Relative-GPS-simulator

1. Scope

This simulator aims to evaluate the capabilities of a GPS sensor to estimate position of a follower satellite relative to leader satellite on low Earth orbit. You have to provide leader and follower trajectories and velocities in ECI as input (in my case, I got it from GMAT simulations) and to estimate the amount of noise in your GPS sensor. For the whole simulation time, this simulator provides relative position estimate by solving GPS nonlinear system using Multivariate Newton's method. It also provides absolute and relative errors on the true input trajectories.

It may seem a bit confusing to do some work to generate an output containing the exact information you have to provide as input… but it's all ok (figure 1). This script can be used to build more complex simulators dealing with sensor fusion or noise correction. For example, you can add a Kalman filter to the current output values to evaluate its capabilities to improve noisy position estimates. You can also use Kalman filter for sensor fusion to merge position estimates from different navigation sensors.

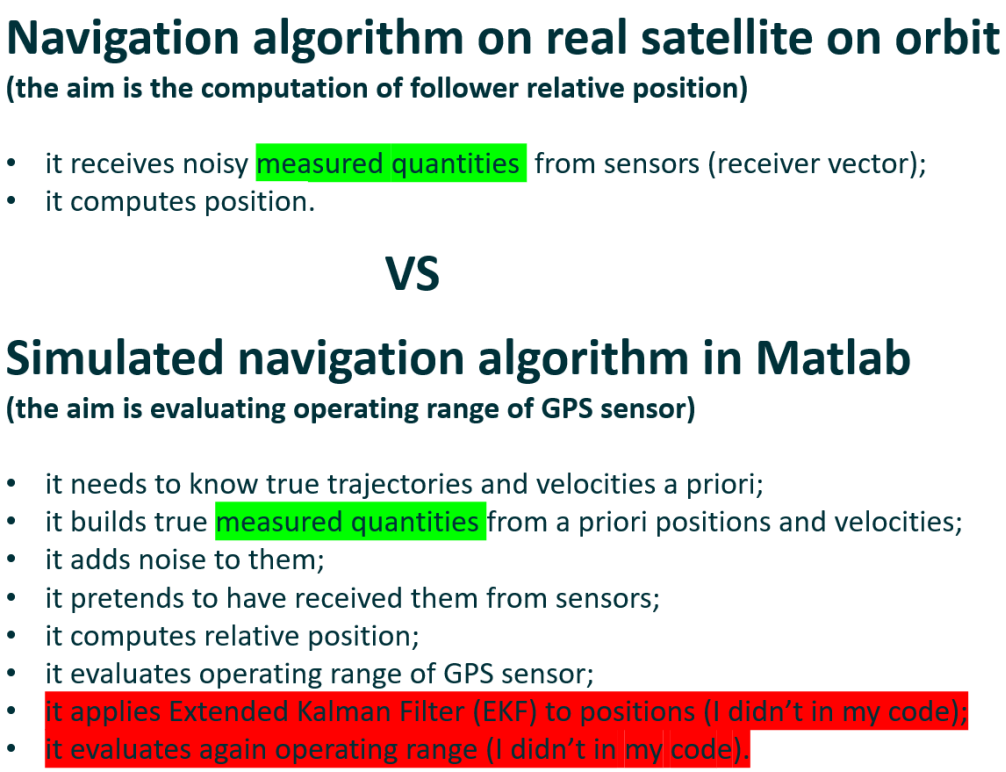

Figure 1: real world vs simulation

2. What you have to do

• Provide your trajectory and velocity data in "ECI_state_leader.txt" and "ECI_state_follower.txt" (data format is suggested in the first rows of txt files),

• Set GPS noise in "rgps_initialization.m",

• Launch "rgps_initialization.m",

• Enjoy results.

3. Acquisition of trajectory data

Acquisition of trajectory data from external simulations is a fundamental step to simulate the behaviour of a navigation sensor as GPS.
When a satellite equipped with GPS antenna is on orbit, GPS data (receiver vectors) are acquired by physical GPS sensor from the external environment. On the contrary, in simulations one has to artificially compute the receiver vectors from trajectory data (mission analysis studies).

This simulator includes GPS orbits from GMAT simulations. A GPS constellation of 24 satellites and 24 hours are considered ("ReportFileGPS_1_to_8.txt", "ReportFileGPS_9_to_16.txt", "ReportFileGPS_17_to_24.txt"). Every time a Matlab simulation is launched, a fraction of GPS trajectory data is used depending on the length of the selected simulation time interval (figure 2). Simulation time must be less than 24 h, otherwise you need to do some modifications in "rgps_initialization.m" and "allGPSpos_gmat.m".

In figure 2, the red stars represent the starting point of each GPS satellite at the initial time and the black star represents the origin of the reference system (ECI). This figure is plotted by the simulator to check trajectories of GPS satellites. Red and blue curves represent original trajectories by txt files and spline interpolation of trajectories by "allGPSpos_gmat.m". If everything is ok, blue and red trajectories overlap.

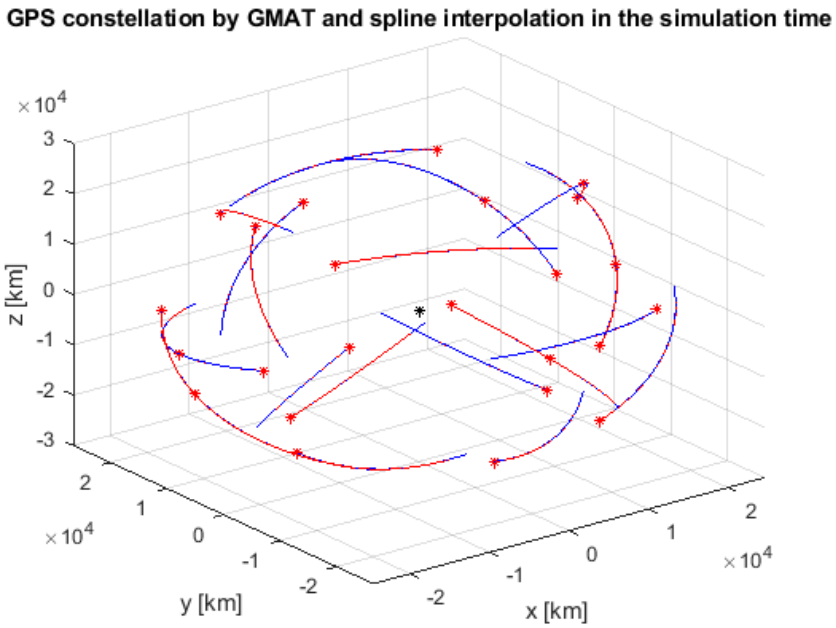

***Figure 2**: selection of GPS orbit fractions according to the simulation time*

4. Implementation of relative GPS algorithm

The general structure of the simulator is described in figure 3.

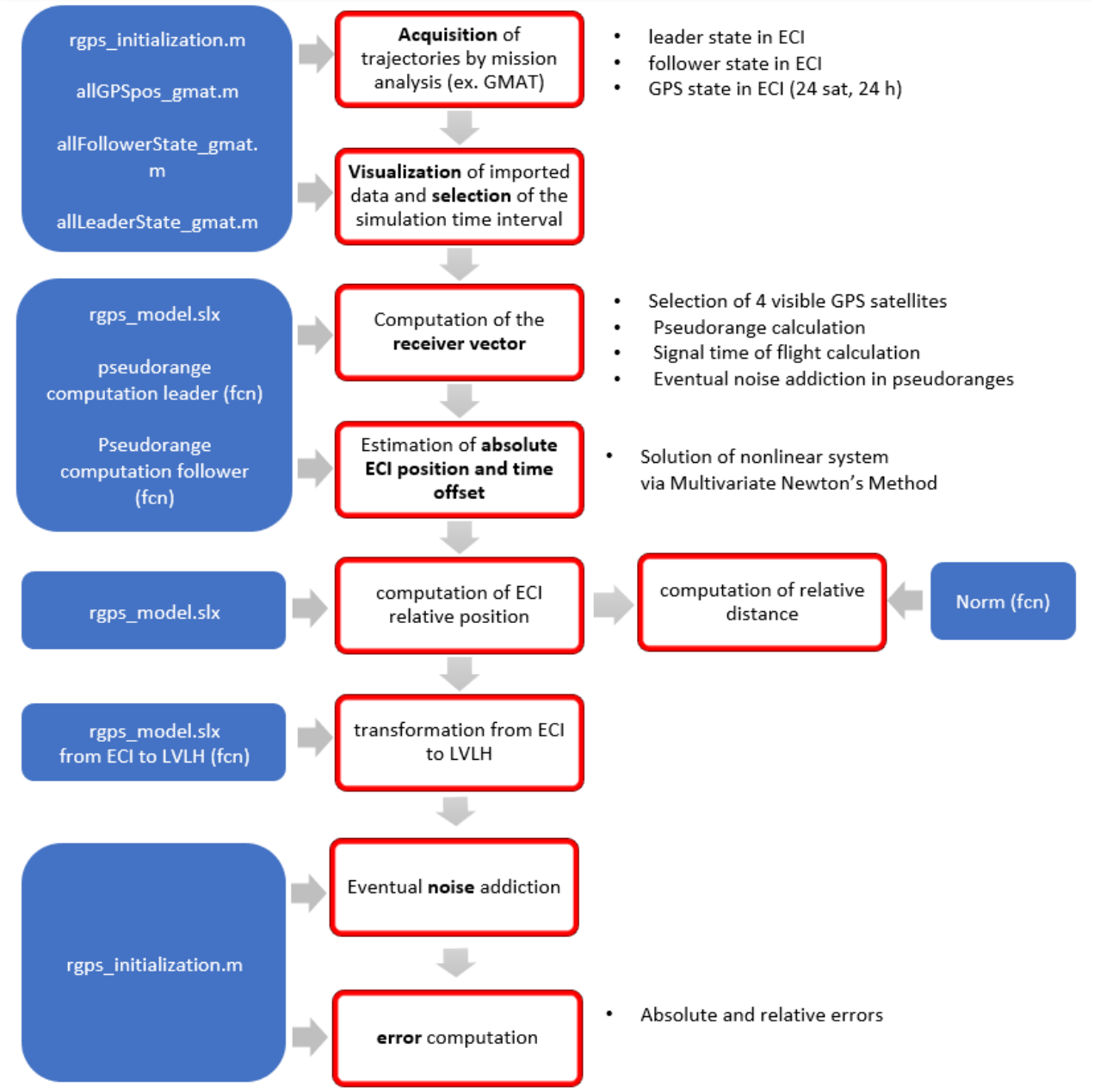

Figure 3: simulation process for relative GPS sensor

The solution of the GPS nonlinear system [1] is the crucial step in the simulation process. It requires the implementation of Multivariate Newton's Method [2].
The nonlinear system is the following:

$$\begin{cases} (x_{n1} - x_u)^2 + (y_{n1} - y_u)^2 + (z_{n1} - z_u)^2 = c^2 \cdot (t_{u1} + t_{bias} - t_{n1})^2 \\ (x_{n2} - x_u)^2 + (y_{n2} - y_u)^2 + (z_{n2} - z_u)^2 = c^2 \cdot (t_{u2} + t_{bias} - t_{n2})^2 \\ (x_{n3} - x_u)^2 + (y_{n3} - y_u)^2 + (z_{n3} - z_u)^2 = c^2 \cdot (t_{u3} + t_{bias} - t_{n3})^2 \\ (x_{n4} - x_u)^2 + (y_{n4} - y_u)^2 + (z_{n4} - z_u)^2 = c^2 \cdot (t_{u4} + t_{bias} - t_{n4})^2 \end{cases}$$

where:

- x, y, z are ECI coordinates,
- t_{bias} is the time offset due to the user clock,
- c is the speed of light,
- $n_{1,2,3,4}$ identifies a GPS visible satellite,
- u identifies the user satellite (follower or leader satellite).

One cannot use powerful Matlab functions as "solve" to solve the nonlinear system, because they do not work in "Simulink / user-defined-function" where the nonlinear system is located (**double click on fcn blocks in the Simulink model to see this code**) [3].
One also cannot use least squares method because the system is not overdetermined (4 GPS satellite surely visible from any position in low Earth orbit and 4 equations).

The nonlinear system is solved twice, once for the follower satellite and once for the leader satellite, so that we have ECI absolute position estimate for both follower and leader. These two estimates are independent of each other. Then, absolute leader position is subtracted from absolute follower position to compute relative position in ECI. The code transforms the reference system from ECI to LVLH through rotation and translation of reference axes. The norm of the relative position vector in LVLH give us the relative distance between leader and follower (it coincides with the norm of relative position vector in ECI).

On orbit position estimate is always possible in low Earth orbits, because at least 4 GPS satellites are visible in every point of all of them (figure 4). However, accuracy of position estimates via GPS is not sufficient for some phases of some missions as docking. This simulator can be useful to establish the operative range for GPS sensor in your particular mission.

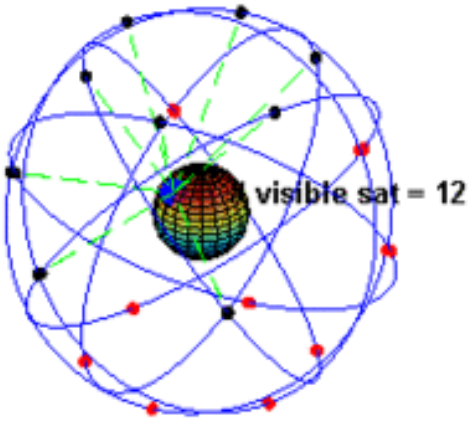

Figure 4: GPS constellation, visible satellites

1. References

[1] W. Fehse, Automated Rendevouz and Docking of Spacecraft, 2003 (cap. 7) → for GPS nonlinear system
[2] http://mason.gmu.edu/~nulloaso/Project2.html → for solution of GPS nonlinear system
[3] https://www.mathworks.com/help/simulink/slref/fcn.html → for simulink user-defined-function