NaveGo: a simulation framework for low-cost integrated navigation systems ⋆

Rodrigo Gonzalez * **Juan Ignacio Giribet** **
Héctor Daniel Patiño ***

* *Laboratorio de Computación Reconfigurable and GridTICs, Facultad Regional Mendoza, Universidad Tecnológica Nacional, Mendoza, Argentina (e-mail: rodralez@frm.utn.edu.ar)*
** *Departamento de Electrónica, Universidad de Buenos Aires, Buenos Aires, Argentina (e-mail: jgiribet@fi.uba.ar)*
*** *Instituto de Automática, Universidad Nacional de San Juan, San Juan, Argentina (e-mail: dpatino@inaut.unsj.edu.ar)*

Abstract: In this work, a simulation framework for low-cost integrated navigation systems called NaveGo is presented. NaveGo simulates measurements from inertial sensors and a GPS receiver. It also executes a loosely-coupled navigation algorithm. Complete mathematical models are shown. As example, simulation results for a particular scenario are exhibited. NaveGo is developed in MATLAB and freely distributed under an open source license. It is the authors' believe that this article is the most complete work about a simulation environment for low-cost integrated navigation systems. NaveGo is validated by using a practical approach, which main goal is to evaluate how close in performance is a simulated navigation system to a real one. Thus, simulated sensors are generated based upon a real trajectory, during which real sensors were logged. Then, simulated sensors are corrupted with noises according to real sensors error profiles. Both simulated and real navigation systems are processed. It is found that absolute differences between real and simulated systems are under 0.6 degrees for attitude, under 23 centimetres for 2D position, and under 10 centimetres for vertical position. As a result, it is verified that NaveGo is a suitable simulation tool for the design and analysis of low-cost integrated navigation systems.

Keywords: simulation; integrated navigation system; inertial sensors; INS; IMU; GPS; MATLAB

1. INTRODUCTION

A strapdown inertial navigation system (SINS) is an electronic device rigidly mounted on a vehicle, compound of an onboard computer and an inertial measurement unit (IMU). An IMU contains accelerometers and gyroscopes, whose axis are aligned respect to the vehicle body frame, to continuously calculate estimates of orientation, position, and velocity.

An integrated navigation system (INS) is mainly composed of a SINS and aiding sensors, such as a GPS receiver and magnetometers, that contribute to bound the accumulative errors of a SINS. An INS also provides estimates of position and velocity, as well as orientation parameters for a moving platform. Recently, with advances in Micro Electro-Mechanical Systems technology (MEMS), commercial-off-the-shelf, MEMS inertial sensors are available. Nowadays, INS built on MEMS inertial sensors are subject of great interest due to their unique characteristics, such as light weight, low power consumption, and low cost. These features make MEMS-based INS suitable for several applications, e.g., positioning of micro-flying and micro-land robots.

Throughout the design stage of an INS, prior to implementation, it is essential that designers have at hand a simulation framework to evaluate and predict the behaviour of this complex, highly nonlinear system. Thus, designers can rapidly try several schemes and components and find the best trade-off between precision and costs. For instance, it should be very practical to analyse different sensors at simulation level, and subsequently choose the best sensors that satisfy the prerequisite of precision at the lowest cost. On the other hand, from an embedded system perspective, navigation algorithms may be evaluated for different numerical precisions (fixed-point and floating-point), numerical stability, computational load, use of memory, and therefore, to determine the best embedded microcomputer architecture that achieves the required computational complexity.

In this article, a simulation framework for low-cost integrated navigation systems called NaveGo is presented. Complete mathematical models are given. Equations to generate simulated data coming from gyroscopes, accelerometers, and a GPS receiver are shown. Additionally, the mathematical model of a loosely-coupled INS is

⋆ This work was supported by the Agencia Nacional de Promoción Científica y Tecnológica, Argentina; and the Universidad Tecnológica Nacional, Argentina.

provided. NaveGo is based upon the mathematical model exhibited in (Gonzalez et al., 2015), which is a model compiled from recognized authors in the field of navigation. NaveGo is written in MATLAB language and is provided as a free open-source toolbox (Navego, 2015).

In the reviewed literature few papers have points in common with this work, and none is so comprehensive. In (Giroux et al., 2003), (Wenling et al., 2010), (Liansheng and Tao, 2011), and (Zhang et al., 2012) only mathematical models of a SINS are shown. Works from (Esposito et al., 2007) and (Lijun et al., 2008) exhibit an INS simulator, but none of them expose complete mathematical models. Although simulation results are shown in all these works, most do not expose a validation method to assess the proposed simulator, except in (Giroux et al., 2003). Thus, to the best of the authors' knowledge, NaveGo is the most complete work about a simulation environment of low-cost integrated navigation systems.

The method to validate NaveGo evaluates the performances of three pairs of INS systems. Each pair is composed of both simulated and real sensors. In (Toth et al., 2011) several IMU and a GPS were logged on field. A DGPS was also implemented. A reference data set is formed fusing a navigation-grade IMU and DGPS. Three real MEMS IMU are chosen to represent MEMS IMU from low to high qualities. Simulation of the three MEMS IMU and GPS is carried out by using the reference data set. Sampling times between simulated and real sensors are matched. Then, simulated data sets are corrupted with noises according to the error characteristics from real sensors. Finally, RMS errors from both simulated and real INS are compared for each MEMS IMU.

The rest of this article is organized as follows. Section 2 shows the mathematical models to simulate of proposed sensors. Section 3 presents the Kalman filter equations used in NaveGo. Section 4 provides the main algorithm of NaveGo. Section 5 displays simulation results for a particular case. Section 6 exposes the validation method for NaveGo. Finally, in Section 7 concluding remarks are commented. Additionally, in Appendix the functions included in the toolbox are listed.

2. SENSORS SIMULATION

This section explains how gyroscopes, accelerometers, and a GPS receiver are simulated.

First, some comments about mathematical notation. Equations presented in this work are in the discrete-time domain, except otherwise stated. Besides, variables with subscript $(-)$ correspond to the former interval sample, $t_{(k-1)}$. On the other hand, variables with subscript $(+)$ belong to the later interval sample, $t_{(k+1)}$. Noisy measurements have as superscript a tilde $(\sim)$. Estimated variables based upon noisy measurements have as superscript a hat $(\wedge)$. A skew symmetric matrix operator is defined as $S_{\{3\times3\}} = [\boldsymbol{r}\times]$ where $\boldsymbol{r}_{\{3\times1\}}$ is a vector (Farrell, 2008, Eq. B.15). Lastly, the symbol '$\circ$' means entry-wise product.

A trajectory generator is needed to provide the following true input data to the simulator,

$$\boldsymbol{T} = [\boldsymbol{e},\, \boldsymbol{p},\, \boldsymbol{v}^n,\, \boldsymbol{a}^n,\, \boldsymbol{t}_S]\,, \tag{1}$$

where vectors $\boldsymbol{e} = \{\boldsymbol{e}_i\}_{i=1}^{k_S}$, $\boldsymbol{p} = \{\boldsymbol{p}_i\}_{i=1}^{k_S}$, $\boldsymbol{v}^n = \{\boldsymbol{v}_i^n\}_{i=1}^{k_S}$, $\boldsymbol{a}^n = \{\boldsymbol{a}_i^n\}_{i=1}^{k_S}$, and $\boldsymbol{t}_S = \{\boldsymbol{t}_i\}_{i=1}^{k_S}$ are discrete-time sequences and have a number of elements equal to k_S.

True attitude vector $\boldsymbol{e} = [\phi, \theta, \psi]$ contains row, pitch, and yaw angles, respectively. $\boldsymbol{p}$ is the true position vector mechanized in local North-East-Down (NED) coordinate system (Cai et al., 2011, Sec. 2.2.3). $\boldsymbol{v}^n$ and $\boldsymbol{a}^n$ are true velocity and acceleration vectors, respectively, both mechanized in the vehicle-carried NED coordinate system (Cai et al., 2011, Sec. 2.2.4). Finally, $\boldsymbol{t}_S$ is the simulation time vector.

2.1 Gyroscopes

Outputs from three ideal orthogonal gyroscopes are expressed as (Titterton and Weston, 2004, Eq. 3.29),

$$\boldsymbol{\omega}_{ib}^b = \boldsymbol{\omega}_{nb}^b + C_n^b\left(\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n\right)\,. \tag{2}$$

Vector $\boldsymbol{\omega}_{nb}^b = [\omega_x, \omega_y, \omega_z]^T$ represents the turn rate of the body frame (b-frame) with respect to the navigation frame (n-frame) expressed in body axes, i.e., the orientation of the vehicle respect to the b-frame.

Since $\boldsymbol{\omega}_{ie}^n$, $\boldsymbol{\omega}_{en}^n$, and C_n^b can be obtained from (Titterton and Weston, 2004, Eq. 3.72, 3.74, and 3.63), respectively, it can be seen from (2) that $\boldsymbol{\omega}_{nb}^b$ is the only missing term to generate gyroscopes outputs. $\boldsymbol{\omega}_{nb}^b$ can be obtained doing some mathematical manipulations.

Matrix C_b^n can be updated at each simulation time step doing (Titterton and Weston, 2004, Eq. 3.36–3.38),

$$C_{b(+)}^n = C_b^n(I + \delta\Psi) \tag{3a}$$

$$\delta\Psi = [\delta\boldsymbol{e}\times]\,. \tag{3b}$$

$\delta\boldsymbol{e}_{\{3\times1\}}$ is a vector that contains incremental values of true Euler angles, projected along the b-frame, between simulation time steps $t_{(k)}$ and $t_{(k+1)}$. In the limit, and dividing $\delta\Psi$ by δt, the simulation time step (Titterton and Weston, 2004, Eq. 3.40),

$$\lim_{\delta t\to 0}\frac{\delta\Psi}{\delta t} = \Omega_{nb}^b\,, \tag{4}$$

where $\Omega_{nb}^b = [\boldsymbol{\omega}_{nb}^b\times]$. Therefore, in order to obtain $\boldsymbol{\omega}_{nb}^b$, $\delta\Psi$ must be clear in (3a). Pre multiplying both sides of (3a) by C_b^{nT}, and recalling that C_b^n is an orthogonal matrix and as result $C_b^{nT}\,C_b^n = I$, yields,

$$C_b^{nT}C_{b(+)}^n = \overbrace{C_b^{nT}C_b^n}^{I}(I + \delta\Psi) \tag{5a}$$

$$\implies \delta\Psi = C_b^{nT}C_{b(+)}^n - I \tag{5b}$$

$$= C_n^b C_{n(+)}^{bT} - I\,. \tag{5c}$$

C_n^b can be computed using $\boldsymbol{e}$ (Titterton and Weston, 2004, Eq. 3.47).

Vector $\boldsymbol{\omega}_{nb}^{b}$ is set up by taking positive elements from $\delta\Psi$ (3b) and dividing them by δt,

$$\boldsymbol{\omega}_{nb}^{b} = \left[\frac{\delta\Psi_{(3,2)}}{\delta t},\ \frac{\delta\Psi_{(1,3)}}{\delta t},\ \frac{\delta\Psi_{(2,1)}}{\delta t} \right]^{T} . \tag{6}$$

It is important to bear in mind that, according to (4), $\boldsymbol{\omega}_{nb}^{b}$ is an approximation to the theoretical value. Hence, a smaller δt will lead to more accurate values of $\boldsymbol{\omega}_{nb}^{b}$.

A gyroscope presents several sources of error, for instance, angle random walk noise (ARW), rate random walk noise, rate ramp noise, biases, cross-coupling errors, scale factor errors, and axes misalignments, among others. In this work, only three sources of error are taken into account, ARW and static and dynamic biases. According to the authors' experience, they have the strongest influence on a MEMS IMU performance.

ARW from three gyroscopes can be expressed as Gaussian white-noise sequences $\boldsymbol{\eta}_g \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\sigma}_g^2)$, where $\boldsymbol{\sigma}_g^2 = [\sigma_{gx}^2, \sigma_{gy}^2, \sigma_{gz}^2]^T$ (Farrell, 2008, Sec. 4.6.3.2). Static bias $\boldsymbol{b}_g$ varies every time the gyroscope is turned on, but stays constant throughout operation (Groves, 2008, Sec. 4.4.1). Hence, it is modelled as a random constant process and is simulated by adding a constant value to $\boldsymbol{\omega}_{ib}^{b}$ (Farrell, 2008, Sec. 4.6.3.1). This number is taken randomly from the interval $[-b_g,\, b_g]$. On the other hand, dynamic bias $\delta\boldsymbol{b}_g$ is usually observed at low frequencies and is associated with a correlation time $\boldsymbol{\tau}_g$. It is modelled as a scalar Gauss-Markov process (Farrell, 2008, Sec. 4.6.3.3), whose discrete formulation is given by (Strus et al., 2008),

$$\boldsymbol{\eta}_{g\delta b(+)} = \boldsymbol{\alpha}\, \boldsymbol{\eta}_{g\delta b} + \boldsymbol{w_c} \tag{7a}$$

$$\boldsymbol{\alpha} = e^{-(\delta t \,\circ\, 1/\boldsymbol{\tau}_g)} \tag{7b}$$

$$\boldsymbol{\sigma}_c^2 = \boldsymbol{\sigma}_{g\delta b}^2 \circ (1 - \boldsymbol{\alpha}^2) . \tag{7c}$$

$\boldsymbol{\eta}_{g\delta b}$ is obtained in an iterative fashion with $\boldsymbol{\eta}_{g\delta b(1)} = \boldsymbol{b}_g$. $\boldsymbol{w}_c$ is a normal sequence with variance $\boldsymbol{\sigma}_c^2$. If $\boldsymbol{\tau}_g$ is unknown, $\boldsymbol{\eta}_{g\delta b}$ should be taken as uncorrelated Gaussian white noise. If variance $\boldsymbol{\sigma}_{g\delta b}^2$ is not provided by the manufacturer, typically could be considered about 10 percent of the static bias (Groves, 2008, Sec. 4.4.1).

Finally, simulated noisy outputs from three orthogonal gyroscopes are,

$$\tilde{\boldsymbol{\omega}}^{b} = \boldsymbol{\omega}_{ib}^{b} + \boldsymbol{\eta}_g + \boldsymbol{b}_g + \boldsymbol{\eta}_{g\delta b} . \tag{8}$$

Usually, gyroscopes manufacturers deliver accuracy specification ARW in units of root PSD. Thus, a conversion of units is needed for SI unit consistency. If ARW is in $\left(\frac{\mathrm{deg}}{\sqrt{\mathrm{h}}}\right)$, then (Farrell, 2008, Sec. 4.6.3.2),

$$\boldsymbol{n}_g = \frac{\mathrm{ARW}}{60} \frac{\pi}{180} \left(\frac{\mathrm{rad/s}}{\sqrt{\mathrm{Hz}}} \right) . \tag{9}$$

Equation (30) is used to get $\boldsymbol{\sigma}_g^2$.

2.2 Accelerometers

Simulated measurements coming from three orthogonal accelerometers are easier to generate when compared with gyroscopes. In the first place, vector $\boldsymbol{a}$ from (1) has to be projected from the vehicle-carried NED frame into the b-frame,

$$\boldsymbol{a}^{b} = C_n^b\, \boldsymbol{a}^{n} . \tag{10}$$

Accelerometers outputs have spurious values from a navigation solution perspective. These distortions are gravity and a Coriolis fictitious force. Gravity vector $\boldsymbol{g}^n$ is obtained from (Titterton and Weston, 2004, Eq. 3.89–3.91). Formula to calculate the Coriolis fictitious force is (Titterton and Weston, 2004, Eq. 3.27),

$$\boldsymbol{f}_{cor}^{n} = S\ (2\ \boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{en}^{n}) , \tag{11}$$

where $S = [\boldsymbol{v}^n \times]$. Gravity and Coriolis vectors must be projected along the b-frame.

Hence, outputs from three orthogonal accelerometers are,

$$\boldsymbol{f}^{b} = \boldsymbol{a}^{b} - C_n^b\, (\boldsymbol{f}_{cor}^{n} + \boldsymbol{g}^{n}) . \tag{12}$$

As with gyroscopes, real accelerometers also present several sources of noise. The proposed error model of accelerometers is similar to the error model of gyroscopes, presented in (8). Lastly, noisy simulated outputs of accelerometers are,

$$\tilde{\boldsymbol{f}}^{b} = \boldsymbol{f}^{b} + \boldsymbol{\eta}_f + \boldsymbol{b}_f + \boldsymbol{\eta}_{f\delta b} , \tag{13}$$

where $\boldsymbol{\eta}_f \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\sigma}_f^2)$ with variance $\boldsymbol{\sigma}_f^2 = [\sigma_{fx}^2, \sigma_{fy}^2, \sigma_{fz}^2]^T$. $\boldsymbol{b}_f$ denotes the static bias and $\boldsymbol{\eta}_{f\delta b}$ indicates the dynamic bias noise with correlation time $\boldsymbol{\tau}_f$ and variance $\boldsymbol{\sigma}_{f\delta b}^2$, as shown in (7).

Generally, accuracy specification of accelerometers is given as velocity random walk (VRW). If VRW is in $\left(\frac{\mathrm{m/s}}{\sqrt{\mathrm{h}}}\right)$, then,

$$\boldsymbol{n}_f = \frac{\mathrm{VRW}}{60} \left(\frac{\mathrm{m/s^2}}{\sqrt{\mathrm{Hz}}} \right) . \tag{14}$$

Equation (30) is used to get values of variance $\boldsymbol{\sigma}_f^2$.

2.3 GPS receiver

A low-cost GPS receiver usually delivers estimates of position as stated in the World Geodetic System 1984 Earth model (WGS84). Additionally, it may also provide estimates of velocity in the vehicle-carried NED frame. Table 1 shows some important constants from the WGS84 Earth model that will be used later.

Most low-cost GPS receivers give estimates at lower frequencies than an IMU. They usually work at frequencies between 1 Hz and 20 Hz. Thus, it is necessary to reduce the vectors' sampling rates from (1) (downsampling). These

Table 1. Constants from WGS84 Earth model.

Semi-major axis	a	6378137.0	m
Semi-minor axis	b	6356752.3142	m

new true vectors will have k_G number of elements, where $k_G < k_S$.

GPS position Position $\boldsymbol{p}$ from (1) must be converted to true GPS position $\boldsymbol{p}^n = [\gamma, \lambda, h]^T$, where γ denotes geodetic latitude, λ denotes longitude, both in radians, and h denotes altitude in meters above the reference ellipsoid (Farrell, 2008, Sec. 2.3.2.1). True GPS position is generated in two steps. First, position in the local NED coordinate system is expressed in ECEF coordinates, $\boldsymbol{p}^e = [x^e, y^e, z^e]^T$. Then, ECEF position is transformed into $\boldsymbol{p}^n$.

Formulas to convert coordinates from local NED to ECEF are (Cai et al., 2011, Eq. 2.23),

$$\boldsymbol{p}^e = \boldsymbol{p}_o^e + R^e\, \boldsymbol{p} \tag{15a}$$

$$R^e = \begin{bmatrix} -\sin\gamma_o \cos\lambda_o & -\sin\lambda_o & -\cos\gamma_o \cos\lambda_o \\ -\sin\gamma_o \sin\lambda_o & \cos\lambda_o & -\cos\gamma_o \sin\lambda_o \\ \cos\gamma_o & 0 & -\sin\gamma_o \end{bmatrix}. \tag{15b}$$

The local reference point $\boldsymbol{p}_o^n = [\gamma_o, \lambda_o, h_o]^T$ is the point where the trajectory begins, and $\boldsymbol{p}_o^e = [x_o^e, y_o^e, z_o^e]^T$ is the local reference point in ECEF coordinates, which can be found as follows (Farrell, 2008, Eq. 2.9–2.11),

$$x_o^e = (R_N(\gamma_o) + h_o) \cos\gamma_o \cos\lambda_o \tag{16a}$$

$$y_o^e = (R_N(\gamma_o) + h_o) \cos\gamma_o \sin\lambda_o \tag{16b}$$

$$z_o^e = \left[R_N(\gamma_o)\left(1 - e^2\right) + h_o\right] \sin\gamma_o\,, \tag{16c}$$

where R_N is provided by (Farrell, 2008, Eq. 2.7).

Next, intermediate calculations to transform from ECEF position to GPS position (Vermeille, 2002, Sec. 3) are,

$$p = \left(x^{e2} + y^{e2}\right) / \; a^2 \tag{17a}$$

$$q = z^{e2}\left(1 - e^2\right) / \; a^2 \tag{17b}$$

$$r = \left(p + q - e^4\right) / \; 6 \tag{17c}$$

$$s = \left(e^4\, p\, q\right) / \left(4\, r^3\right) \tag{17d}$$

$$t = \sqrt[3]{1 + s + \sqrt{s(2+s)}} \tag{17e}$$

$$u = r\left(1 + t + \frac{1}{t}\right) \tag{17f}$$

$$v = \sqrt{u^2 + e^4\, q} \tag{17g}$$

$$w = e^2\,(u + v - q) / \; (2\, v) \tag{17h}$$

$$c = \sqrt{u + v + w^2} - w \tag{17i}$$

$$E = c\sqrt{x^{e2} + y^{e2}} / \left(c + e^2\right)\,, \tag{17j}$$

where constants a and e are taken from Table 1.

Finally, true GPS position $\boldsymbol{p}^n = [\gamma, \lambda, h]^T$ is expressed as (Vermeille, 2002, Sec. 3),

$$\gamma = \arctan2\,(z^e, E) \tag{18a}$$

$$\lambda = \arctan2\,(y^e, x^e) \tag{18b}$$

$$h = \frac{c + e^2 - 1}{c}\sqrt{E^2 + z^{e2}}\,. \tag{18c}$$

A GPS receiver delivers estimates of position with certain dispersion. Consequently, noise must be added to $\boldsymbol{p}^n$. Usually, GPS manufacturers give horizontal (2D) position accuracy expressed as circular error probable (CEP) in meters. CEP is the radius of the circle centred on the correct location that contains 50% of the expected horizontal position errors (Farrell, 2008, Sec. 4.9.1.2). If latitude and longitude standard deviations, $\sigma_{\gamma m}$ and $\sigma_{\lambda m}$ respectively, are considered to be equal, then (Farrell, 2008, Table 4.2),

$$\sigma_{\gamma\, m} = \sigma_{\lambda\, m} = 0.8493 \text{ CEP (m)}\,. \tag{19}$$

GPS standard deviations are also needed in units of radians and can be obtained as follows,

$$\sigma_\gamma = \frac{\sigma_{\gamma m}}{R_M(\gamma_o) + h_o} \tag{20a}$$

$$\sigma_\lambda = \frac{\sigma_{\lambda m}}{(R_N(\gamma_o) + h_o)\cos\gamma_o}\,, \tag{20b}$$

where R_M is given by (Farrell, 2008, Eq. 2.6).

Lastly, to find GPS position estimates $\tilde{\boldsymbol{p}}_G^n = [\tilde{\gamma}_G, \tilde{\lambda}_G, \tilde{h}_G]^T$, $\boldsymbol{p}^n$ is added with Gaussian white noises $\eta_\gamma \sim \mathrm{N}(0, \sigma_\gamma^2)$, $\eta_\lambda \sim \mathrm{N}(0, \sigma_\lambda^2)$, and $\eta_h \sim \mathrm{N}(0, \sigma_h^2)$,

$$\tilde{\gamma}_G = \gamma + \eta_\gamma \tag{21a}$$

$$\tilde{\lambda}_G = \lambda + \eta_\lambda \tag{21b}$$

$$\tilde{h}_G = h + \eta_h\,. \tag{21c}$$

GPS velocity GPS velocity $\tilde{\boldsymbol{v}}_G^n = [\tilde{v}_{N_G}, \tilde{v}_{E_G}, \tilde{v}_{D_G}]^T$ is straightforwardly generated. Vector $\boldsymbol{v}^n$ from (1) is added with Gaussian white noise $\boldsymbol{\eta}_v \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\sigma}_v^2)$, where $\boldsymbol{\sigma}_v$ is given by the GPS manufacturer,

$$\tilde{\boldsymbol{v}}_G^n = \boldsymbol{v}^n + \boldsymbol{\eta}_v\,. \tag{22}$$

3. EXTENDED KALMAN FILTER

The Kalman filter is an algorithm for linear systems that operates recursively on noisy input and output data to produce a statistically optimal estimate of the system states. The extended Kalman filter (EKF) is a nonlinear extension of the Kalman filter, which linearises about an estimate of the current mean and covariance. The EKF can be formulated to fuse information coming from inertial and aiding sensors to get state estimates with less errors when compared with sensors used separately. In doing so, NaveGo implements a version of the EKF called complementary filter (Farrell, 2008, Sec. 4.10).

The SINS error model is obtained by perturbing the navigation equations, and is given by a series of first order differential equations. The resulting system is linear although time-variant. Fig. 1 exposes how SINS, GPS and

EKF work together. This kind of integration is known as feedback implementation (Farrell, 2008, Sec. 5.10.5.3), where corrections in $\hat{\boldsymbol{x}}$ are fed back and used it as prior information to calculate estimates at the SINS.

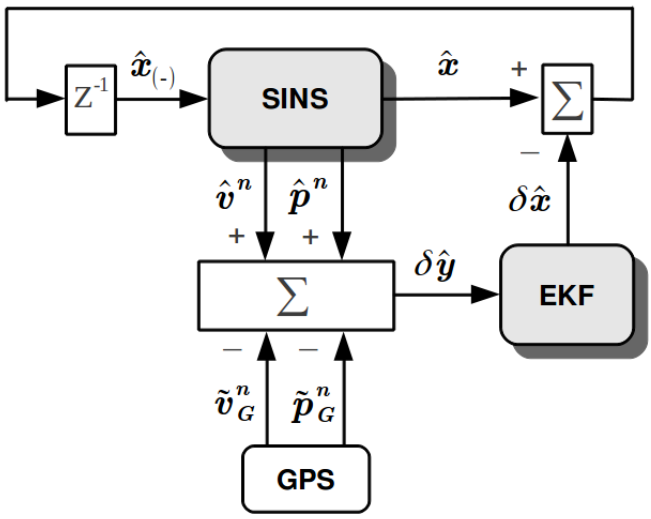

Fig. 1. Diagram of SINS, GPS, and EKF integration.

The continuous state-space model of the system in time domain is (Brown and Hwang, 1997, Sec. 7.1),

$$\delta\dot{\hat{\boldsymbol{x}}}_{(t)} = F_{(t)}\,\delta\hat{\boldsymbol{x}}_{(t)} + G_{(t)}\,\boldsymbol{u}_{(t)} + \boldsymbol{\zeta}_{(t)} \tag{23a}$$
$$\delta\hat{\boldsymbol{y}}_{(t)} = H\,\delta\hat{\boldsymbol{x}}_{(t)} + \boldsymbol{\nu}_{(t)}\,. \tag{23b}$$

The discrete-time, state-space model of the system is (Brown and Hwang, 1997, Sec. 5.5),

$$\delta\hat{\boldsymbol{x}}_{(+)} = \Phi\,\delta\hat{\boldsymbol{x}} + G\,\boldsymbol{u} + \boldsymbol{\zeta} \tag{24a}$$
$$\delta\hat{\boldsymbol{y}} = H\,\delta\hat{\boldsymbol{x}} + \boldsymbol{\nu}\,. \tag{24b}$$

Vectors $\boldsymbol{\zeta} \sim \mathrm{N}(\mathbf{0}, Q)$ and $\boldsymbol{\nu} \sim \mathrm{N}(\mathbf{0}, R)$ are known as driving noise and measurement noise, respectively. Vectors $\boldsymbol{u} \in \mathbb{R}^{12}$, $\delta\hat{\boldsymbol{x}} \in \mathbb{R}^{21}$, and $\delta\hat{\boldsymbol{y}} \in \mathbb{R}^{6}$ are defined as,

$$\boldsymbol{u} = \left[\tilde{\boldsymbol{\omega}}^{b\,T},\ \tilde{\boldsymbol{f}}^{b\,T},\ \boldsymbol{\eta}_{g\delta b}^{T},\ \boldsymbol{\eta}_{f\delta b}^{T}\right]^{T} \tag{25a}$$
$$\delta\hat{\boldsymbol{x}} = \left[\delta\hat{\boldsymbol{e}}^{T},\ \delta\hat{\boldsymbol{v}}^{nT},\ \delta\hat{\boldsymbol{p}}^{nT},\ \hat{\boldsymbol{b}}_{g}^{T},\ \hat{\boldsymbol{b}}_{f}^{T},\ \delta\hat{\boldsymbol{b}}_{g}^{T},\ \delta\hat{\boldsymbol{b}}_{f}^{T}\right]^{T} \tag{25b}$$
$$\delta\hat{\boldsymbol{y}} = \left[\delta\hat{\boldsymbol{y}}_{v}^{T},\ \delta\hat{\boldsymbol{y}}_{p}^{T}\right]^{T} \tag{25c}$$
$$\delta\hat{\boldsymbol{y}}_{v} = \left[\hat{\boldsymbol{v}}^{n} - \tilde{\boldsymbol{v}}_{G}^{n}\right] \tag{25d}$$
$$\delta\hat{\boldsymbol{y}}_{p} = \hat{T}_{p}^{r}\left(\hat{\boldsymbol{p}}^{n} - \tilde{\boldsymbol{p}}_{G}^{n}\right) + \hat{C}_{b}^{n}\,\boldsymbol{l}_{ba}^{b} \tag{25e}$$
$$\hat{T}_{p}^{r} = \mathrm{diag}\left(\left[(\hat{R}_{M} + \hat{h}), (\hat{R}_{N} + \hat{h})\cos(\hat{\gamma}), -1\right]\right), \tag{25f}$$

where $\boldsymbol{l}_{ba}^{b}$ is the lever arm from the GPS antenna to the IMU. $\hat{T}_{p}^{r}$ is the curvilinear-to-Cartesian transformation matrix (Groves, 2008, Eq. 12.81).

If $I_{\{3\times3\}}$ is an identity matrix and $\mathbf{0}_{\{3\times3\}}$ is a null matrix, state-space matrices are,

$$F_{(t)\{21\times21\}} = \begin{bmatrix} F_{ee} & F_{ev} & F_{ep} & -\hat{C}_{b}^{n} & \mathbf{0} & -\hat{C}_{b}^{n} & \mathbf{0} \\ F_{ve} & F_{vv} & F_{vp} & \mathbf{0} & \hat{C}_{b}^{n} & \mathbf{0} & \hat{C}_{b}^{n} \\ \mathbf{0} & F_{pv} & F_{pp} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\dfrac{1}{\boldsymbol{\tau}_{g}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\dfrac{1}{\boldsymbol{\tau}_{f}} \end{bmatrix} \tag{26}$$

$$G_{\{21\times12\}} = \begin{bmatrix} -\hat{C}_{b}^{n} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \hat{C}_{b}^{n} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{bmatrix} \tag{27}$$

$$H_{\{6\times21\}} = \begin{bmatrix} \mathbf{0} & I & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \hat{T}_{p}^{r} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{28}$$

Submatrices F_{nn} from (26) are shown in (Titterton and Weston, 2004, Eq. 12.28).

Covariance matrix Q (Groves, 2008, Eq. 12.67) is defined as,

$$Q_{\{12\times12\}} = \mathrm{diag}\left(\left[\boldsymbol{n}_{g}^{2\,T},\ \boldsymbol{n}_{f}^{2\,T},\ \boldsymbol{n}_{g\delta b}^{2\,T},\ \boldsymbol{n}_{f\delta b}^{2\,T}\right]\right), \tag{29}$$

where $\boldsymbol{n}_{g}^{2}$ (9), $\boldsymbol{n}_{f}^{2}$ (14), $\boldsymbol{n}_{g\delta b}^{2}$, and $\boldsymbol{n}_{f\delta b}^{2}$ (7) are the power spectral densities (PSD) of, respectively, the gyroscopes random noise, accelerometers random noise, gyroscopes dynamic bias, and accelerometers dynamic bias. These parameters can be obtained as (Groves, 2008, Eq. 12.68 and 12.69),

$$\boldsymbol{n}_{g}^{2} = \boldsymbol{\sigma}_{g}^{2}\,\delta t \tag{30a}$$
$$\boldsymbol{n}_{f}^{2} = \boldsymbol{\sigma}_{f}^{2}\,\delta t \tag{30b}$$
$$\boldsymbol{n}_{g\delta b}^{2} = \boldsymbol{\sigma}_{g\delta b}^{2} \circ \boldsymbol{\tau}_{g} \tag{30c}$$
$$\boldsymbol{n}_{f\delta b}^{2} = \boldsymbol{\sigma}_{f\delta b}^{2} \circ \boldsymbol{\tau}_{f}\,. \tag{30d}$$

Covariance matrix R is defined as,

$$R_{\{6\times6\}} = \mathrm{diag}\left(\left[\boldsymbol{\sigma}_{v}^{2\,T},\ \sigma_{\gamma m}^{2},\ \sigma_{\lambda m}^{2},\ \sigma_{h}^{2}\right]\right). \tag{31}$$

Diagonal entries from R are provided by the GPS error profile. Matrix $\hat{T}_{p}^{r}$ in (25e) motivates the use of $\sigma_{\gamma m}$ and $\sigma_{\lambda m}$ in R, not σ_{γ} and σ_{λ}. This way, diagonal elements from R are of the same order, which produces better numerical stability in calculating the Kalman gain, K (Alg. 1, line 4).

Algorithm 1 presents a modified version of the EKF algorithm for one iteration. At each iteration, the state prediction equation reduces to $\delta\hat{\boldsymbol{x}}^{-} = \mathbf{0}$, which is the optimal predicted error estimate. Hence, the original equation to update the actual state estimates, $\delta\hat{\boldsymbol{x}} = \delta\hat{\boldsymbol{x}}^{-} + K(\delta\hat{\boldsymbol{y}} - \delta\hat{\boldsymbol{x}}^{-})$, is simplified as shown in line 5 of Alg. 1 (Farrell, 2008, Eq. 5.103).

Since the master clock of an INS is the GPS clock, the EKF is run only at GPS time steps, t_{G}.

Algorithm 1 Simplified extended Kalman filter.

1: Update $\delta t_{_G} = t_{_G} - t_{_{G(-)}}$, $\delta\hat{\boldsymbol{y}}$ (25c), F (26), and G (27).
2: $\Phi \quad = I + F\,\delta t_{_G}$
3: $Q_k \quad = G\;Q\;G^T \delta t_{_G}$
4: $K \quad = \left(P_{(-)}\;H^T\right)\left(R + H\;P_{(-)}\;H^T\right)^{-1}$
5: $\delta\hat{\boldsymbol{x}} \quad = K\delta\hat{\boldsymbol{y}}$
6: $P_t \quad = (I - K\;H)\;\;P$
7: $P \quad = \Phi\;P_t\;\Phi^T + Q_k$
8: $P \quad = \frac{1}{2}\left(P + P^T\right)$

The covariance matrix $P_{\{21\times 21\}}$ must be set up initially before the first execution of Alg. 1. It is formed as a diagonal matrix whose entries are chosen according to the expected initial variances of $\delta\hat{\boldsymbol{x}}$ (25b) as,

$$P_{(1)} = \text{diag}\left([\sigma_\phi^2,\, \sigma_\theta^2,\, \sigma_\psi^2,\, \boldsymbol{\sigma}_v^{2\,T},\, \sigma_\gamma^2,\, \sigma_\lambda^2,\, \sigma_h^2,\, \boldsymbol{b}_g^{2\,T},\, \boldsymbol{b}_f^{2\,T},\, \boldsymbol{\sigma}_{g\delta b}^{2\,T},\, \boldsymbol{\sigma}_{g\delta f}^{2\,T}]\right) . \tag{32}$$

Kalman filter may have serious numerical problems (Grewal and Andrews, 2008, Ch. 6). To avoid this situation, it is necessary but not sufficient that covariance matrix P be symmetric at each iteration. Line 8 of Alg. 1 ensures this condition (Farrell, 2008, Sec. 5.9).

4. NAVEGO MAIN ALGORITHM

NaveGo is a simulation framework for low-cost integrated navigation systems. It works with variables in units of the International System of Units (SI). NaveGo takes data generated by a trajectory generator, vector $\boldsymbol{T}$ (1), and produces simulated measurements delivered by gyroscopes, accelerometers, and a GPS receiver. Also, it can process data from real sensors.

The mathematical model of NaveGo is based upon the reference model shown in (Gonzalez et al., 2015). NaveGo implements a loosely-couple INS to combine data from inertial sensors and aiding sensors into a more accurate estimate of attitude, velocity, and position. A loosely-coupled system is chosen between other types of integration since it can be implemented with most low-cost GPS receivers.

NaveGo is developed in MATLAB for being the facto standard programming language for simulation and mathematical computing. Nevertheless, NaveGo can run with FreeMat (FreeMat, 2014), an open-source, mathematical computing tool, syntax-compatible with MATLAB. NaveGo is developed as an open-source toolbox (Navego, 2015). Two type of floating-point precision are supported: double and single. Precision is set by the argument 'precision' supported by some NaveGo's functions (Sec. 7). The default value is double.

NaveGo considers that discrete-time series $\tilde{\boldsymbol{\omega}}^b$ and $\tilde{\boldsymbol{f}}^b$ are synchronized, i.e., they are associated with the same vector time $t_{_S}$, which has a number of elements equal to $k_{_S}$. On the other hand, GPS discrete-time series $\tilde{\boldsymbol{p}}_{_G}^n$ and $\tilde{\boldsymbol{v}}_{_G}^n$ are synchronized with $t_{_G}$, which has a number of elements equal to $k_{_G}$. Each sensor's measurements and error profile are put together in a data structure, so they can be accessed efficiently.

Algorithm 2 details the operation of NaveGo. State vector $\hat{\boldsymbol{x}}$ is defined as,

$$\hat{\boldsymbol{x}} = [\hat{\boldsymbol{e}}^T,\; \hat{\boldsymbol{v}}^{nT},\; \hat{\boldsymbol{p}}^{nT},\; \hat{\boldsymbol{b}}_g^T,\; \hat{\boldsymbol{b}}_f^T,\; \delta\hat{\boldsymbol{b}}_g^T,\; \delta\hat{\boldsymbol{b}}_f^T]^T . \tag{33}$$

Algorithm 2 NaveGo main algorithm.

1: Set $\boldsymbol{\sigma}_g^2$, $\boldsymbol{\sigma}_f^2$, $\boldsymbol{b}_g$, $\boldsymbol{b}_f$, $\boldsymbol{\sigma}_{g\delta b}^2$, $\boldsymbol{\sigma}_{f\delta b}^2$, $\boldsymbol{\tau}_g$, and $\boldsymbol{\tau}_f$
2: Set $\boldsymbol{p}_o^n$, $\sigma_{\gamma m}^2$, $\sigma_{\lambda m}^2$, σ_γ^2, σ_λ^2, σ_h^2, and $\boldsymbol{\sigma}_v^2$
3: Set $\boldsymbol{n}_g$, $\boldsymbol{n}_f$, $\boldsymbol{n}_{g\delta b}$, and $\boldsymbol{n}_{f\delta b}$
4: Generate $\tilde{\boldsymbol{\omega}}^b$ (8) and $\tilde{\boldsymbol{f}}^b$ (13) until $k_{_S}$, or load real IMU data set
5: Downsample $\boldsymbol{T}$ (1) and generate $\tilde{\boldsymbol{p}}_{_G}^n$ (21) and $\tilde{\boldsymbol{v}}_{_G}^n$ (22) until $k_{_G}$, or load real GPS data set
6: Set $\hat{\boldsymbol{e}}_{(1)} = [\hat{\phi}_o, \hat{\theta}_o, \hat{\psi}_o]$
7: Calculate $\hat{\boldsymbol{q}}_{(1)}$ with $\hat{\boldsymbol{e}}_{(1)}$ (Titterton and Weston, 2004, Eq. 3.65)
8: Calculate $\hat{C}_{n(1)}^b$ with $\hat{\boldsymbol{q}}_{(1)}$ (Titterton and Weston, 2004, Eq. 3.63)
9: Set $\hat{\boldsymbol{p}}_{(1)}^n = \tilde{\boldsymbol{p}}_{_{G(1)}}$ and $\hat{\boldsymbol{v}}_{(1)}^n = \tilde{\boldsymbol{v}}_{_{G(1)}}$
10: Set $\hat{\boldsymbol{b}}_g = \boldsymbol{0}$, $\hat{\boldsymbol{b}}_f = \boldsymbol{0}$, $\delta\hat{\boldsymbol{b}}_g = \boldsymbol{0}$, and $\delta\hat{\boldsymbol{b}}_f = \boldsymbol{0}$
11: Set $\boldsymbol{l}_{ba}^b$, Q (29), R (31), and $P_{(1)}$ (32)
12: Guarantee that $t_{_{G(1)}} < t_{_{S(1)}} =< t_{_{G(2)}}$
13: Guarantee that $t_{_{S(k_S-1)}} < t_{_{G(k_G)}} =< t_{_{S(k_S)}}$
14: Set $i = 2$
15: **for** $k = 2$ to $k_{_G}$ **do**
16: **while** $t_{_{S(i)}} =< t_{_{G(k)}}$ **do**
17: Update $\hat{\boldsymbol{\omega}}_{ie}^n$ and $\hat{\boldsymbol{\omega}}_{en}^n$ (Titterton and Weston, 2004, Eq. 3.72 and 3.74)
18: Update $\hat{\boldsymbol{q}}$ (Crassidis and Junkins, 2011, Eq. 7.39)
19: Update $\hat{C}_b^n$ (Titterton and Weston, 2004, Eq. 3.63)
20: Update $\hat{\boldsymbol{e}}$ (Titterton and Weston, 2004, Eq. 3.66)
21: Update $\hat{\boldsymbol{g}}^n$ (Titterton and Weston, 2004, Eq. 3.89–3.91)
22: Update $\hat{\boldsymbol{v}}^n$ (Titterton and Weston, 2004, Eq. 3.69)
23: Update $\hat{\boldsymbol{p}}^n$ (Titterton and Weston, 2004, Eq. 3.79–3.81)
24: $i = i + 1$
25: **end while**
26: Update $\delta\hat{\boldsymbol{x}}$ (Alg. 1)
27: Correct $\hat{\boldsymbol{x}}$: $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \delta\hat{\boldsymbol{x}}$
28: Correct $\hat{\boldsymbol{q}}$ (34)
29: Correct $\hat{C}_b^n$ (3)
30: **end for**

Formulas for correction of quaternions are (Crassidis and Junkins, 2011, Eq. 7.34, A.172a, and A.173),

$$\hat{\boldsymbol{q}} = \hat{\boldsymbol{q}} + \frac{1}{2}\,\Xi(\hat{\boldsymbol{q}})\,\delta\hat{\boldsymbol{e}} \tag{34a}$$

$$\Xi(\hat{\boldsymbol{q}})_{\{4\times 3\}} = \begin{bmatrix} q_4\, I_{\{3\times 3\}} + [\boldsymbol{\varrho}\times] \\ -\boldsymbol{\varrho}^T \end{bmatrix} , \tag{34b}$$

where $\boldsymbol{\varrho}$ comes from the definition of quaternion,

$$\boldsymbol{q} = [\ \boldsymbol{\varrho},\ q_4\ ]^T = [q_1,\ q_2,\ q_3,\ q_4]^T\ , \tag{35}$$

and $\delta\hat{\boldsymbol{e}}$ is delivered by the EKF (25b).

The master clock of NaveGo is the GPS clock. Therefore, the EKF is only executed when new GPS data are available. Between GPS measurements, the SINS keeps updating estimates of attitude, velocity, and position.

Attitude may be initialized with initial true angles or, a more realistic approach, with estimated angles at rest. At this point, one may force large initial attitude errors and test how the system recovers. Position and velocity are initialized with corresponding GPS estimates.

When a correlation time is not specified, it must be set to inf. This way, NaveGo will process corresponding noise $\boldsymbol{\eta}_{\delta b}$ as uncorrelated Gaussian white noise.

NaveGo has certain limitations. Neither corrections in computation of attitude are made for conning effects, nor velocity is corrected for sculling effects. In addition, GPS position and velocity are derived under the assumption that they are only affected by a zero-mean Gaussian distributions. In particular cases, as when impulse noise interference is relevant, GPS noise distribution cannot be considered as Gaussian (Liu and Amin, 2008).

5. SIMULATION RESULTS

In order to expose some simulation results, it is proposed to evaluate the performances from two INS compound of different IMU.

A real-world data set is used as input data to NaveGo. In (Toth et al., 2011), IMU of several grades and a Novatel OEM4 dual-frequency GPS receiver were mounted on a vehicle. Measurements from these sensors were logged during an open-sky trajectory while the vehicle was driven. In addition, a DGPS system was implemented with a Topcon Legacy GPS base station. A reference data set $\boldsymbol{T_R} = [\boldsymbol{e}, \boldsymbol{v}^n, \boldsymbol{p}^n, \boldsymbol{t}_S]$ is obtained by fusing navigation-grade IMU Honeywell H764G-1 and DGPS measurements.

Fig. 2 displays the trajectory that $\boldsymbol{p}^n$ from $\boldsymbol{T_R}$ describes. The trajectory takes place in the vicinity of the Ohio University, USA. The globe on the left with a circle marks the beginning of the trajectory. On the other hand, the globe on the right with a square points the end of the trajectory. As seen from Fig. 2, the trajectory contains high and low dynamics. First, the vehicle performs several curves on a parking. Then, it is driven along a road. The path covers 14.67 minutes throughout which the availability of GPS signal is 100%.

To simulate accelerometers, vector $\boldsymbol{a}^n$ is produced by taking first order derivative of $\boldsymbol{v}^n$. A Savitzky-Golay FIR smoothing filter is applied to $\boldsymbol{a}^n$ to eliminate the noise produced by computing the derivative.

Noise characteristics of GPS, gyroscopes and accelerometers are based upon parameters taken from real sensors. Chosen inertial sensors are Analog Devices IMU ADIS16405 (Analog Devices, 2009a) and ADIS16488 (Analog Devices, 2009b). The GPS receiver to simulate is the GPS 18-5Hz from Garmin (Garmin International, 2005). Tables 2 and 3 summarize error parameters from

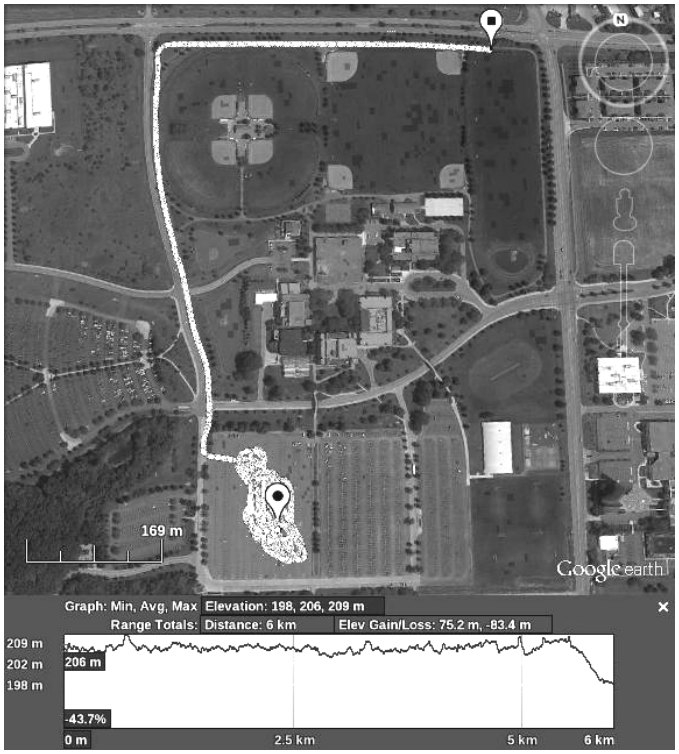

Fig. 2. Reference trajectory from (Toth et al., 2011) (image courtesy of Google Earth (Google Earth, 2014)).

each IMU and GPS, respectively. SINS operation frequency is set to 128 Hz and GPS's is set to 5 Hz. It is clear from Table 2 that IMU ADIS16488 presents smaller errors than IMU ADIS16405, and thus, a better navigation performance is expected from the former.

Table 2. IMU profiles.

	ADIS16405	ADIS16488	
ARW	2	0.3	deg/$\sqrt{\text{h}}$
VRW	0.2	0.029	m/s/$\sqrt{\text{h}}$
$b_g(1\sigma)$	3	0.2	deg/s
$b_f(1\sigma)$	50	16	mg
$\sigma_{g\delta b}$	7E-3	1.8E-3	deg/s
$\sigma_{f\delta b}$	0.2	0.1	mg
Frequency	128	128	Hz

Table 3. GPS 18-5Hz profile.

Position error (2σ)	< 15	m
Velocity error	0.1	knot
Frequency	5	Hz

Since NaveGo works with variables in SI units, most values from Tables 2 and 3 must be converted to be processed. Tables 4 and 5 summarize converted input data to the simulator. Elements of variance vectors are set equally.

Table 6 shows the average RMS errors from both INS and GPS for 10 simulations. Both INS present similar performances except in attitude where the INS with IMU ADIS16488 is more accurate. As expected, it is observed that both INS have better precision in 3D position when compared with the GPS-only solution.

Table 4. Input data to NaveGo I.

	ADIS16405	ADIS16488	
σ_g	5.582E-3	2.746E-4	rad/s
σ_f	3.771E-2	1.521E-3	m/s^2
b_g	5.235E-2	3.490E-3	rad/s
b_f	4.905E-1	1.569E-1	m/s^2
$\sigma_{g\delta b}$	1.221E-4	3.151E-5	rad/s
$\sigma_{f\delta b}$	1.962E-3	9.810E-4	m/s^2
n_g	5.818E-4	8.727E-5	rad/s/Hz
n_f	3.333E-3	4.833E-4	m/s^2/Hz
$n_{g\delta b}$	1.222E-4	3.151E-5	rad/s/Hz
$n_{f\delta b}$	1.962E-3	9.810E-4	m/s^2/Hz
$\boldsymbol{\tau}_g$	inf	inf	s
$\boldsymbol{\tau}_f$	inf	inf	s

Table 5. Input data to NaveGo II.

$\sigma_{\gamma m}$	5	m
$\sigma_{\lambda m}$	5	m
σ_h	10	m
σ_γ	7.8591E-7	rad
σ_λ	1.0219E-6	rad
σ_v	0.0514	m/s
$\hat{\boldsymbol{e}}_{(1)}$	$[0.0223, 0.0234, 0.0550]^T$	rad
$\boldsymbol{p}_o^n$	$[0.698145481, -1.449307157, 204.691]^T$	[rad,rad,m]
$\boldsymbol{l}_{ba}^b$	$[0, 0, 0]^T$	m

Table 6. Average RMS errors from both INS and GPS.

	INS/ADIS16405	INS/ADIS16488	GPS-only	
ϕ	0.1595	0.0564	–	deg
θ	0.2284	0.0540	–	deg
ψ	0.6599	0.3166	–	deg
v_N	0.0224	0.0177	0.0514	m/s
v_E	0.0239	0.0184	0.0517	m/s
v_D	0.0194	0.0164	0.0515	m/s
γ	0.5366	0.5335	5.0632	m
λ	0.6159	0.6125	5.0015	m
h	0.5944	0.5949	9.8185	m

6. VALIDATION METHOD

The aim of this section is to validate the simulation of sensors in particular, and NaveGo in general. In doing so, it is evaluated how close in performance is a simulated INS to a real INS for the same scenario. The proposed validation method simulates sensors with a real trajectory during which real measurements were collected from field. Then, performances from both systems are compared.

6.1 Experiment setup

Real sensors data sets are provided by (Toth et al., 2011), which were logged during the trajectory of Fig. 2 (see Sec. 5). Only three MEMS IMU are taken into account, namely, Xbow IMU400CD (XBOW1), Xsens MTi (XSENS1), and Gladiator Landmark 10 (GLAD). These MEMS IMU represent inertial sensors of different grades, sorted by high to low qualities.

Reference data set $\boldsymbol{T_R}$ (Sec. 5) is used to simulate sensors. Since simulated data sets have the same sampling time of the reference data set, they are downsampled according to sampling times from real sensors. Thus, simulated sensors have close sampling times to corresponding real MEMS IMU and Novatel GPS.

It is worth mentioning that the same three MEMS IMU units used in (Toth et al., 2011) were profiled in (Hasnur-Rabiain, 2010), in which error characteristics were determined by using the Allan variance and PSD techniques. MEMS IMU noise sources are taken from (Hasnur-Rabiain, 2010, Table 4.4, column AV), except static biases that are taken from (Hasnur-Rabiain, 2010, Table 4.3, column Mean). These values are shown in Table 7 for completeness.

Table 7. MEMS IMU profiles.

		XBOW1	XSENS1	GLAD	
σ_g	X	8.0521E-3	1.4364E-2	3.4972E-2	rad/s
	Y	6.3292E-3	1.3617E-2	3.3640E-2	
	Z	7.2158E-3	1.5617E-2	3.2998E-2	
σ_f	X	1.5621E-2	1.4930E-2	5.7851E-1	m/s^2
	Y	1.5621E-2	1.5890E-2	5.7992E-1	
	Z	1.1372E-2	1.7930E-2	5.9787E-1	
b_g	X	7.6969E-2	4.2324E-2	1.0646E-3	rad/s
	Y	6.9813E-3	1.0018E-2	5.6199E-3	
	Z	1.9896E-2	1.8151E-2	1.5620E-2	
b_f	X	0.151	0.3070	0.1730	m/s^2
	Y	0.007	-0.2800	-0.0700	
	Z	0.019	0.0059	0.0212	
$\sigma_{g\delta b}$	X	5.9812E-5	1.9931E-4	2.0141E-4	rad/s
	Y	3.5395E-5	2.1467E-4	2.0943E-4	
	Z	3.7576E-5	2.6633E-4	2.0141E-4	
$\sigma_{f\delta b}$	X	1.7870E-4	4.2740E-4	3.2320E-3	m/s^2
	Y	1.8430E-4	3.3960E-4	3.1710E-3	
	Z	2.1490E-4	4.7690E-4	3.6100E-3	
n_g	X	6.9743E-4	1.4364E-3	2.4748E-3	$\frac{\text{rad/s}}{\sqrt{\text{Hz}}}$
	Y	3.5482E-4	1.3617E-3	2.3806E-3	
	Z	6.2500E-4	1.5617E-3	2.3352E-3	
n_f	X	1.3530E-3	1.4930E-3	4.0940E-2	$\frac{\text{m/s}^2}{\sqrt{\text{Hz}}}$
	Y	1.3530E-3	1.5890E-3	4.1040E-2	
	Z	9.8500E-4	1.7930E-3	4.2310E-2	
τ_g	X	250.7	274.4	458.9	s
	Y	432.2	463.9	289.7	
	Z	395.2	132.2	370.9	
τ_f	X	196.9	33.4	260.2	s
	Y	332.8	124.3	382.8	
	Z	79.2	39.2	351.3	
Frequency		133	100	200	Hz

GPS error profile is shown in Table 8 (Novatel, Inc., 2005). Also, lever arm vectors are provided (Toth et al., 2011).

Table 8. Novatel GPS profile.

$\sigma_{\gamma m}$	0.6660	m
$\sigma_{\lambda m}$	0.5062	m
σ_h	1.2242	m
$\boldsymbol{\sigma}_v$	$[0.1925, 0.1521, 0.0521]^T$	m/s
$\boldsymbol{l}_{ba}^b$ XBOW1	$[-0.781, 0.438, 1.072]^T$	m
$\boldsymbol{l}_{ba}^b$ XSENS1	$[-0.714, 0.434, 1.098]^T$	m
$\boldsymbol{l}_{ba}^b$ GLAD	$[-0.646, 0.539, 1.097]^T$	m
Frequency	5	Hz

Table 9. RMS errors from real and simulated INS.

	INS/XBOW1			INS/XSENS1			INS/GLAD			Average	
	Real	Sim	Diff	Real	Sim	Diff	Real	Sim	Diff	|Diff|	
$\hat{\phi}$	0.337	0.322	0.015	0.677	0.541	0.136	1.362	0.720	0.642	0.264	deg
$\hat{\theta}$	0.656	0.644	0.012	1.109	0.765	0.344	1.454	1.102	0.352	0.236	deg
$\hat{\psi}$	2.518	2.841	-0.323	2.651	2.695	-0.044	5.413	4.186	1.227	0.531	deg
$\hat{v}_N$	0.201	0.130	0.071	0.221	0.146	0.075	0.220	0.155	0.065	0.071	m/s
$\hat{v}_E$	0.168	0.080	0.088	0.170	0.100	0.071	0.198	0.107	0.091	0.083	m/s
$\hat{v}_D$	0.021	0.017	0.004	0.021	0.026	0.007	0.080	0.039	0.041	0.017	m/s
$\hat{\gamma}$	0.446	0.219	0.227	0.462	0.231	0.231	0.474	0.242	0.232	0.230	m
$\hat{\lambda}$	0.357	0.140	0.217	0.350	0.129	0.221	0.386	0.154	0.232	0.224	m
$\hat{h}$	0.233	0.132	0.101	0.218	0.134	0.085	0.221	0.125	0.096	0.094	m

Finally, ideal simulated sensors are added with noises in accordance with the linear error models shown in Sec. 2.

6.2 Result analysis

In order to evaluate the precision of the proposed sensor error models, two types of integrated systems are implemented for each MEMS IMU; a real one (Real) composed of real IMU and real GPS, and a simulated one (Sim) composed of both simulated IMU and simulated GPS from NaveGo. Lastly, real and simulated data sets are processed by using NaveGo. Vectors $\hat{\boldsymbol{e}}_{(1)}$ and $\boldsymbol{p}_o^n$ are taken from Table 5.

Table 9 displays RMS errors (RMSE) from each INS, Real and Sim, with respect to the reference data set, for the three MEMS IMU. Diff columns expose the difference between both real and simulated RMSE, for each IMU. Last column shows the average of absolute values from columns Diff.

Values from Real columns of Table 9 are analysed. As expected, since XBOW1 and XSENS1 have similar error characteristics, their INS performances are also alike. On the other hand, INS/GLAD shows a worse performance, which is coherent with GLAD quality. Thus, performances from the three real INS are coherent with corresponding MEMS IMU error profiles. Therefore, it is shown that NaveGo overall execution is correct.

It can be seen from Table 9 that simulated INS generally have better performance than real INS, although very close to the real ones. Two primary factors cause this situation. Firstly, simulated sensors have pure-Gaussian probability distributions, thus KF estimates are optimal. On the other hand, real sensors' probability distributions are close to Gaussian but not exactly Gaussian. So, KF estimates for these sensors are suboptimal. Second, sensors are represented in the KF as first-order linear systems. This model matches the simulated sensors models, but not the real sensors models since the latter have some level of nonlinearity in their corresponding output signals.

Average column from Table 9 shows that the proposed models to simulate inertial sensors and GPS work well for integrated systems composed of MEMS IMU of different qualities. On average, attitude difference between real and simulated systems is under 0.6 degrees, and 2D position difference is under 23 centimetres. Additionally, vertical position difference is under 10 centimetres. These numbers expose that NaveGo is an adequate tool for assessing the performance of real integrated navigation systems at simulation stage.

7. CONCLUSIONS

In this work, a simulation framework for low-cost integrated navigation systems called NaveGo is presented. This article exposes complete mathematical models to simulate measurements coming from inertial sensors and a GPS receiver, as well as the algorithm of a loosely-couple INS. NaveGo is distributed as an open-source toolbox and can be freely obtained (Navego, 2015).

Proposed models to simulate mentioned sensors in particular, and NaveGo in general, are validated by using a practical procedure. It is exhibited that the performance of a low-cost INS at simulation level is comparable to the performance of a real INS. On average, and for three different MEMS IMU, absolute differences between real and simulated systems are under 0.6 degrees for attitude, under 23 centimetres for 2D position, and under 10 centimetres for vertical position.

As a rule of thumb, in evaluating certain sensors for a low-cost INS and having at hand both precise IMU and GPS error profiles, one can expect that the performance obtained with NaveGo will be close to the operation of a real INS.

As a result, it is concluded that NaveGo is a suitable simulation tool for the design and analysis of low-cost integrated navigation systems.

ACKNOWLEDGEMENTS

The authors would like to thank to Dr. C. Toth, Dr. A. Kealy, and MSc. A. Hasnur-Rabiain for generously share IMU and GPS data sets, and, in particular, for Hasnur-Rabiain's unselfish support. They would also like to thank to Dr. C.A. Catania (from Universidad Nacional de Cuyo, Argentina) for his comments on this work.

REFERENCES

Analog Devices, Inc. (2009a). *ADIS16400/ADIS16405 Data Sheet Rev. B.*

Analog Devices, Inc. (2009b). *ADIS16488 Data Sheet Rev. C.*

Brown, R.G. and Hwang, P.Y.C. (1997). *Introduction to Random Signals and Applied Kalman Filtering, 3rd Ed.* John Wiley & Sons, Inc., USA.

Cai, G., Chen, B.M., and Lee, T.H. (2011). *Unmanned Rotorcraft Systems*. Springer London, UK.

Crassidis, J.L. and Junkins, J.L. (2011). *Optimal Estimation of Dynamic Systems, 2nd Ed.* Chapman and Hall/CRC, USA.

Esposito, F., Accardo, D., Moccia, A., Ciniglio, U., Corraro, F., and Garbarino, L. (2007). Real-time simulation and data fusion of navigation sensors for autonomous aerial vehicles. In *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, 127–136. Springer Netherlands.

Farrell, J. (2008). *Aided Navigation: GPS With High Rate Sensors*. McGraw-Hill Professional, USA.

FreeMat (2014). FreeMat home page. URL `http://freemat.sourceforge.net`.

Garmin International, Inc. (2005). *GPS 18 5Hz Technical specifications Rev. D*.

Giroux, R., Landry, R.J., Leach, B., and Gourdeau, R. (2003). Validation and performance evaluation of a Simulink inertial navigation system simulator. *Canadian Aeronautics and Space Journal*, 49(4), 149–161. doi:10.5589/q03-015.

Gonzalez, R., Giribet, J.I., and Patio, H.D. (2015). An approach to benchmarking of loosely coupled low-cost navigation systems. *Mathematical and Computer Modelling of Dynamical Systems*, 21(3), 272–287. doi: 10.1080/13873954.2014.952642.

Google Earth (2014). The Ohio State University, USA. 40.001319N, 83.039045W. Date of original imagery: 5/29/2010. Date accessed: 4/17/2014.

Grewal, M.S. and Andrews, A.P. (2008). *Kalman Filtering: Theory and Practice Using MATLAB (3rd Ed.)*. John Wiley & Sons, Inc., USA.

Groves, P.D. (2008). *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, USA.

Hasnur-Rabiain, A. (2010). *Performance Evaluation of MEMS based INS/GPS Integration*. Master's thesis, Department of Geomatics, The University of Melbourne.

Liansheng, L. and Tao, J. (2011). Research on strap-down inertial navigation system simulation. In *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, volume 2, 1168–1171. doi:10.1109/ICICTA.2011.577.

Lijun, W., Huichang, Z., and Xiaoniu, Y. (2008). The modeling and simulation for GPS/INS integrated navigation system. In *Microwave and Millimeter Wave Technology, 2008. ICMMT 2008. International Conference on*, volume 4, 1991–1994. doi:10.1109/ICMMT.2008.4540881.

Liu, L. and Amin, M. (2008). Performance analysis of GPS receivers in non-gaussian noise incorporating precorrelation filter and sampling rate. *Signal Processing, IEEE Transactions on*, 56(3), 990–1004. doi: 10.1109/TSP.2006.890827.

NaveGo (2015). An open-source MATLAB toolbox for simulating low-cost integrated navigation systems. URL `http://www.github.com/rodralez/navego`.

Novatel, Inc. (2005). *OEM4 Family User Manual (OM-20000046 Rev 19)*.

Strus, J.M., Kirkpatrick, M., and Sinko, J.W. (2008). Development of a high accuracy pointing system for maneuvering platforms. *Inside GNSS*, March/April, 30–37.

Titterton, D.H. and Weston, J.L. (2004). *Strapdown Inertial Navigation Technology (2nd Ed.)*. Institution of Engineering and Technology, USA.

Toth, C., Brzezinska, D., Politi, N., and Kealy, A. (2011). Reference data set for performance evaluation of MEMS-based integrated navigation solutions. In *FIG Working Week 2011*. Marrakech, Morocco.

Vermeille, H. (2002). Direct transformation from geocentric coordinates to geodetic coordinates. *Journal of Geodesy*, 76, 451–454.

Wenling, L., Pei, C., and Chao, H. (2010). Simulation based on MEMS INS performance analyses in flying mission. In *Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on*, 1205–1208. doi:10.1109/PCSPA.2010.296.

Zhang, W., Ghogho, M., and Yuan, B. (2012). Mathematical model and MATLAB simulation of strapdown inertial navigation system. *Modelling and Simulation in Engineering*, 2012(Article ID 264537), 25. doi: http://dx.doi.org/10.1155/2012/264537.

APPENDIX: LIST OF FUNCTIONS

NaveGo is composed by 34 functions. Next, functions are listed in alphabetical order and a brief description is given:

(1) `acc_gen.m` simulates accelerometers outputs in the b-frame (Sec. 2.2).
(2) `acc_nav2body.m` projects accelerations from n-frame to b-frame (10).
(3) `att_update.m` updates attitude (Crassidis and Junkins, 2011, Eq. 7.39).
(4) `coriolis_b.m` computes Coriolis force in the b-frame (11).
(5) `coriolis.m` computes Coriolis force (11).
(6) `dcm_update.m` updates DCM with Euler angles $\boldsymbol{e}$ (Titterton and Weston, 2004, Eq. 11.4).
(7) `dcm2euler.m` converts DCM to Euler angles (Titterton and Weston, 2004, Eq. 3.66).
(8) `earthrate.m` computes the Earth rate (Titterton and Weston, 2004, Eq. 3.72).
(9) `ecef2llh.m` converts ECEF position to n-frame position (17).
(10) `ecef2ned.m` converts ECEF position to local NED position (inverse process of (15)).
(11) `euler2dcm.m` converts Euler angles to DCM (Titterton and Weston, 2004, Eq. 3.47).
(12) `euler2qua.m` converts Euler angles to quaternion (Titterton and Weston, 2004, Eq. 3.65).
(13) `F_update.m` updates matrix F (26).
(14) `gps_err_profile.m` transforms GPS errors in meters to radians (20).
(15) `gps_gen.m` generates measurements of GPS position and GPS velocity in the n-frame (Sec. 2.3).
(16) `gravity_b.m` computes gravity vector in the b-frame.
(17) `gravity.m` computes gravity vector in the n-frame (Titterton and Weston, 2004, Eq. 3.89–3.91).
(18) `gyro_gen_delta.m` computes incremental angles between time steps (5).
(19) `gyro_gen.m` simulates gyroscopes outputs in the b-frame (Sec. 2.1).
(20) `imu_err_profile.m` adjusts IMU error profile to SI units ((9) and (14), and (30)).

(21) `ins.m` computes the loosely-coupled integration (Alg. 2).
(22) `kalman.m` computes the Kalman filter (Alg. 1).
(23) `llh2ecef.m` converts n-frame position to ECEF position (16).
(24) `navego_example_of_use.m` compares the performances of two INS with different IMU (Sec. 5).
(25) `ned2ecef.m` converts local NED position to ECEF position (15).
(26) `pos_update.m` updates position (Titterton and Weston, 2004, Eq. 3.79–3.81).
(27) `qua2dcm.m` converts quaternion to DCM (Titterton and Weston, 2004, Eq. 3.63).
(28) `qua2euler.m` converts quaternions to Euler angles (see (Farrell, 2008, Eq. 10.3–10.5, pp. 354–355)).
(29) `qua_update.m` updates quaternions (Crassidis and Junkins, 2011, Eq. 7.39).
(30) `radius.m` computes R_M and R_N (Farrell, 2008, Eq. 2.6 and 2.7).
(31) `rmse.m` computes root-mean-square error between two vectors.
(32) `skewm.m` composes a skew symmetric matrix from a vector.
(33) `transportrate.m` computes the transport rate (Titterton and Weston, 2004, Eq. 3.74).
(34) `vel_update.m` updates velocity (Titterton and Weston, 2004, Eq. 3.69).