S Vector
- A Matrix
- B Matrix

2.9.1 Vector S:

All the uncalibrated slant TEC values go into this vector S. Order of the values is the same as it appears in the observation files (arranged by epochs). It is a dense vector whose size depends on number of epochs, number of satellites per epoch, and observation frequency in data files.

2.9.2 Matrix A:

This Matrix stores the values corresponding to unknown coefficients (Section 3.8). Since in this calibration algorithm we fix the set of coefficients for a given time interval (sampling-time, usually 10 minutes), we get structure of Matrix A as a block diagonal. Only non-zero blocks of this matrix are stored as shown in figure 2.3. The dimension of A depends on number of values in Vector S and the number of coefficients in polynomial defining vTECeq.

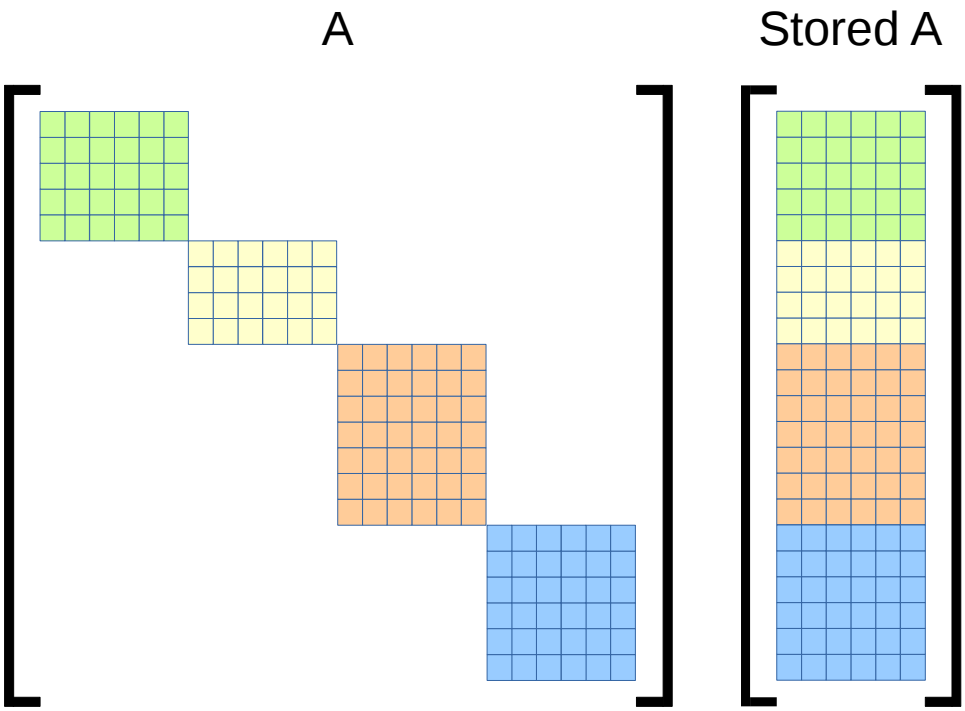

Figure 2.3: Matrix A block structure

2.9.3 B Matrix:

This is a boolean matrix relating a given sTEC value in vector S to a specific arc number. These arc numbers are defined by the preprocessing section (Section 2.7.2). Matrix is defined as follows:

$$B_{ij} = \begin{cases} 1 : S_i \in Arc_j \\ 0 : elsewhere \end{cases}$$

The dimensions of B depend on number of values in vector S and number of arcs formed after preprocessing section.

2.9.4 Sparsity:

As described above matrices A and B are sparse matrices, and in the case of matrix A, we only store non-zero values. We also take advantage of the block structure of A and all operations with A are performed on individual blocks(see section solution).

Chapter 3

Solver

3.1 Intel MKL Library

For the main solver routine we integrate this code with Intel MKL[27], which feature highly optimized, threaded, and vectorized routines to maximize performance. It utilizes the de facto standard C/Fortran APIs, in order to be compatible with BLAS, LAPACK and other math libraries.

3.1.2 List of used MKL routines

Here we present the list of used routines in the whole solver and their description[28].

- cblas_dgemm: Computes a matrix-matrix product
- LAPACKE_dgetrf + LAPACKE_dgetri: To comput LU factorization and then inverse of a matrix, used together
- cblas_dgemv: To compute a matrix-vector product
- cblas_daxpy: For vector-vector or matrix-matrix subtraction

3.2 Compute for Blocks of A:

For this Section of solver we compute for individual blocks of matrix the required computations as described in section 1.10. We loop over individual blocks of matrix A and the corresponding sections of vector S and Matrix B.

3.3 Solution for Arc-Offset:

Once we have the values for S_{AcS} and B_{AcS} from the previous section we move on with the solution of Arc-Offset β, as described in Section 1.10.1.

Afterwards we compute $calibrated\,sTEC = S - B\beta$ as described in Section 1.10.

3.4 Computing Residuals:

Once we are done with Arc-Offset Solution we calculate residuals by

$residuals = (S - AcS) - (B - AcB)\beta$ as described in Section 1.10.

3.5 Solution for vTECeq:

Finally the Solution for vTECeq is computed by going over precomputed c and computing $C = c(calibrated\,sTEC)$ as described in Section 1.10.2.

Chapter 4

Code Structure and Documentation

4.1 Code Structure

The new code for the calibration tool is written from scratch in C++ language, complying to at-least standard C++11[22]. The code is structured in classes using object-oriented techniques. Not everything is modeled around OOP concept during this development project, but rather we take the middle option to structured some section of code in classes and leaving some sections flat as per the compromise between beauty and performance. Following is a series of short descriptions about some sample classes in the code with class diagrams.

4.1.1 Class internalTime:

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 as Integers or by providing a string which would be parse to store time in both formats. The later being more useful for reading observation files, as epochs appear in observation files as strings.

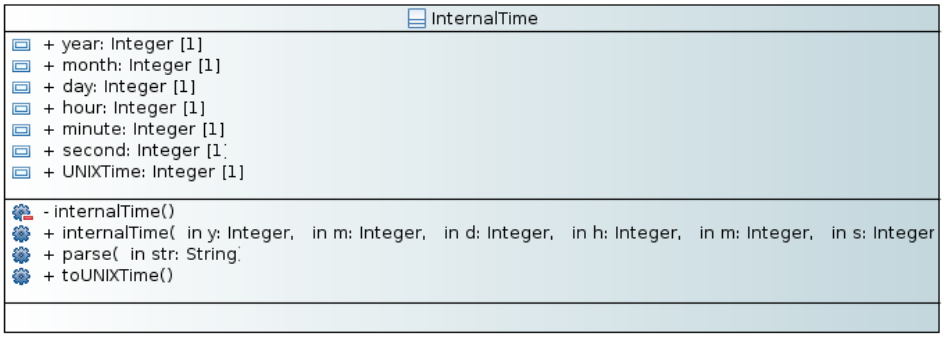

Figure 4.1: Class internalTime

4.1.2 Class triple:

This is the most simple class in the project, being providing an object defining a point using three dimensional coordinates. Instances of this class are used extensively throughout the project.

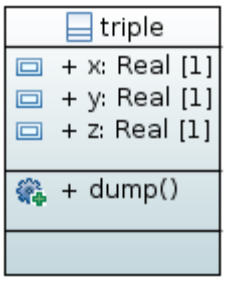

Figure 4.2: Class triple

4.1.3 Class ObsData

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.

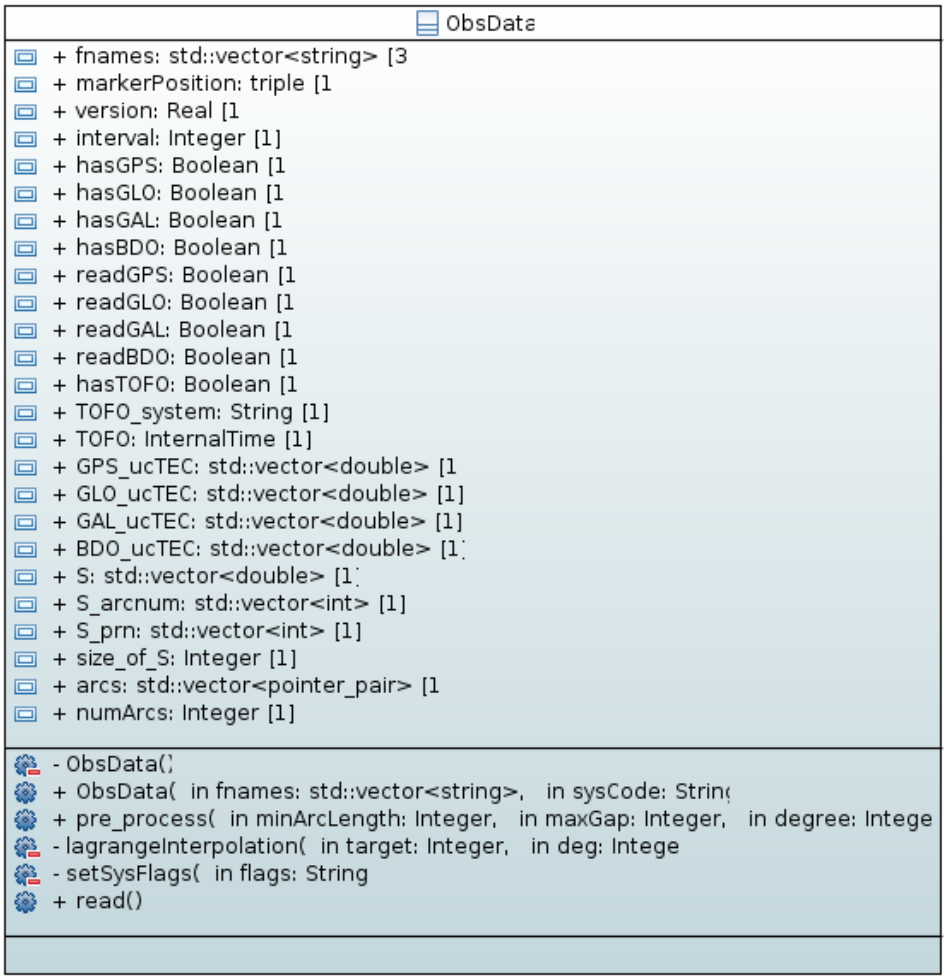

Figure 4.3: Class ObsData

4.1.4 Class ephemerisGEC:

This class defines ephemeris data object based on GPS/Galileo/BeiDou ephemeris record in navigation files. This class serves as the data type for for vectors containing navigation data for GPS/Galileo/BeiDou.

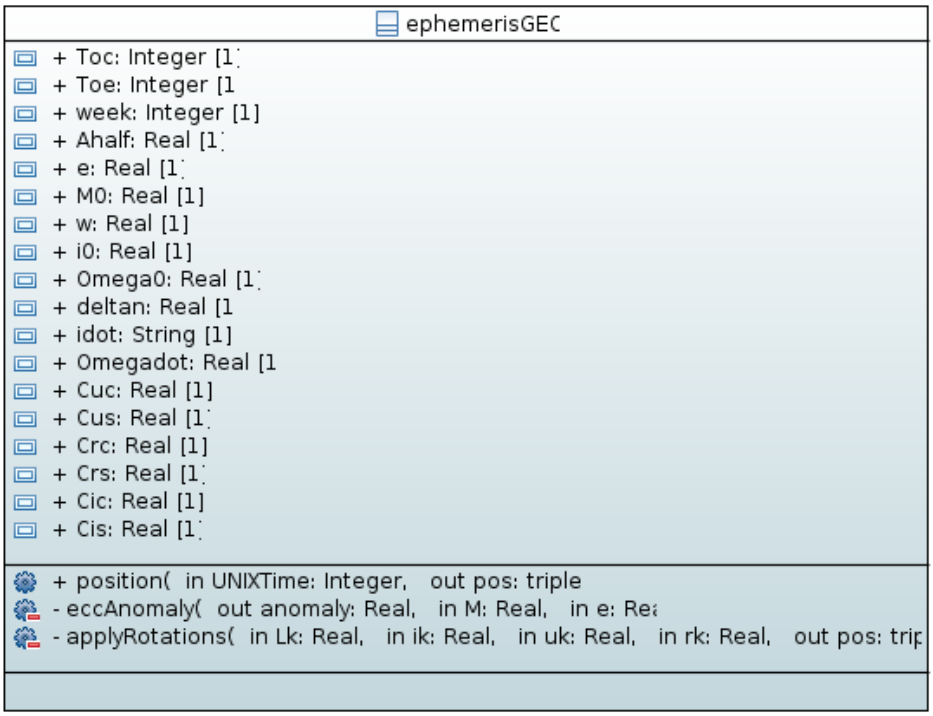

Figure 4.4: Class ephemerisGEC

4.1.5 Class ephemerisR:

This class defines ephemeris data object based on GLONASS ephemeris record in navigation files. This class serves as the data type for for vectors containing navigation data for GLONASS.

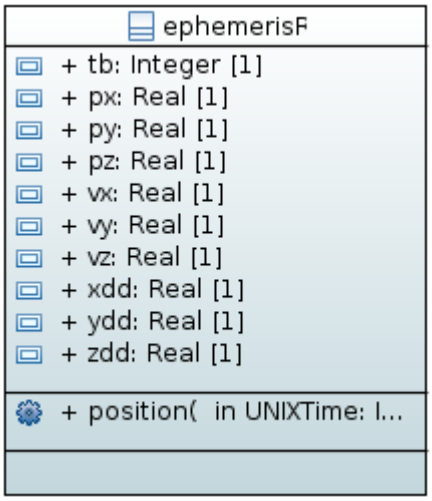

Figure 4.5: Class ephemerisR

4.1.6 Class NavData:

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

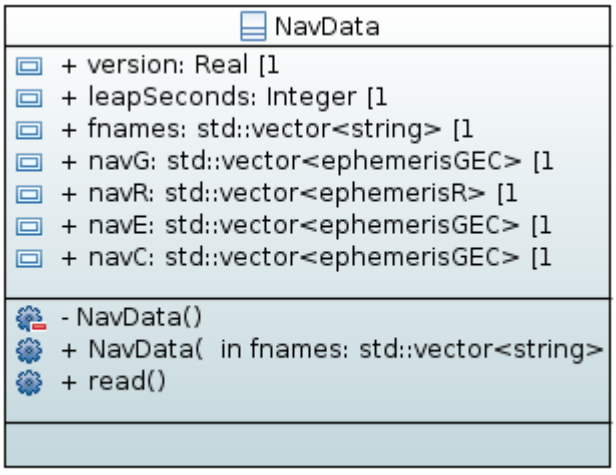

Figure 4.6: Class NavData

4.1.7 Class Dependency:

following is a class dependency diagram for sample classes described above:

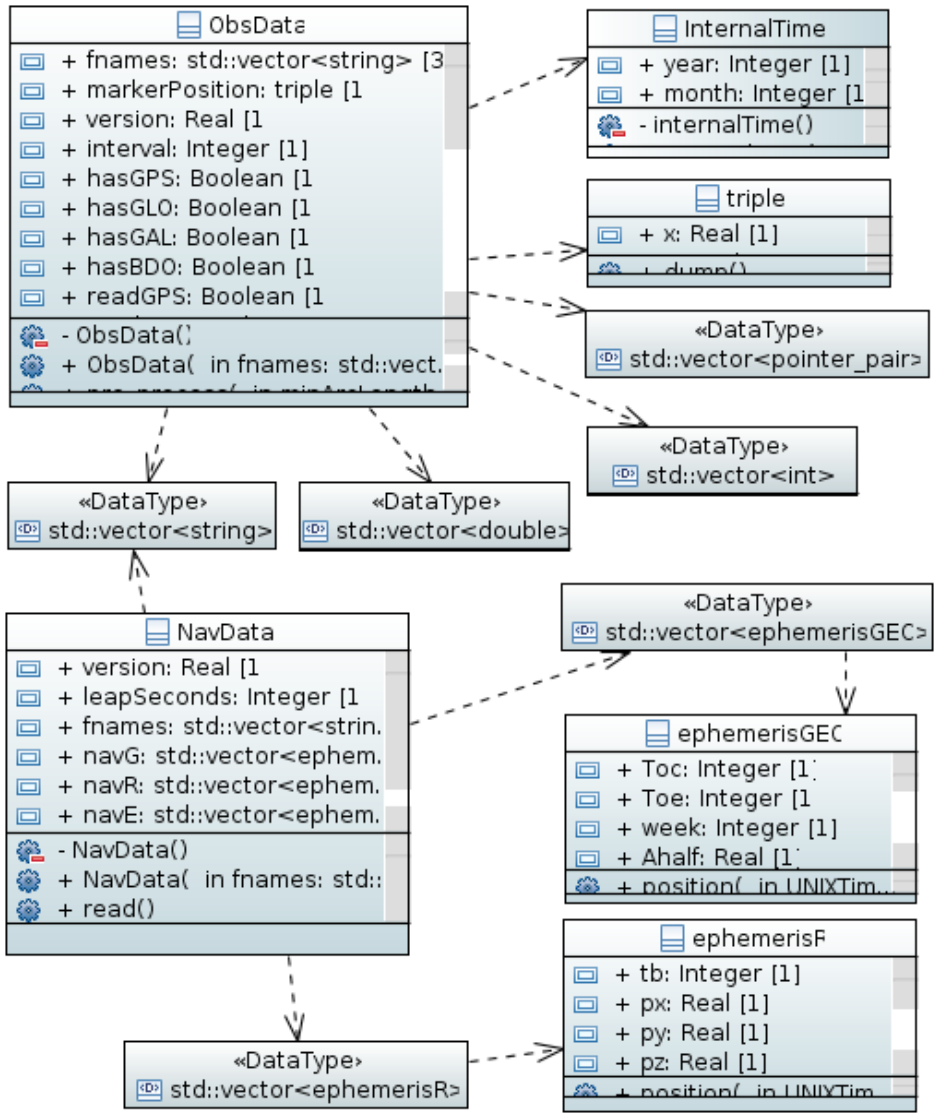

Figure 4.7: Class Dependency

4.2 Documentation using doxygen:

Besides the new code being well structured using classes, the code is also well documented. Documentation is done by means of doxygen. Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D[23]. We can generate documentation as HTML or pdf using this tool. Following are some snaps from HTML version of sample class documentation:

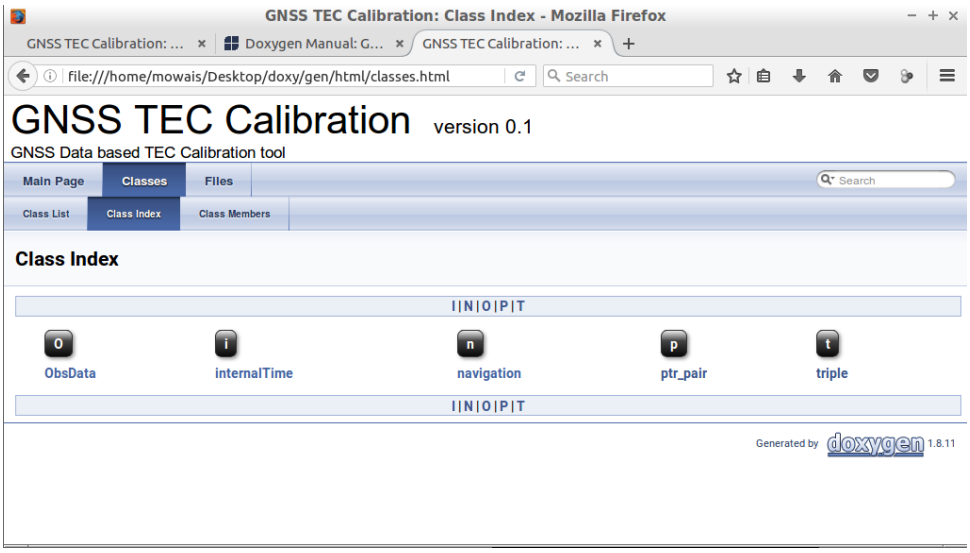

Figure 4.8: Doxygen documentation Classes

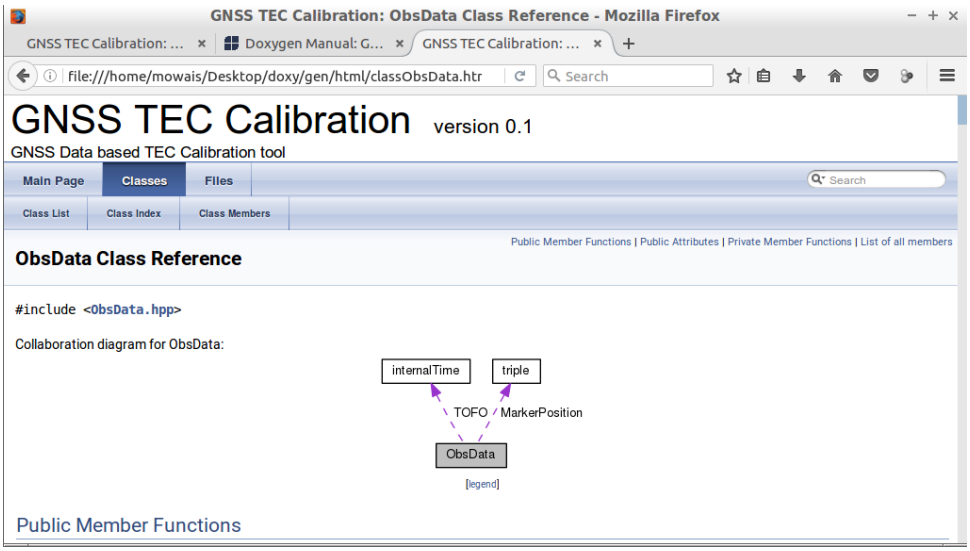

Figure 4.9: Doxygen documentation Class ObsData

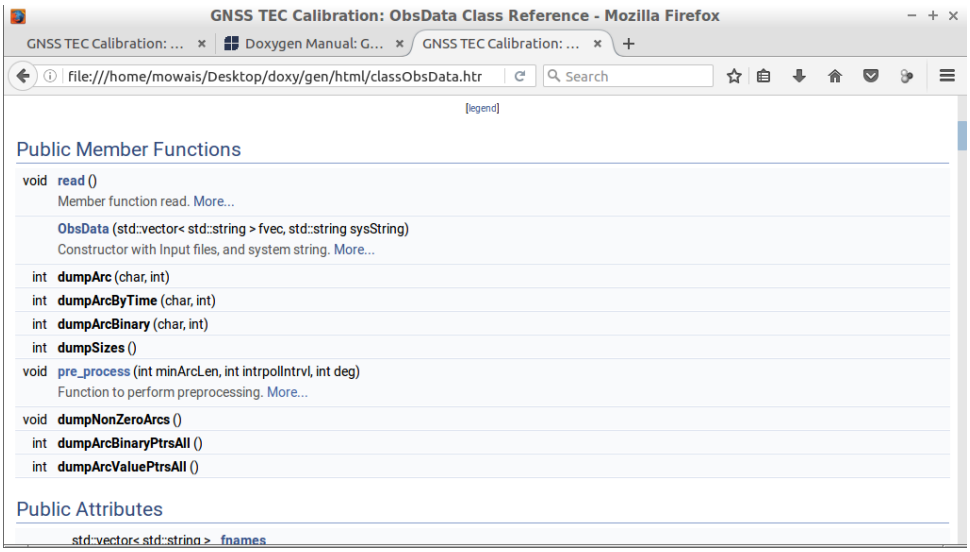

Figure 4.10: Doxygen documentation public member functions

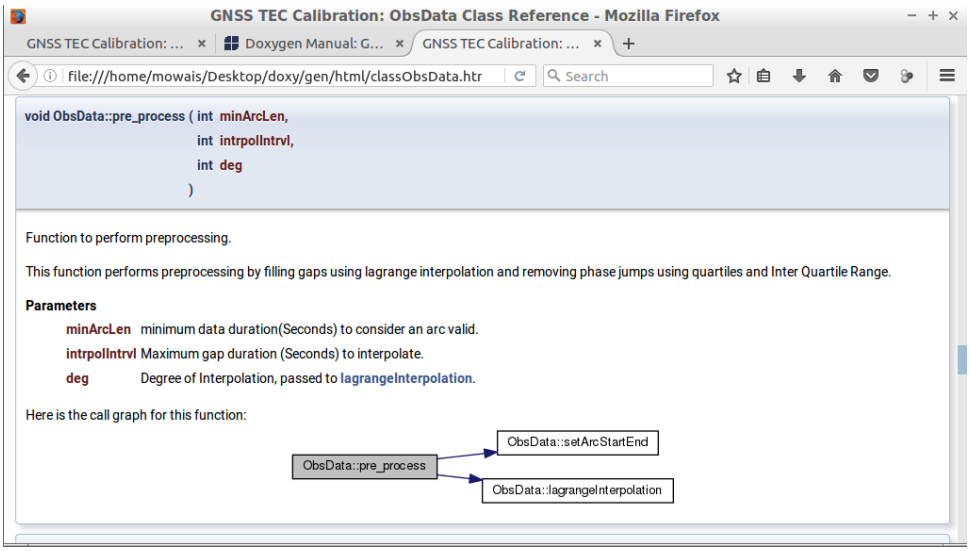

Figure 4.11: Doxygen documentation member function pre_process

Chapter 5

Benchmarking of Code

5.1 Tests on a standalone workstation

In this section we present the benchmarking results on a standalone workstation. We choose a workstation similar to the configuration which is used by T/ICT4D Lab for calibration tool. It is important to note that the old version of the software runs on Windows, therefore we choose a workstation with both Linux and Windows configured and use this machine for benchmarking.

5.1.1 Workstation Configuration:

The workstation used in this benchmark is based on Intel Quad-Core i7 – 3770 with 4 Cores, and 8 Threads with Hyper-threading[24], having base frequency at 3.40 GHz, and up to 3.90 GHz with single core turbo boost[25].

5.1.2 Workstation Test1:

In this test we compare execution time (time to solution) for serial versions with different system sizes (Varying observation interval)

- fsolve is single precision version with serial MKL
- dsolve is double precision version with serial MKL
- Time to solution for fsolve is scaled to 40
- Time to solution for dsolve is scaled to 25

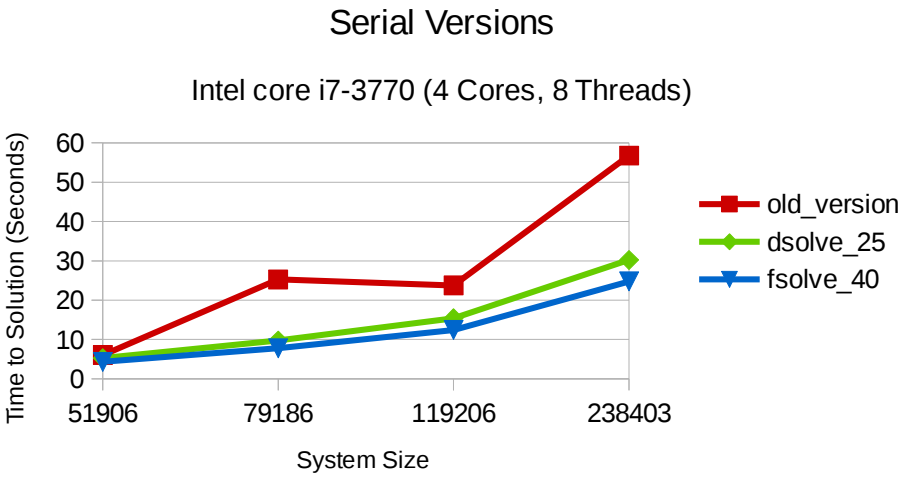

Figure 5.1: Workstation test 1 Serial Versions

5.1.3 Workstation Test2:

- Parallel versions compared with fixed highest possible system size (Interval 15) against number of threads
- fsolve is single precision version with Parallel MKL
- dsolve is double precision version with Parallel MKL

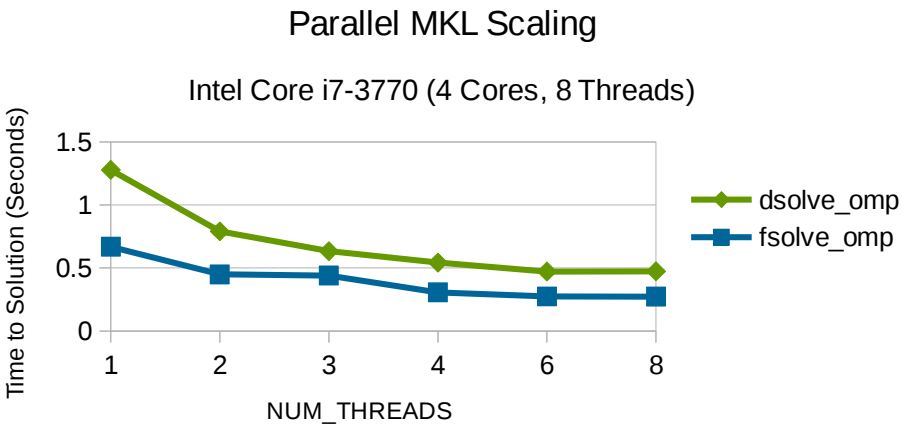

Figure 5.2: Workstation test 2 Parallel MKL Scaling

5.2 Tests on a Compute node from Ulysses Cluster:

This section discusses benchmarks ran on a Compute node from Ulysses Cluster being operated by SISSA[26].

5.2.1 Compute node Specifications:

A single compute node on Ulysses cluster is based on dual Intel Xeon CPU E5-2680 v2 base frequency at 2.80GHz. This is a 10 core processor, therefore we have 20 cores in total on two processor sockets. Following is the NUMA processor affinity policy used for this test.

5.2.2 NUMACTL Policy:

NUMACTL physcpubind:

• NUM_Threads 1 → CPUs (0)

• NUM_Threads 2 → CPUs (0,10)

• NUM_Threads 4 → CPUs (0,1,10,11)

• NUM_Threads 8 → CPUs (0,1,2,3,10,11,12,13)

• NUM_Threads 16 → CPUs (0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17)

5.2.3 Compute node Test 1:

- following graph shows scaling on single Ulysses node using NUMA policy mentioned above
- It describes scaling of whole solver code (including times for all sections of the solver)
- Notice that this scales up to 8 cores
- dsolve_omp is double precision version with Parallel MKL

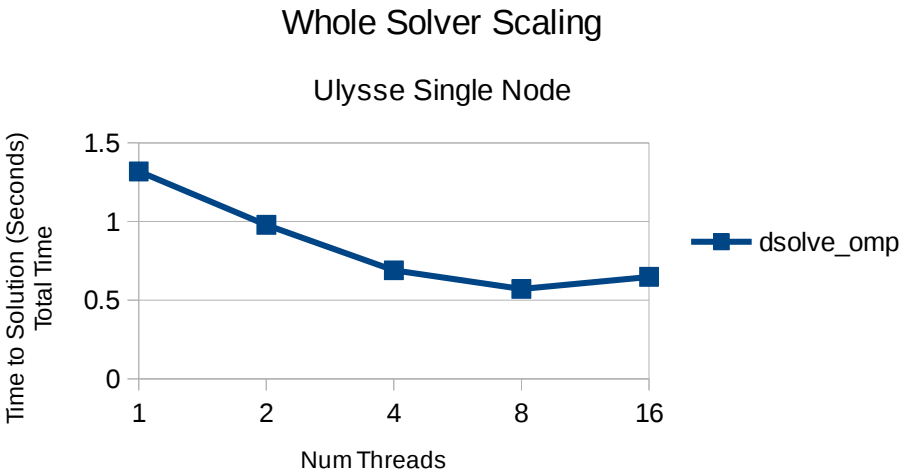

Figure 5.3: Cluster Compute Node test 1 Whole Solver Scaling

5.2.4 Compute node Test 2:

- following graph shows scaling on single Ulysses node using NUMA policy mentioned above
- It describes scaling of only Arc-Offset solution (including only times for Arc-Offset)
- This Section of code was the most time consuming in the old version (Section 5.2.5)
- Notice that this scales up to 16 cores
- dsolve_omp is double precision version with Parallel MKL

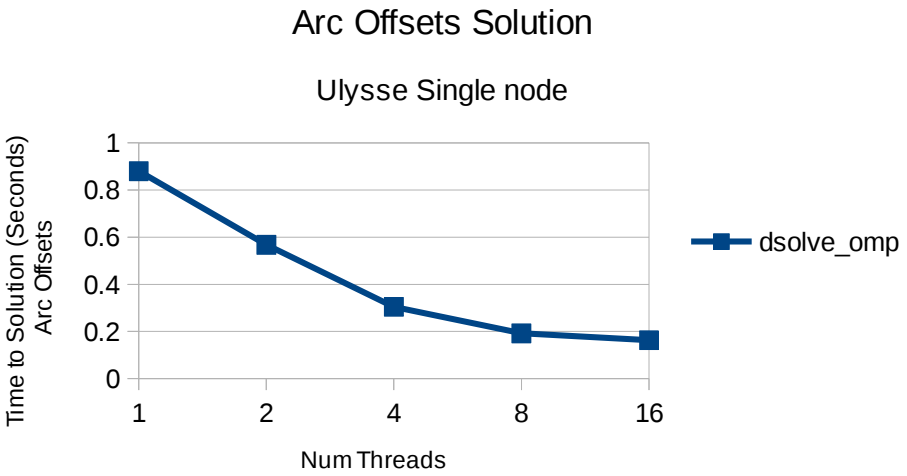

Figure 5.4: Cluster Compute Node test 2 Arc-Offset Solution

5.2.4 Compute node Test 2:

- following graph shows scaling on single Ulysses node using NUMA policy mentioned above
- It describes scaling of only A_Blocks solution (including only times for A_Blocks section)
- This Section of code was the 2nd most time consuming in the old version (Section 5.2.5)
- Notice that this doesn't scale much, and after 8 cores it blows timing.
- Affect of this section could be seen in whole solver timing (Section 5.2.3)
- dsolve_omp is double precision version with Parallel MKL

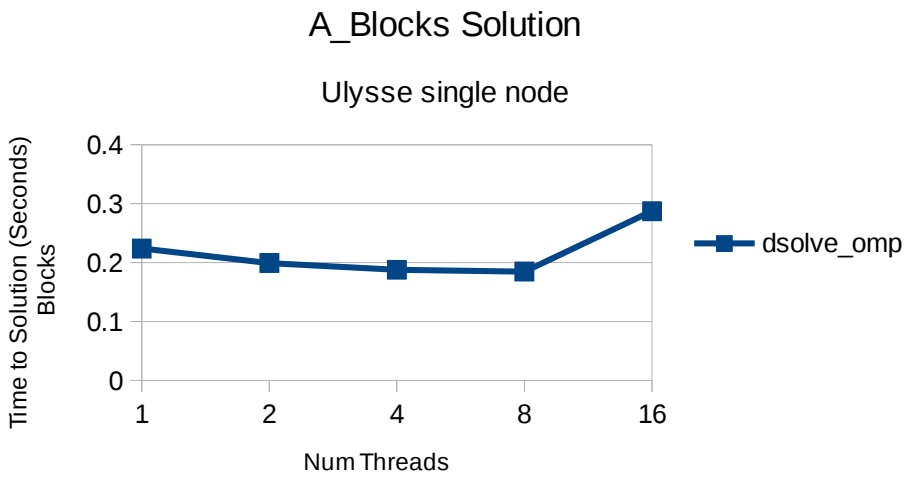

Figure 5.5: Cluster Compute Node test 2 A_Blocks Solution

5.2.5 Old Code Timing:

- This graph shows timings of different solver sections in the old version
- T_blocks is timing for (A_Blocks) section of the code
- T_ArcOffsets is timing for (Arc-Offset) section of the code
- T_Residuals and T_vTECeq have negligible share in time to solution

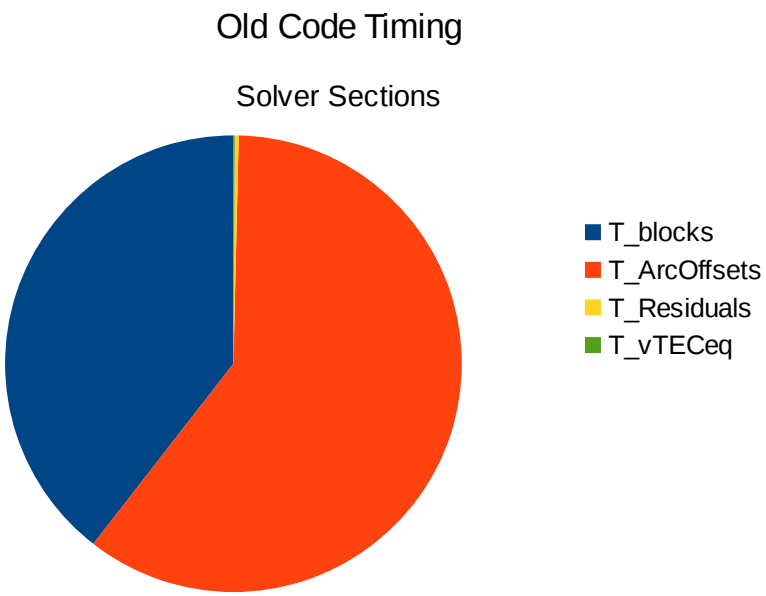

Figure 5.5: Old Code Timing

Chapter 6

Conclusion

As prominent by the benchmarking results shown in the previous chapter, we improve the time to solution in new serial version of the code by a factor of 45, considering the double precision new serial version (Section 5.1.2). Furthermore we improve the time to solution by a factor of more than 100, considering old serial version and multi-threaded new version on the same hardware (Section 5.1.3). The code is now well structured, flexible, and maintainable. The new implementation now makes the code portable to both Linux and Windows, or to any other operating system. The code now includes interfaces to standard LAPACK libraries which allows to drastically improve performance when linked with high-performance libraries such as MKL. The code also provides in-depth and structured documentation with a possibility of easy update of the documentation as this version improves in future. Furthermore the new code and the work of restructuring greatly enhance the collaborative development, which is fundamental in modern science.

Although we improve the code by an impressive factor, still we do not deal with the sparse nature of matrix B (Section 2.9.4). This could be a future room for more improvement. Furthermore we don't explore other parallel platforms like accelerators for this problem, it would be interesting to see if different sections of the code ported to other platforms can provide better scaling and performance or it may be better to go for heterogeneous CPU/Accelerator code for the optimal results.

APPENDIX A

Acronyms

RINEX	Receiver Independent Exchange
GNSS	Geographical Navigation Satellite System
GPS	Global Positioning System
GLONASS	Globalnaya Navigatsionnaya Sputnikovaya Sistema (Russian: Global Navigation Satellite System)
QZSS	Quasi-Zenith Satellite System
SBAS	Satellite-Based Augmentation System
IGS	International GNSS Service
GPST	GPS Time
GLONASST	GLONASS Time
GST	Galileo System Time
BDT	BeiDou Time
UT	Universal Time
UTC	Universal Time Coordinated
TAI	Temps Atomique International (International Atomic time)
IERS	International Earth Rotation and Reference systems Service
WGS	World Geodetic System
TRF	Terrestrial Reference Frame
RKM	Runge-Kutta Method
IPP	Ionospheric Pierce Point

MODIP	Modified Dip Latitude
TEC	Total Electron Content
vTECeq	Vertical TEC Equivalent
sTEC	Slant TEC
IGRF	International Geomagnetic Reference Field
IMSL	International Mathematics and Statistics Library
SV	Satellite Vehicle
FLOP	Floating Point Operations
ICTP	International Center for Theoretical Physics
SISSA	International School for Advanced Studies
MKL	Math Kernel Library
API	Application Programming Interface
T/ICT4D	Telecommunications/ICT for Development Laboratory
LAPACK	Linear Algebra Package

APPENDIX B

Bibliography

[1] IGS Formats
http://kb.igs.org/hc/en-us/articles/201096516-IGS-Formats

[2] RINEX Version 2.10
ftp://igs.org/pub/data/format/rinex210.txt

[3] RINEX Version 2.11
ftp://igs.org/pub/data/format/rinex211.txt

[4] RINEX Version 3.01
ftp://igs.org/pub/data/format/rinex301.pdf

[5] RINEX Version 3.02
ftp://igs.org/pub/data/format/rinex302.pdf

[6] RINEX Version 3.03
ftp://igs.org/pub/data/format/rinex303.pdf

[7] GPS Interface Specification, Navstar GPS Space Segment/Navigation User Segment Interfaces (IS-GPS-200G), GPS Directorate, Revision G, 21 September 2011

[8] GLONASS-ICD Interface Control Document Version 5.1 2008, English Version
http://kb.unavco.org/kb/assets/727/ikd51en.pdf

[9] Galileo Open Service Signal In Space Interface Control Document (OS SIS ICD), GSA, Issue 1.1, September 2010

[10] BeiDou Signal In Space Interface Control Document, Open Service Signal (BDS-SIS-ICD-2.0), China Satellite Navigation Office, Version 2.0, December 2013

[11] “Atomic Time” navipedia article by J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain. (http://www.navipedia.net/index.php/Atomic_Time)

[12] International Earth Rotation Service (https://www.iers.org/IERS/EN/Home/home_node.html)

[13] Global Positioning System Standard Positioning Service Signal Specification
http://www.navcen.uscg.gov/pubs/gps/sigspec/gpssps1.pdf

[14] “GPS and Galileo Satellite Coordinates Computation” navipedia article by J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain.
http://www.navipedia.net/index.php/GPS_and_Galileo_Satellite_Coordinates_Computation

[15] “Reference Frames in GNSS” navipedia article by J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain.
http://www.navipedia.net/index.php/Reference_Frames_in_GNSS

[16] “Transformation between Terrestrial Frames” navipedia article by J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain.
http://www.navipedia.net/index.php/Transformation_between_Terrestrial_Frames

[17] “GLONASS Satellite Coordinates Computation” navipedia article by J. Sanz Subirana, J.M. Juan Zornoza and M. Hernández-Pajares, Technical University of Catalonia, Spain.
http://www.navipedia.net/index.php/GLONASS_Satellite_Coordinates_Computation

[18] International Geomagnetic Reference Field: the 12th generation, Erwan Thébault, Christopher C Finlay, Ciarán D Beggan, Patrick Alken, Julien Aubert, Olivier Barrois, Francois Bertrand, Tatiana Bondar, Axel Boness, Laura Brocco, Elisabeth Canet, Aude Chambodut, Arnaud Chulliat, Pierdavide Coïsson, Francois Civet, Aimin Du, Alexandre Fournier, Isabelle Fratter, Nicolas Gillet, Brian Hamilton, Mohamed Hamoudi, Gauthier Hulot, Thomas Jager, Monika Korte, Weijia Kuang, Xavier Lalanne, Benoit Langlais, Jean-Michel Léger, Vincent Lesur, Frank J Lowes et al. Earth, Planets and Space 2015, 67:79 (27 May 2015)

[19] Rawer, K., B. Landmark (Ed.), Meteorological and Astronomical Influences on Radio Wave Propagation, Pergamon Press, Oxford (1963), pp. 221–250

[20] Ciraolo, L., Azpilicueta, F., Brunini, C. et al. “Calibration errors on experimental slant total electron content (TEC) determined with GPS” Journal of Geodesy (2007), Volume 81, Issue 2, pp 111–120.

[21] IMSL FORTTRAN Numerical Libraries
http://www.roguewave.com/products-services/imsl-numerical-libraries/fortran-libraries

[22] ISO C++11 Standard https://isocpp.org/std/standing-documents

[23] doxygen a documentation generator by Dimitri van Heesch
http://www.stack.nl/~dimitri/doxygen/

[24] Intel Hyper-Threading Technology (Intel HT Technology)
http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html

[25] Intel Turbo Boost Technology
http://www.intel.it/content/www/it/it/architecture-and-technology/turbo-boost/turbo-boost-technology.html

[26] Ulysses High Performance Computing Cluster – SISSA
https://www.itcs.sissa.it/services/computing/hpc

[27] Intel Math Kernel Library
https://software.intel.com/en-us/intel-mkl

[28] Intel Math Kernel Library Documentation
https://software.intel.com/en-us/mkl-reference-manual-for-c

[29] Calibration Fortran code implemented by Katy Alazo-Cuartas
http://t-ict4d.ictp.it/nequick2/gnss-tec-calibration

[30] GNSS TEC Calibration online tool - ICTP
http://t-ict4d.ictp.it/nequick2/gps-tec-calibration-online