

DANIEL ESTÉVEZ <

HTTPS://DESTEVEZ.NET/>

*Scientific & Technical Amateur Radio — Home of
EA4GPZ / MOHXM*

Simulating delta-range observations in GMAT

During the **DSLWP-B < <https://destevez.net/tags/dslwp>**
(Longjiang-2 <
https://en.wikipedia.org/wiki/Chang%27e_4#Longjiang_microsatellites
>) mission, we made a number of **VLBI observations**
< <https://destevez.net/2019/05/results-of-dslwp-b-amateur-vlbi-experiment-on-2018-11-21/>
> of the spacecraft's UHF signal by performing GPS-synchronized recordings at Dwingeloo (The Netherlands), Shahe and Harbin (China), and Wakayama (Japan). The basic measurement for these observations is the time difference of arrival (TDOA), which measures the differences between the time that it takes the spacecraft's signal to arrive to each of the groundstations. This can be interpreted in terms of

the difference of distances between the spacecraft and each groundstation, so this measurement is also called delta-range.

One very interesting practical application of the VLBI observations is to perform orbit determination. The delta-range measurements can be used to constrain and determine the state vector of the spacecraft. This would give us an autonomous means of tracking Amateur deep-space satellites, without relying on ranging by a professional deep-space network. Even though the measurements we made showed good agreement with the ephemerides computed by the Chinese deep-space network, during the mission we never ran orbit determination with the VLBI observations, mainly due to the lack of appropriate software.

While GMAT has good support for orbit determination, it doesn't support delta-range measurements. Its basic orbit determination data type is two-way round-trip time between a groundstation (or two) and the satellite, as shown in the **orbit determination tutorial** <

http://gmat.sourceforge.net/doc/R2018a/html/Orbit_Estimation_using_DSN_Range_and_Doppler_Data.html > .

I have started to modify GMAT in the **gmat-dswlp** < **<https://github.com/daniestevez/gmat-dslwp>** > Github repository to implement the support for this kind of VLBI

observations. As a first step, I am now able to create and simulate delta-range observations.

Mathematical setting

Here I describe the equations that need to be computed to simulate a delta-range observation. In the observation there are two participating groundstations, which I call the *reference* station and the *other* station. These stations give rise to two measurement legs. Measurements are time-tagged with respect to the reference station. Let x_R , x_O and x_S denote the positions of the reference station, the other station and the spacecraft in some inertial reference frame. In GMAT this frame will be centred on the Earth if the spacecraft is Earth-orbiting, and the solar system barycentre if not (groundstations are assumed to be located near the surface of the Earth, so in this way a common inertial frame for all the participants is chosen).

For the reference leg, if t_R is the time at which the spacecraft's signal is received in the reference groundstation, then the reference range at t_R is

$$r_R = \|x_R(t_R) - x_S(t_R - \tau_R)\| + c\delta_R.$$

Here τ_R denotes the time of flight or light-time of the signal, so that the signal was transmitted by the spacecraft at time $t_R - \tau_R$,

and δ_R denotes additional corrections such as atmospheric and relativistic corrections. The light-time is computed as a solution of the equation

$$c\tau_R = \|\mathbf{x}_R(t_R) - \mathbf{x}_S(t_R - \tau_R)\|,$$

which can be solved iteratively. Note that this equation doesn't include the additional corrections δ_R . This is an approximation that is usual to make. It is also possible to include all or some of the corrections that form part of δ_R . The fact that to compute r_R the satellite position is not evaluated at time t_R but rather at time $t_R - \tau_R$ is called the light-time correction.

For the other leg, the time of transmission is taken to coincide with the time of transmission of the reference leg. Therefore, the other range at measurement time t_R is

$$r_O = \|\mathbf{x}_O(t_R - \tau_R + \tau_O) - \mathbf{x}_S(t_R - \tau_R)\| + c\delta_O,$$

where now the light-time τ_O is a solution of

$$c\tau_O = \|\mathbf{x}_O(t_R - \tau_R + \tau_O) - \mathbf{x}_S(t_R - \tau_R)\|,$$

and δ_O denotes the additional corrections for this leg.

The delta-range at time t_R is the difference of ranges, $r_R - r_O$. Note that the roles of the groundstations are not quite

interchangeable, since the reception time at the other station is not exactly t_R in general.

Implementation in GMAT

GMAT already has all the basic building blocks that are needed to calculate delta ranges, since it has code that can be used to compute ranges, including light-time corrections, in a very flexible way. This code is contained in the Estimation Plugin, and its C++ object-oriented architecture makes it easy to add new measurements to extend the functionality.

I have based my work on the **GMAT-R2019aBeta1** < <https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2019a-Beta/>> release, which has important changes to the Estimation Plugin in comparison to GMAT-R2018a, such as the addition of measurements for orbit estimation with angular quantities (azimuth, elevation, right ascension and declination).

I haven't been able to find the code for GMAT-R2019aBeta1 in **GMAT's git repository** < <https://sourceforge.net/p/gmat/git/ci/GMAT-R2018a/tree/>> , so I've set up a new repository in Github to work with the code. This repository, called **gmat-dslwp** < <https://github.com/daniestevez/gmat-dslwp>> , contains submodules with the binary release and source code release of

GMAT-R2019aBeta1. By having the binary release, it is only necessary to rebuild the Estimation plugin and link it against this binary release. Also, this makes it possible to ship the binary release with the modified Estimation Plugin to users that don't necessarily want to build the code.

In the Estimation Plugin, different measurements are implemented by classes derived from the **TrackingDataAdapter** < <https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/TrackingDataAdapter.hpp>> class. The range calculations are implemented by a class called **GNRRangeAdapter** < <https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/GNRRangeAdapter.hpp>> , that derives from TrackingDataAdapter through RangeAdapterKm. GNRRangeAdapter uses under the hood the **CalculateMeasurement()** < <https://github.com/daniestevez/gmat-dslwp-source/blob/387a2c148e82bbbd92796d21c9d624d40b834ba8/GMAT->

[R2019aBeta1/plugins/EstimationPlugin/src/base/measurementmodel/MeasureModel.cpp#L1164](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measurementmodel/MeasureModel.cpp#L1164)> method of the [MeasureModel](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measurementmodel/MeasureModel.hpp) < <https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measurementmodel/MeasureModel.hpp>> class to do all the range and light-time calculations.

Actually, adapter classes rely on [MeasureModels](https://github.com/daniestevez/gmat-dslwp-source/tree/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme) < <https://github.com/daniestevez/gmat-dslwp-source/tree/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme>[ntmodel](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme)> to do all the heavy lifting calculation, and there are currently only two MeasureModels: the MeasureModel class, which does computations in terms of ranges between spacecrafts and/or groundstations, and the [GPSPointMeasureModel](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme) < <https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme>[ntmodel/GPSPointMeasureModel.hpp](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/measureme)> , which is a rather different approach that uses GPS ECEF state vectors for orbit determination.

To implement delta-range measurements, I have added a new class [DeltaRangeAdapter](#) <

[https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)

[R2019aBeta1/plugins/EstimationPlugin/src/base/adapte](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)
[r/DeltaRangeAdapter.hpp](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)> which is inspired on

GNRangeAdapter and also takes some ideas from

[TDRSDopplerAdapter](#) <

[https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)

[R2019aBeta1/plugins/EstimationPlugin/src/base/adapte](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)
[r/TDRSDopplerAdapter.hpp](https://github.com/daniestevez/gmat-dslwp-source/blob/upstream/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)> about how to handle different

propagation paths (or legs) in the same TrackingDataAdapter class.

The [CalculateMeasurement\(\)](#) <

[https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)

[R2019aBeta1/plugins/EstimationPlugin/src/base/adapte](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)
[r/DeltaRangeAdapter.cpp#L469](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)> method of

DeltaRangeAdapter follows the mathematical description

outlined above, calling

`GNRangeAdapter::CalculateMeasurement()` to compute the ranges r_R and r_O using the correct timestamps.

The way to define the measurement strand in the GMAT script is as `{reference_groundstation, spacecraft,`

`other_groundstation}`. There is some code <

<https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT->

[R2019aBeta1/plugins/EstimationPlugin/src/base/trackingfile/TrackingFileSet.cpp#L2735](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/trackingfile/TrackingFileSet.cpp#L2735)> in the `TrackingFileSet` class

that takes care of constructing the reference and other legs accordingly.

Error modelling for the delta-range measurements is also new. In the other kinds of measurements supported by GMAT, the error model is associated with the receiving groundstation. Here it makes more sense to have an error model which is associated with the baseline (i.e., the pair of participating groundstations). To accomplish this, in the script editor an `ErrorModel` with `Type = 'DeltaRange'` can be created to represent the error model of a baseline. This `ErrorModel` should be assigned to exactly the two stations participating in the baseline. The `DeltaRangeAdapter`

class includes new implementations of the methods

[ComputeMeasurementBias\(\)](#) <

[https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)

[R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/DeltaRangeAdapter.cpp#L667](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/DeltaRangeAdapter.cpp#L667)> and

[ComputeMeasurementNoiseSigma\(\)](#) <

[https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapte)

[R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/DeltaRangeAdapter.cpp#L783](https://github.com/daniestevez/gmat-dslwp-source/blob/bb160973960c906bbb0b472a1f1d037ae07bf4d6/GMAT-R2019aBeta1/plugins/EstimationPlugin/src/base/adapter/DeltaRangeAdapter.cpp#L783)> which are similar to those in

TrackingDataAdapter but search for an `ErrorModel` which is assigned to the two groundstations in the path.

Test GMAT script

There is a [GMAT script](#) <

<https://github.com/daniestevez/gmat-dslwp/blob/master/scripts/test-deltarange.script>> that can be

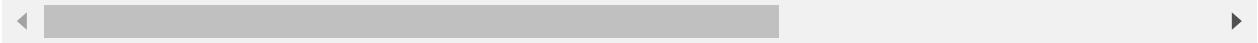
used to test the `DeltaRangeAdapter` class with a GMAT

simulation. The script includes real DSLWP-B ephemeris from 28 Jun 2019 (the first I could find in my old DSLWP-B script files,

actually), and the groundstation definitions for PI9CAM (Dwingeloo) and Shahe. A single delta-range measurement between PI9CAM (as the reference station) and Shahe (as the other station) is simulated. The output is saved to the file DSLWP-B.gmd in GMAT's output/ folder. This file contains the following:

```
% GMAT Internal Measurement Data File
```

```
28662.7087615740740740771674 DeltaRange -1 22222 1111
```



The last column is the simulated delta-range in km.

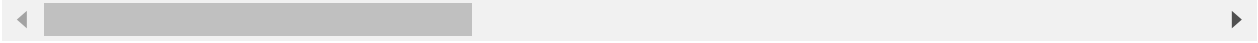
The simulation contains all the corrections such as light-time, relativistic effects, ET-TAI, and tropospheric corrections. It doesn't include the ionospheric correction because the IRI2007 model data bundled with GMAT-R2019aBeta1 only covers until 2018-03-22. I'm looking for a way to update this file.

Worked example

The **[debug branch < https://github.com/daniestevez/gmat-dslwp-source/tree/debug >](https://github.com/daniestevez/gmat-dslwp-source/tree/debug)** of **[gmat-dslwp-source < https://github.com/daniestevez/gmat-dslwp-source >](https://github.com/daniestevez/gmat-dslwp-source)** enables additional debug output for the Estimation Plugin that can be used with the GMAT script described above to check manually that all the intermediate calculations are correct.

The full debug output can be seen in [this gist < https://gist.github.com/daniestevez/7f6c77293b2adb7761ddb4abdfa5710>](#) . Here I will go through the more relevant calculations.


The measurement needs to be computed at the time t_R given 28 Jun 2019 05:00:00 UTC, which corresponds to 2019-06-28 05:00:37 TAI, which gives a Julian date of 2458662.708761574235. This is equivalent to GMAT's MJD 28662.708761574235. Now, internally GMAT uses the time A.1 for its calculations, which is defined by $A.1 = TAI + 0.0343817$ seconds. This gives an A.1 MJD of 28662.70876197217.

In the debug output one of the first things we can see is the line
Range, relativity correction, and ET-TAI correction c


The A.1 MJD timestamp given there differs from the measurement timestamp by only 13.8 us (and the error is most likely caused by me doing the calculations carelessly with [Astropy's Time < https://docs.astropy.org/en/stable/time/>](#) , since the error when writing a Julian date in double precision is on the order order of 21 us).

So we see that GMAT is computing the reference leg for the appropriate reception time t_R . Then we see

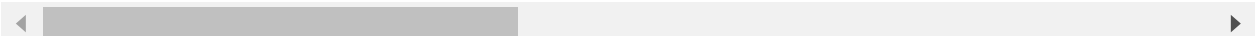
```
Compute Range Vector before light time correction for
...
DeltaT for light travel over distance 387661.907 km =
```



Here GMAT computes the positions of DSLWP-B (in the LunaMJ2000 system) and PI9CAM (in the EarthMJ2000 system) at time t_R , and relates them through the positions of the Moon and Earth in the SSBMJ2000 system (which is centred on the solar system's barycentre). The distance between these positions is 387661.907 km, which corresponds to 1.293101 s of light-time.

Now GMAT goes ahead and uses 1.293101 s to set the first approximation of τ_R , computing the positions of DSLWP-B and the Moon at $t_R - \tau_R$. This yields

```
Light Time range = 387629.161154257366 km
Relativity correction = 0.007573161548 km
==> dEpoch = -1.293100933975e+00 second, dR = 387629
```



As we can see, by updating the transmission time in -1.293101 s, the distance between the positions has now changed to 387629.16872742 km. This is a change of -32.7458 km in the distance, which gives a change of -109.230 us in the light-time.

The relativistic correction works out to 7.57 additional metres of range, which gives an additional light-time of 25.26 ns, so the total change is -109.204 us of light-time. The approximation for τ_R is updated with this change to give 1.29299172939 s.

The algorithm continues a couple of iterations more, updating the value of τ_R until obtaining

```
Light Time range = 387629.163919469516 km
Relativity correction = 0.007573161601 km
==> dEpoch = -1.292991738614e+00 second, dR = 387629
```

which is deemed a good enough approximation for the light-time τ_R , as the last update was already on the order of 1e-12 seconds.

This final value is taken as the correct value for τ_R and the spacecraft and Moon's positions are recomputed to obtain the final solution for this leg.

Compute Range Vector after light time correction for ...

Summary for signal leg from DSLWP_B to PI9CAM:

- . Geometric range = 387661.907438495546 km
- . Light time solution range = 387629.163919478131 km
- . Relativity correction = 0.007573161601 km
- . ET-TAI correction = 32.978179363250 km
- . Feasibility = true

The geometric range of 387661.907 km was the first distance ignoring the light-time correction, while the distance taking into account the light-time correction is 387629.164 km. The difference of 32.744 km shows the importance of taking into account the light-time correction.

Note the slight difference of 8.6 micrometers between the light-time solution obtained in this step and the light-time solution obtained in the previous step. I think this happens because for the calculation of the light-time correction the positions of DSLWP-B and the Moon are updated by propagating in a straight line, using their velocity vectors at t_R . However, here the positions are computed again using the full numerical propagator. However, I haven't checked if the source code actually does this.

The relativity correction is 7.6 m, so it is a small correction, and the ET-TAI correction, which I haven't understood completely yet, is rather large. The feasibility flag indicates that the measurement path is not obstructed.

After this step, we have the calculation of atmospheric corrections. First, the frequencies of the signals are worked out. The arrival frequency shown below should be ignored, as it is only valid when the leg starts from a spacecraft transponder, to which

a signal has arrived. In this case, the spacecraft only has a transmit frequency. Range rate as seen by the groundstation is approximated as the projection of the velocity vector onto the line-of-sight vector in order to compute the Doppler shifted frequency received in the groundstation. The atmospheric corrections are computed for this receive frequency (the frequency doesn't matter for the troposphere correction, but it does matter for the ionosphere correction which we are not computing here).

```

+++++
+++ Signal Frequency calculation for leg from DSLWP_
+++++
. Arrival frequency : -1.000000000000e+00 Mhz
. Transmit frequency : 4.354000000000e+02 Mhz
. Doppler shift frequency: 4.353999891583e+02 Mhz

+++++
+++ Media corrections calculation for leg from DSLWF
+++++
.Frequency : 4.353999891583e+02 Mhz
.Troposphere range correction : 0.004284564790 m
.Troposphere elevation correction : 0.000471377966 ra

```

After this low-level calculations, the GNRangeAdapter finishes its work of computing r_R and prints out a summary:


```

=====
==== GNRangeAdapter (DSNsimData_{DSLWP_B,PI9CAM}_Rang
=====
. Path : DSLWP_B, PI9CAM,
. Measurement epoch : 28662.708761972011
. Measurement type : <DeltaRange>
. C-value w/o noise and bias : 387662.153956567810 krr
. Noise adding option : true
. Bias adding option : true
. C-value with noise and bias : 387662.153956567810 k
. Measurement epoch A1Mjd : 28662.708761972011
. Transmit frequency at receive epoch : 4.354000000000e
. Transmit frequency at transmit epoch : 4.354000000000e
. Measurement is feasible
. Feasibility reason : N
. Elevation angle : 34.728773536118 degree
. Covariance matrix : <0x55bf39780460>
. Covariance matrix size = 1
[ 1.000000e+00 ]

```

Note that noise and bias haven't been added to this range measurement, despite what the printout says. They will be added by `DeltaRangeAdapter` in a final step.

Now it is time for the computation of r_O , the range of the other leg. The interesting thing is that this is done somehow in reverse. The transmit time at the spacecraft is fixed to coincide with the transmit time $t_R - \tau_R$ computed for the reference leg. The

receive time is unknown and is updated by iterating an approximation of the light-time. This starts as

```

+++++ Range, relativity correction, and ET-TAI correct
+++++ Compute Range Vector before light time correction for
...
DeltaT for light travel over distance 389263.327 km =
Starting: dEpoch = 0.000000000000e+00 second, dR = 38

```

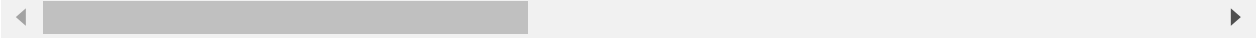
Note that the time given is A.1 MJD

28662.7087470068282567825742, which is the time of transmission, approximately 1.293 seconds before the time of reception at PI9CAM.

The positions of DSLWP and Shahe at this time are computed, and the distance between these positions gives a light-time of approximately 1.298 seconds. The reception time at Shahe is updated (note that now dT is positive) and its position at the new reception, as well as the position of the Earth, are calculated again to refine the light-time.

After a couple of iterations, the approximation converges, yielding

```
==> dEpoch = 1.298334164409e+00 second, dR = 389230.
```



Note that the light-times between DSLWP-B and PI9CAM, and DSLWP-B and Shahe are not exactly the same. This difference is precisely what the delta-range is measuring, and it implies that the reception time at Shahe is not the measurement timestamp.

As before, the positions of Shahe and the Earth are computed a final time with this light-time correction, obtaining

Summary for signal leg from DSLWP_B to Shahe:

- . Geometric range = 389263.327458642016 km
- . Light time solution range = 389230.782847323397 km
- . Relativity correction = 0.007605963255 km
- . ET-TAI correction = 33.451455045338 km
- . Feasibility = true

As before, there is a considerable difference of 32.545 km between the geometric range (which ignores the light-time correction) and the light-time solution. The relativity correction is again nearly 7.6 m, while the ET-TAI correction is large.

Since we are subtracting r_R and r_O to compute the delta-range, the correction terms for each of the legs are subtracted, so corrections that are very similar cancel out. In this way, the difference between considering the light-time correction and ignoring it manifests as a difference of 199 metres in delta-range.

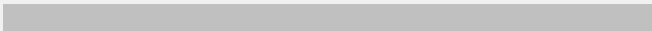
The difference between relativistic corrections is only 3.2cm. Such small value is to be expected. Now, the difference between ET-TAI corrections is 473 metres, which I find surprisingly high. I don't know if this is correct or even if it is meaningful to apply the ET-TAI correction to this simulation.

As before, we have the calculation of receive frequency and atmospheric corrections at Shahe. The tropospheric correction is 7.6 m, which is larger than the 4.3 m at PI9CAM, owing to the lower elevation of 18.5° at Shahe versus the 34.7° at PI9CAM. The effect of the ionosphere is also significant at this low frequency, and any precision work would need to model it.

```

+++++
+++ Signal Frequency calculation for leg from DSLWP_
+++++
. Arrival frequency : -1.000000000000e+00 Mhz
. Transmit frequency : 4.354000000000e+02 Mhz
. Doppler shift frequency: 4.353992753690e+02 Mhz
+++++
+++ Media corrections calculation for leg from DSLWF
+++++
.Frequency : 4.353992753690e+02 Mhz
.Troposphere range correction : 0.007635640939 m
.Troposphere elevation correction : 0.000968731245 ra

```



The summary for the GNRangeAdapter calculation of the other leg looks like this:

```

=====
==== GNRangeAdapter (DSNsimData_{DSLWP_B,Shahe}_Range
=====
. Path : DSLWP_B, Shahe,
. Measurement epoch : 28662.708747006829
. Measurement type : <DeltaRange>
. C-value w/o noise and bias : 389264.249543972895 km
. Noise adding option : true . Bias adding option : t
. C-value with noise and bias : 389264.249543972895 km
. Measurement epoch A1Mjd : 28662.708747006829
. Transmit frequency at receive epoch : 4.354000000000e
. Transmit frequency at transmit epoch : 4.354000000000e
. Measurement is feasible . Feasibility reason : N
. Elevation angle : 18.503745381468 degree
. Covariance matrix : <0x55bf39781f30>
. Covariance matrix size = 1
[ 1.000000e+00 ]

```

Now it's the turn for the DeltaRangeAdapter to subtract both ranges and apply noise and bias. It prints the following output:

```

=====
==== DeltaRangeAdapter (DSNsimData_{PI9CAM,DSLWP_B,Sh
=====
. Path : [ DSLWP_B -> PI9CAM ] - [ DSLWP_B -> Shahe ]
. Measurement epoch : 28662.708761972011
. Measurement type : <DeltaRange>
. Reference leg C-value w/o noise and bias : 387662.1
. Other leg C-value w/o noise and bias : 389264.24954
. C-value w/o noise and bias : -1602.095587405085 km
. Reference leg corrections : 32.990037089641 km
. Other leg corrections : 33.466696649532 km
. Corrections : -0.476659559891 km
. Noise adding option : true
. Bias adding option : true
. Range noise sigma : 1.100000000000 km
. Range bias : -3.000000000000 km
. Multiplier : 0.000000000000
. C-value with noise and bias : -1605.728116406530 km
. Measurement epoch A1Mjd : 28662.708761972011
. Transmit frequency at receive epoch : 0.000000000000
. Transmit frequency at transmit epoch : 0.000000000000
. Measurement is feasible
. Feasibility reason : N
. Elevation angle : 18.503745381468 degree
. Covariance matrix : <0x55bf3976b0d0>
. Covariance matrix size = 1
[ 1.210000e+00 ]

```

The measurement epoch is taken correctly as the the reception time at PI9CAM. The C-values for the reference and other legs are

taken correctly from the `GNRRangeAdapter` calculations. These include the corrections, but the corrections are also handled separately, so we can see their magnitude. We see that the delta-range computation amounts to -1602.096 km, of which 477 metres are corrections (mainly due to ET-TAI).

For the noise and bias we use a noise sigma of 1.1 km and a bias of -3 km. We can see these get applied to the delta-range.

The handling of the transmit and receive frequencies by this class is not yet implemented. The elevation angle is chosen as the minimum elevation of the two stations (since this is used as an indication of goodness of the measurement). The covariance matrix is computed correctly as the square of 1.1 km.

Conclusion

This post shows that, contrary to my first impression when I considered tackling this problem during the DSLWP-B mission, extending the orbit determination code in GMAT is not that difficult. After some familiarization with the class architecture it is easy to move things around, and the detailed debug information is really helpful to see what is going on.

Of course this work is just the first step to have a fully functional orbit determination pipeline for the DSLWP-B VLBI observations.

The code shown here needs to be tested with more simulations, and be used also for state estimation. Additionally, a measurement class for delta-range rates needs to be implemented. After the first experience, all this seems a natural and mostly straightforward continuation of this line of work.

One comment

Pingback: **Advances with delta-range and delta-range rate observations in GMAT – Daniel Estévez < <https://destevez.net/2020/06/advances-with-delta-range-and-delta-range-rate-observations-in-gmat/>>**

This site uses Akismet to reduce spam. **Learn how your comment data is processed < <https://akismet.com/privacy/>>**

.

DANIEL ESTÉVEZ < [HTTPS://DESTEVEZ.NET/](https://destevez.net/)>

*Proudly powered by **WordPress < <https://en-gb.wordpress.org/>>** .*