REAL TIME IMPLEMENTATION OF A LOW-COST INS/GPS SYSTEM USING xPC TARGET

José Adalberto França Junior

Missile and Rockets Section, Army Tecnological Center, Rio de Janeiro, Brazil, adalberto@ctex.eb.br

Jorge Audrin Morgado

Missile and Rockets Section, Army Tecnological Center, Rio de Janeiro, Brazil, audrin@ime.eb.br

Abstract: *A Low Cost INS/GPS system (Inertial Navigation System / Global Positioning System) was built with an IMU (Inertial Measurement Unit) based on MEMS (Micro Electro-Mechanical System) technology. The chosen platform for the real time implementation of the integrated INS/GPS algorithm was MATLAB/Simulink with Real Time Workshop (RTW) and xPC Target toolboxes. To make the sensor's data fusion, the linear Kalman Filter were implemented based on the error dynamics navigation equations. The INS/GPS System was mounted on a vehicle which traveled a georeferenced path, and its results were recorded and evaluated. With the many GPS outages occurred during the vehicle trajectory, a relative simple way to overcome this issue is proposed. The equipment used in this experiments were a PC104 computer which processed in real time the data from a Crossbow IMU and a Garmin GPS which reads position, velocity and heading. Also, a NAV420, a commercial INS/GPS System from Crossbow, was used to compare with xPC Target results.*

Keywords: *INS/GPS, Kalman Filter, xpC Target, MEMS, IMU, Simulink.*

1 Introduction

This work describes an INS/GPS algorithm and how to implement it in MATLAB/Simulink and xPC Target toolbox. With an INS/GPS System mounted on a vehicle, a pre-set path was made on a georeferenced way. The results generated and recorded by xPC Target was evaluated, and some issues regarding the GPS outages were had been led in account. Also, a NAV420, a commercial INS/GPS System from Crossbow, was used to compare with xPC Target results.

The main cause of the INS accumulated errors growing with time are due to the sensor's errors measurement. These problem can be attenuated if we use the INS integrated with other non inertial sensors like GPS, baroaltimeter, magnetometer, etc. If we choose the GPS to aid the INS, for example, we have a INS/GPS System. Knowing that the GPS also has its error's components, the Kalman Filter is normally used to make the INS/GPS data fusion and thus supply the desired parameters with the biggest exactness possible. On this work, a linear Kalman filter were implemented based on the error dynamics navigation equations.

Regarding of real time implementation, until now, the development of the executable code implementing an Inertial Navigation System required good C/C++ programming skills or any other low level languages and, sometimes, a deep knowledge of the numerical methods involved. An interesting way to overcome such a hurdle is offered by the MATLAB/Simulink/Real-Time-Workshop, or simple RTW. The RTW controls the translation of the blocks in a series of C functions that can be compiled and linked to obtain an executable file. The generated code can run on most Real Time Operating Systems (RTOS) like, QNX, VXWorks, RTAI Linux, xPC Target, etc. The usual learning time associated with a new software is virtually eliminated, meaning both a training time/cost saving and the possibility of focusing on more sophisticated issues regarding navigation and data fusion strategies.

2 INS mechanization

The navigation parameters at navigational frame (n) are constituted by 9 parameters (Savage, 1998). Parameters of position (r^n), which defines the position of the body by Latitude (φ), Longitude (λ) and Altitude (h); parameters of speed (v^n), which specifies the linear velocities of the body by North Velocity (V_N), East Velocity (V_E) and Down Velocity (V_D); and parameters of attitude (C_b^n), which specifies the angular orientation of the body by Roll (ϕ), Pitch (θ) and Yaw (ψ).

In vector form we have:

$$r^n = [\varphi \; \lambda \; h]^T$$

$$v^n = [v_N \; v_E \; v_D]^T$$

$$C_b^n = \left[\begin{array}{c|c|c} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\theta s\phi c\psi \\ \hline c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ \hline -s\theta & s\phi c\theta & c\phi c\theta \end{array}\right]$$

Where C_b^n is the Direction Cosine Matrix (DCM) that transforms attitude coordinates from the body frame (b) to the navigational frame (n) (Bekir, 2007).

As follows, we have the equations that describes the time evolution of the navigation parameters in function of the data received from the inertial sensors (accelerometers and gyroscopes) (Titterton and Weston, 1997):

$$\begin{pmatrix} \underline{\dot{r}}^n \\ \underline{\dot{v}}^n \\ \dot{C}_b^n \end{pmatrix} = \begin{pmatrix} D^{-1}\underline{v}^n \\ C_b^n \underline{f}^b - (2\underline{\omega}_{ie}^n + \underline{\omega}_{en}^n) \times \underline{v}^n + \underline{g}^n \\ -\Omega_{bn}^n C_b^n \end{pmatrix} \quad (1)$$

with

$$D^{-1} = \begin{pmatrix} \frac{1}{R_m+h} & 0 & 0 \\ 0 & \frac{1}{(R_p+h)cos\varphi} & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

where - $\underline{f}^b$ is the specific force in (b) frame or the accelerometer measurement;

- ω_{ie}^n is the Earth rate with respect to inertial frame (i) expressed in navigational frame (n);

- ω_{en}^n is the angular rate of (n) frame relative to earth frame (e) expressed in (n) frame;

- $\underline{g}^n$ is the gravity vector expressed in (n) frame;

- Ω_{bn}^n is the skew-symmetric matrix of ω_{bn}^n which is the angular rate of (n) frame relative to (b) frame expressed in (n) frame; and

- R_m and R_p are respectively the meridian radius of curvature and the prime radius of curvature (Bekir, 2007).

3 INS/GPS integration Kalman filter

On this paper the Loosely Coupled implementation to integrate the GPS and INS was used. The nine-state INS/GPS integration Kalman filter was built using the error dynamics equations (Shin, 2001). The error analysis utilizes perturbation methods to linearize the nonlinear system differential navigation equations (Briting, 1971).

The mathematical modeling of inertial systems is usually performed in the framework of linear dynamic systems using a state-space representation. In this case, a system of non-linear first-order differential equations is implemented and the solution of it is provided through available kinematic measurements. The solution of such differential equations will provide the position, velocity and attitude of the inertial system carrier. Typically, the provided solution will contain errors due to the existing inertial sensor errors. Hence, these errors are determined first through error models and then compensation for them is carried out. In this case, the dynamic behavior of INS errors is represented in matrix form by a discrete state-space representation of the form:

$$\underline{x}_{k+1} = \Phi_k \underline{x}_k + G_k \underline{w}_k,$$

$$\underline{z}_k = H_k \underline{x}_k + \underline{e}_k$$

with

$$\underline{x}_k = [\delta\underline{r}^n \;\; \delta\underline{v}^n \;\; \underline{\epsilon}^n]^T$$

$$\underline{z}_k = \begin{pmatrix} \varphi_{INS} - \varphi_{GPS} \\ \lambda_{INS} - \lambda_{GPS} \\ h_{INS} - h_{GPS} \\ v_{N_{INS}} - v_{N_{GPS}} \\ v_{E_{INS}} - v_{E_{GPS}} \\ v_{D_{INS}} - v_{D_{GPS}} \end{pmatrix}$$

where - $\delta\underline{r}^n$ is the vector of position errors (latitude φ , longitude λ and height h);

- $\delta\underline{v}^n$ is the vector of velocity errors (north, east and down);

- $\underline{\epsilon}^n$ is the vector of misalignment errors for the transformation between the b-frame and the n-frame

- $\underline{x}_{k+1}$ is the system error state vector to be estimated at time t_{k+1};

- Φ_k is the system state transition matrix;

- $\underline{w}_k$ is the vector of the system input random noise;

- G_k is the coefficient matrix associated with the system input noise;

- $\underline{z}_k$ is the vector of the system observations (updating measurements) at time t_{k+1};

- H_k is the design matrix relating the system measurements to the system error states; and

- $\underline{e}_k$ is the vector of measurements random noise.

As mentioned before, for the optimal estimation of the INS error state vector components, a Kalman Filter is usually used. The discrete KF equations is summarized in Titterton and Weston (1997).

4 Initialization and coarse alignment with MEMS sensors

On this work, the static initialization was used (Titterton and Weston, 1997). The initial parameters of position (latitude, longitude and height) are supplied from the GPS itself with a mean over the initialization time of the GPS position data. The initial velocities data are supplied with zero, once the vehicle is stopped. The attitude angles however, is initialized by a alignment procedure which can varies according to the type of inertial sensors used. In this case, and in this test with the vehicle, only a coarse alignment for MEMS sensors was used to estimate the initial attitude parameters. This procedure are described in Bekir (2007) and consists in calculate the roll and pitch angles through the accelerometers measurements which is shown in (2) and (3):

$$\hat{\phi} = tan^{-1}(\frac{\hat{a}_y}{\hat{a}_z}) \tag{2}$$

$$\hat{\theta} = tan^{-1}(\frac{-\hat{a}_x}{\sqrt{\hat{a}_y^2 + \hat{a}_z^2}}) \tag{3}$$

where $\hat{a}_x$, $\hat{a}_y$ and $\hat{a}_z$ is the components of the linear accelerations measured by the accelerometers.

The heading angle is normally calculated through the gyros measurements. However, with MEMS technology, the gyros output errors is so high that they are rendered unsuitable. Thus, another Attitude and Heading Reference System (AHRS) like a magnetometer is used to aid the heading information, not only in the initialization phase but in the navigation too.

5 Real time implementation with xPC Target

The xPC Target product is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. It is an environment that uses a target PC, separate from a host PC, for running real-time applications.

The xPC Target toolbox offers a particularly attractive Target for two very simple reasons: 1) the toolbox included interface blocks which were essentially device drivers written in C and 2) the toolbox turns any desktop PC into a Real-Time system on which the Pointing Algorithm could be deployed. This was accomplished by booting the Desktop PC into the Vendor provided RTOS. Since this toolbox comes with the device driver blocks, it is not necessary to re-write these device drivers for the Vendor's RTOS. Furthermore, the source code is available for inspection and customization by the user. Fig. 1 shows the model in Simulink used for code generation and deployment on xPC Target.

The block 4 of Fig. 1 it's the main block of the whole algorithm, it is executed at each system sample time calculating the navigation parameters. Inside this block it's implemented the navigation equations (1) and the discrete KF equations. Although block 4 is executed at the system sample rate which is the same rate of the IMU's

Table 1. Crossbow IMU400CD-200 Specifications

Gyro Characteristics	**Value**
Range Roll, Pitch, Head (°/sec)	± 200
Bias Roll, Pitch, Head (°/sec)	< ± 1.0
Scale Factor Accuracy (%)	< 1
Non-Linearity (% FS)	< 0.3
Resolution (°/sec)	< 0.05
Bandwidth (Hz)	> 25
Random Walk (°/hr$^{1/2}$)	< 4.5
Accel Characteristics	**Value**
Range X, Y, Z (g)	± 4
Bias X, Y, Z (mg)	< ± 12
Factor Accuracy (%)	< 1
Non-Linearity (% FS)	< 1
Resolution (mg)	< 0.6
Bandwidth (Hz)	> 75
Random Walk (m/s/hr$^{1/2}$)	< 1

reading, the Kalman Filter is executed at a slower rate, which is the GPS sample rate. Thus, the INS compensation through the vector $\underline{x}_k$ is done only at GPS timing. Fig. 2 shows the block 4 of Fig. 1 exploded in more details.

The block 3 of Fig. 1 is the initialization block. in this block it's implemented the procedure discussed in section 4. Still in Fig. 1, the blocks 1 and 2 are the external interfaces with the sensors. Figure 3 shows block 1 implementing the serial communication with the Crossbow IMU400. It can be noticed the "Baseboard Serial" block which is a xPC's box that make the RS232 readings. At the same way, Fig. 4 shows block 2 implementing the serial communication with the Garmin GPS.

6 Test setup

This section shows the test made with the INS/GPS System mounted on a vehicle. This system can be seen in Fig. 5 and Fig. 6.

The commercial IMU used in this experiment was a Crossbow model IMU400CD-200. It is composed of 3 accelerometers and 3 gyros with temperature compensated and with digital (RS-232) and analog (12-bit DAC) interface outputs(Crossbow, 2007). This strapdown inertial system provides measurement of linear acceleration and angular rate. The data acquisition was performed through a RS-232 communication serial port, whose protocol is a Crossbow proprietary format. Its maximum sampling rate is 133Hz configurable. On this work was used a sample rate of 20Hz. The IMU has a label on the top illustrating the IMU coordinate system as shown in Fig. 5. For taking measurements, the scaled sensor mode (Crossbow, 2007) has been used, so the analog sensors were sampled, properly converted to digital data, temperature compensated and scaled to engineering units. Table 1 gives the main specifications of the IMU.

The GPS used was a Garmin GPS 18-5Hz which has high-sensitivity, WAAS-enabled with RS-232 interface. It measures position (latitude, longitude and height), speed (Vn, Ve and Vd) and heading information. It has a maximum speed of 5Hz data measurement which was the frequency used in this work. Its main specifications can be seen in table 2. Also, it can be seen at the system assemble, the CrossBow NAV420 and its own GPS. The

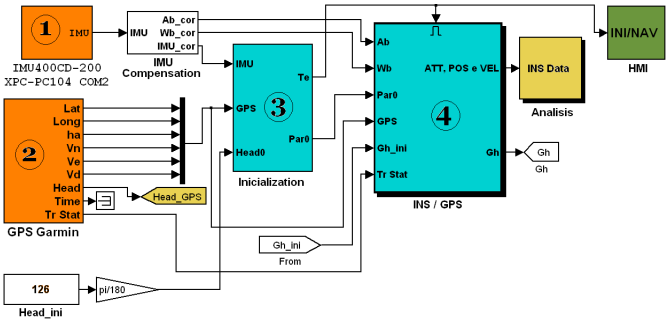

Figure 1. INS/GPS Algorithm implemented in Simulink for code generation on xPC Target

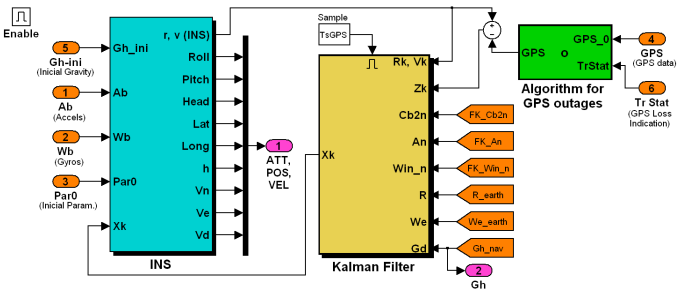

Figure 2. Block 4 in details - INS/GPS integrated algorithm using the Loosely Coupled implementation

Table 2. Garmin GPS 18-5Hz Specifications

GPS Characteristics	Value
Receiver	WAAS-enabled; 12 parallel channel GPS receiver continuously tracks and uses up to 12 satellites to compute and update your position
GPS position accuracy	< 15 meters, 95% typical
GPS velocity accuracy	0.1 knot RMS steady state
Dynamics	999 knots, 6 g's
Measurement Pulse Output	5 Hz pulse, programmable width, 1 microsecond accuracy
Operating temperature	$-30°C$ to $80°C$

NAV420 is a combined Navigation and AHRS that measures stabilized pitch, roll and yaw angles in a dynamic environment along with GPS-based position and velocity. In other words its a complete solution in a INS/GPS system, and it will be used for comparison with the navigation parameters calculated by our system. The main specifications of the NAV420 can be seen in table 3:

In this experiment the magnetometer was not used as an auxiliary AHRS, which is extremely recommended in MEMS inertial systems. Instead a magnetometer, we used the GPS itself to provide the heading information, which is the critical data provided by a MEMS INS system. This is because the heading angle is estimated based on the Earth's angular velocity (w_e), and MEMS gyros cannot read it. However, the GPS heading information is only available when the vehicle is moving, and since the alignment is motionless, we still need the initial heading, which was provided by the NAV420 and inserted manually in our algorithm. Instead of compensate the heading

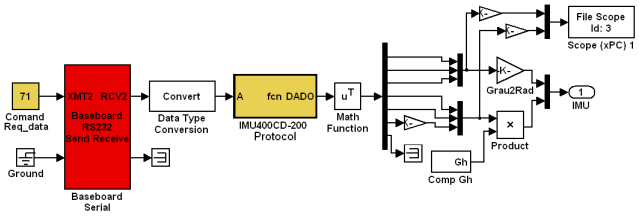

Figure 3. Block 1 in details - Serial communication with the Crossbow IMU400CD-200 using xPC Target toolbox

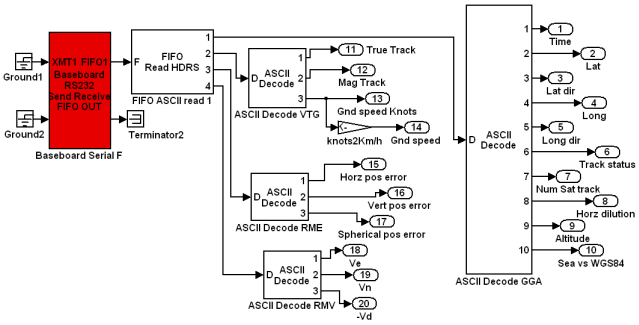

Figure 4. Block 2 in details - Serial communication with the Garmin GPS 18-5Hz using xPC Target toolbox

Figure 5. INS/GPS System mounted on the vehicle

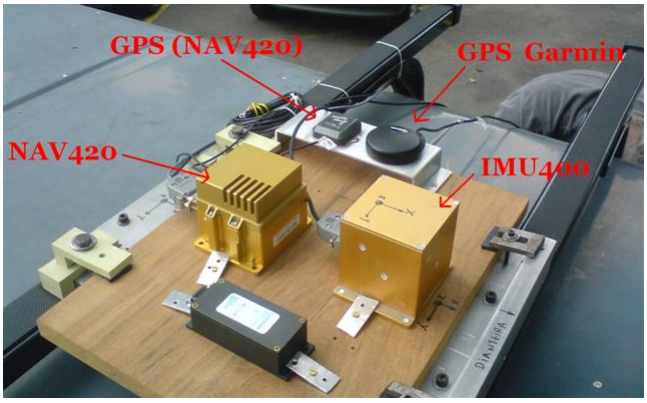

Figure 6. INS/GPS System

Table 3. CrossBow NAV420 Specifications

NAV420 Characteristics	**Value**
Performance	
Update Rate (Hz)	2-100 (Programmable)
Position/Velocity	
Position Accuracy (m)	3 (Internal GPS, not augmented)
X,Y Velocity Accuracy (m/s rms)	< 0.4 (GPS available)
Z Velocity Accuracy (m/s rms)	< 0.5 (GPS available)
Attitude	
Range: Roll, Pitch (°)	± 180, ± 90
Accuracy (° rms)	< 0.75 (GPS available)
Accuracy (° rms)	< 2.5 (GPS unavailable)
Heading	
Range (°)	± 180
Accuracy (° rms)	< 3.0
Angular Rate	
Range: Roll, Pitch, Yaw (°/sec)	± 200
Bias: Roll, Pitch, Yaw (°/sec)	< ± 0.1 (Kalman filter stabilized)
Bias: Roll, Pitch, Yaw (°/sec)	< ± 0.75 (Kalman filter off)
Acceleration	
Input Range: X/Y/Z (g)	± 4
Bias: X/Y/Z (mg)	< ± 15

parameter with the Kalman Filter estimation, it was compensated directly with the heading given by GPS.

To process the sensors data and calculate the navigation parameters, a PC104 (Pentium II 400MHz with 128 MB Ram) was used inside the vehicle, as well a laptop connected to one of its serial ports acting as HMI (Human Machine Interface).

The trajectory covered for the vehicle on this test can be seen in Fig. 7. To be able to compare the navigation parameters calculated by our algorithm with reliable results, we added landmarks over the path signaled by traffic cones where their positions of latitude, longitude and height were measured by a DGPS. With the cone's latitude and longitude information, these landmarks had been located in Google Earth through Matlab with the commands "kmlwrite" and " winopen" and are represented by yellow diamonds in Fig. 7.

The vehicle made the following steps: got ready in landmark 1 (diamond 1 of Fig. 7). With the vehicle turned off, the sensors (IMU400, NAV420 and GPS Garmin) was turned on followed, 1 minute later, by the PC104, which initiated the algorithm (time zero). When the algorithm switched from the alignment phase to the navigation phase, which could be verified by the HMI 170 seconds later (time 170s), the vehicle was turned on at time 260s and started the navigation, making the described trajectory by the red arrows (Fig. 7). When passing by landmarks, the associated time was inserted in the algorithm manually. This times are necessary to compare the knowing position of cones with the position calculated by the algorithm at the same time. During the running, all data, from sensors reading to navigation parameters were recorded to the PC104's hard disk.

7 Tests results

During the many runnings made on this trajectory, some issues were noticed regarding the GPS information. First of all, the path is covered for many trees that blocks the satellites signals, and therefore, the altitude, velocities and heading information had been a little degraded by momentary brusque peaks and falls which was sufficient to corrupt the system response. All of these brusque variations of the GPS had coincided with the GPS outages (Fig. 8).

The traditional way to overcome this issue is simply to use only the INS part of the algorithm (pure INS) without the corrections of the KF, until the GPS signals return. However, this way alone is only valid for high grade inertial sensors, and is not valid for MEMS sensors whose navigation without the GPS does not remains enough time with the necessary precision. This was verified on this experiment, and therefore, a sufficiently simple artifice was used to minimize this problem. The artifice was simply never use the pure INS only, and when the GPS lacks, its last ones data would be read instead of reading the real GPS signals. Although simple, this method revealed efficient for this specific case where the GPS loss had not exceeded 15 seconds. For even better results, more complex algorithms with trajectory predictions can be implemented, instead of repeating the last GPS datas.

The navigation results can be seen on figures bellow. Figures 9, 11 and 12 shows the landmark's position parameters compared separately with the INS/GPS System and the NAV420.

Comparing the NAV420 and the INS/GPS System, we can observe that the NAV420 has a lower error in its latitude and longitude signals, probably because to its treatment given for the GPS outages that must include more sophisticated algorithms of trajectory predictions. For the altitude signal however, we can see the NAV420 error a little higher. Being the altitude the information of biggest error of a GPS, we assume that the NAV420's GPS presented a bigger variation in this signal during the GPS loss than the Garmin GPS.

Finally, it is shown the navigation results in Google Earth. Figure 10 shows the data of the GPS Garmin and INS/GPS System together with the landmarks.

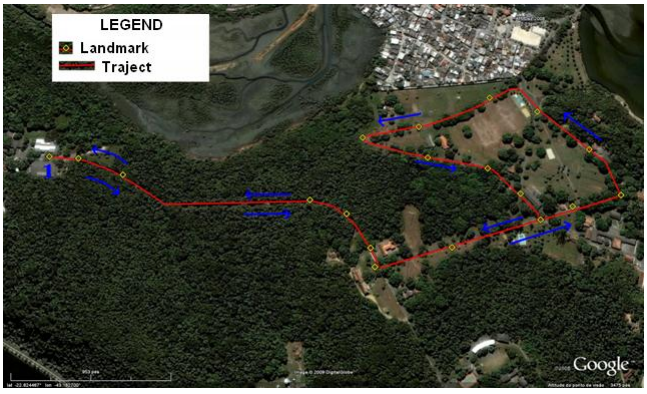

Figure 7. Trajectory covered by the vehicle in the test

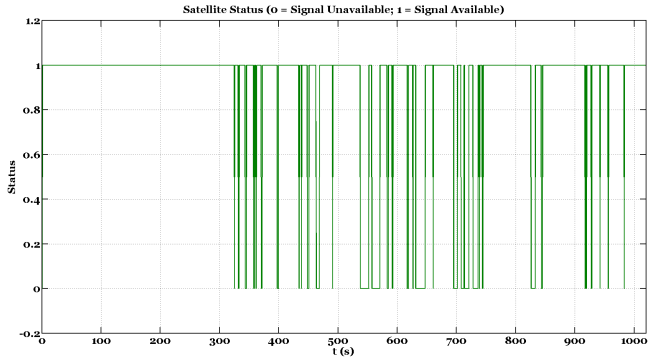

Figure 8. GPS outages

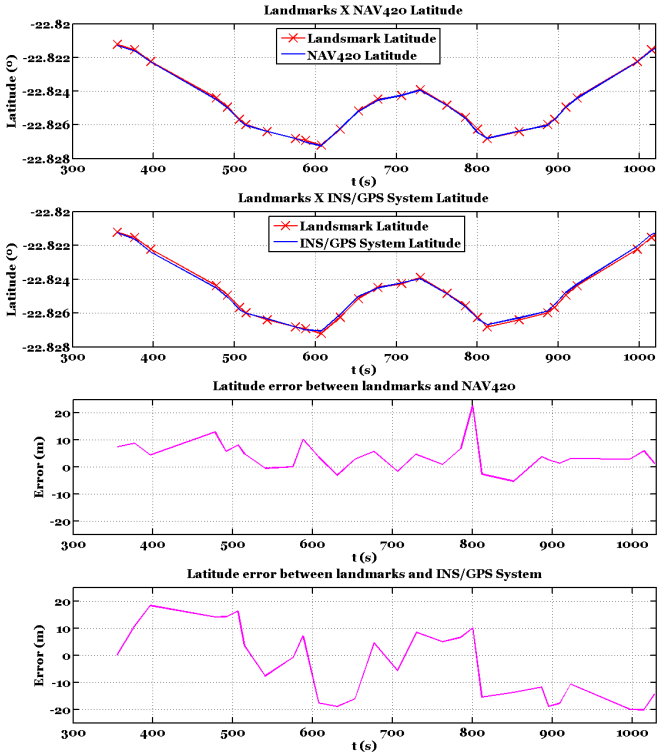

Figure 9. Latitude comparison between NAV420 and INS/GPS System with Landmarks

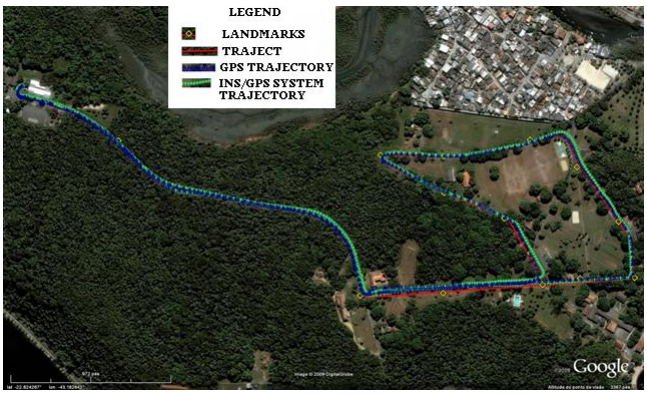

Figure 10. Google Earth Plot - Latitude and longitude information of the NAV420, INS/GPS System and landmarks

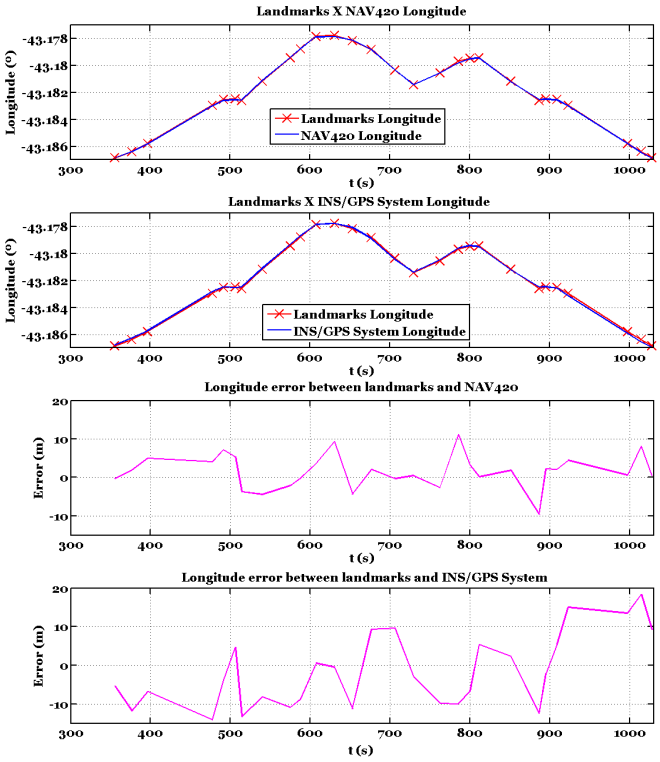

Figure 11. Longitude comparison between NAV420 and INS/GPS System with Landmarks

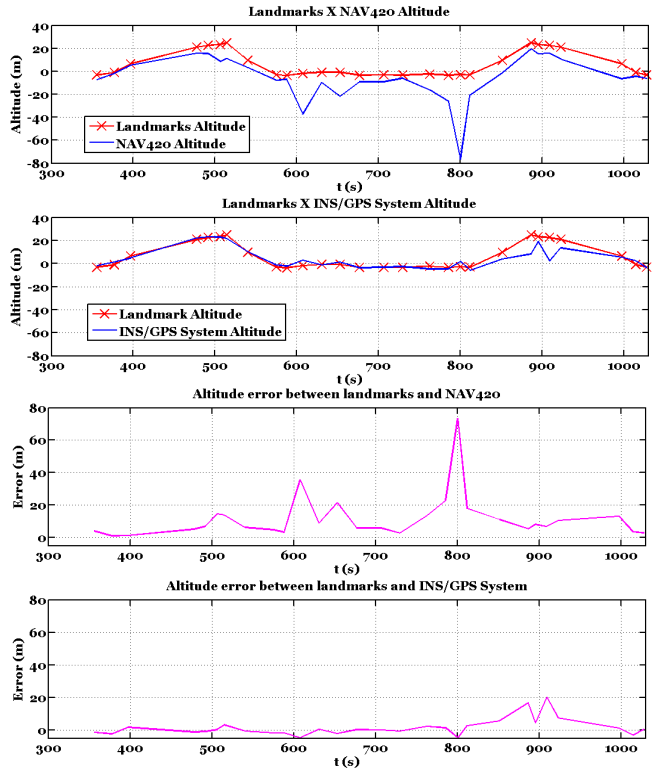

Figure 12. Altitude comparison between NAV420 and INS/GPS System with Landmarks

7 Conclusion

This paper presents an approach to implement in real time a INS/GPS algorithm with xPC Target. The accomplished objective was to make a low-cost INS/GPS System with a easy to use tool which is provided by Mathworks. With the experimental test analysis, the MEMS based INS/GPS System presented a good exactness on its results if we consider the great GPS outages over the trajectory. The software MATLAB/Simulink with the toolboxes Real Time Workshop and xPC Target shown us to be robust and reliable tools for real time implementation. Being tools with friendly and intuitive interfaces, they save a considerable time if we compare with others low-level languages like C/C++ and FORTRAN for example. However, as the model complexity increases, also increases the level of knowledge necessary for these tools.

8 References

Bekir, E. *Introduction to Modern Navigation Systems*. World Scientific, 2007.

Briting, K. R. *Inertial Navigation Systems Analysis*. 1971.

Savage, P. G. Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, 1998.

Shin, E.-H. Accuracy improvement of low cost INS/GPS for land applications. Master's thesis, University of Calgary, Canada, 2001.

Titterton, D. H. and Weston, J. *Strapdown Inertial Navigation Technology*. Peter Peregrinus Ltd., 1997.