Code Optimization for Strapdown Inertial Navigation System Algorithm

Ivana Todić and Vladimir Kuzmanović

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71732

Abstract

Inertial navigation systems are in common use for decades due to its advantages. Since INS outputs are usually used for inputs in different control algorithms (depending on applications), INS will induce certain errors and limitations. This chapter deals with optimization of the inertial navigation algorithm against limitations due to the accuracy and stability of signals from the sensors and constraints resulting from the integration step and processor speed used for embedded applications. Inertial navigation considered here is "strapdown" inertial navigation system (SINS) which assumes a fixed inertial measurement unit (IMU). In this chapter, fundamentals of strapdown inertial navigation will be presented as well as three different algorithms which will be analyzed in regard to numerical stability, time consumption and processor load criteria.

Keywords: strapdown inertial navigation system, quaternions, forward Euler integration, code optimization, code analyses

1. Introduction

INS is inertial navigation system, the system that determines the position based on the output of the motion sensors: accelerometers and gyroscopes. The first INS was based on accelerometers mounted on gimbal platform, to ensure measurement of acceleration in navigational frame. Nowadays "strapdown" inertial navigation system (SINS) is in common use, due to its mechanical simplicity, reduced size and price compered to platform INS. Strapdown inertial navigation system implies a fixed inertial measurement unit (IMU), whereby the analytical picture of the navigation system is obtained from the integration of the gyroscope rates.

The main problem that arises when SINS is used is the exact determination of the orientation based on the gyroscopes outputs. Every error made in this stage will affect the error of projection of the gravitational acceleration. Accelerations are integrated twice in order to determine the position, so any errors made when determining the orientation will cause the error in position determination to increase exponentially with integration time.

Errors when determining orientation are caused by the gyroscope performance and precision, as well as signal processing methods used for processing gyroscope outputs. Besides hardware limitations of the gyroscopes, algorithms used for orientation calculation also cause errors. This chapter focuses only on errors caused by applied algorithms and on optimization of these algorithms in terms of time consumption and processor load.

2. Fundamentals of inertial navigation

The basic idea of inertial navigation is based on the integration of acceleration measured by the accelerometers; see [1]. The accelerometers measure the specific force that can be represented as:

$$\mathbf{f} = \mathbf{a} - \mathbf{g} \tag{1}$$

where a is the absolute acceleration, acceleration in relation to the inertial coordinate frame, g is the gravitational acceleration.

In this chapter, the effect of the rotation of the Earth (which can simply be introduced into equations for the needs of systems operating in a longer time interval) is neglected.

In accordance with the previous assumption, the following relationship between acceleration and velocity in relation to the inertial coordinate frame is:

$$\begin{aligned} \mathbf{a} &= \left.\tfrac{d\mathbf{V}}{dt}\right|_I \\ \left.\tfrac{d\mathbf{V}}{dt}\right|_I &= \left.\tfrac{d\mathbf{V}}{dt}\right|_N + \boldsymbol{\omega}_N \times \mathbf{V} \end{aligned} \tag{2}$$

where $\left.\frac{d\mathbf{V}}{dt}\right|_N$ is the speed derivative relative to the navigation coordinate frame, $\boldsymbol{\omega}_N$ is the absolute angular velocity of the navigation coordinate frame.

In the inertial navigation algorithm, for the navigation coordinate frame, the ENUp coordinate frame has been adopted; see [2]. This choice is made due to the desire to have the height coordinate positive and on the other hand in order to more accurately determine the azimuth numerically.

In accordance with the ENUp coordinate frame, the following relations apply:

$$\begin{aligned} f_E &= \frac{dV_E}{dt} + \omega_N V_{up} - \omega_{up} V_N \\ f_N &= \frac{dV_N}{dt} - \omega_E V_{up} + \omega_{up} V_E \\ f_{up} &= \frac{dV_{up}}{dt} + \omega_E V_N - \omega_N V_E + g \end{aligned} \tag{3}$$

As a result of the WGS84 standard for the Earth shape (see [3]), projection of angular speeds of the ENUp coordinate frame has been adopted in the following form:

$$\begin{aligned}\omega_E &= -\frac{V_N}{R_\phi + h}\\ \omega_N &= \frac{V_E}{R_\lambda + h}\\ \omega_{up} &= \frac{V_E}{R_\lambda + h}\tan\phi\end{aligned} \tag{4}$$

where h is the height above the reference ellipsoid, R_ϕ, R_λ is the radius of the curvature of the reference ellipsoid in the north-south and east-west directions, respectively.

$$\begin{aligned}R_\phi &= \frac{R_e\left(1 - e^2\right)}{\left(1 - e^2\sin^2\phi\right)^{\frac{3}{2}}}\\ R_\lambda &= \frac{R_e}{\left(1 - e^2\sin^2\phi\right)^{\frac{1}{2}}}\end{aligned} \tag{5}$$

where R_e is the equatorial radius of the Earth, $e^2 = 1 - \frac{b^2}{a^2}$ is the eccentricity of the reference ellipsoid.

As the accelerometers measure acceleration in the coordinate frame related to the object, it is necessary to determine the transformation matrix from the body frame (see [4]) into the navigation frame, using information from the gyroscopes.

The navigation algorithm adopted here can be divided into two parts. The first part that works with higher frequency plays the role of determining velocity and angle increments, while the other part of the algorithm that works eight times slower provides information on the position and the speed in the navigation coordinate frame (usually required by the guidance law in the case of the missile application). Such algorithm is advantageous from the point of optimization of the calculation time in the control computer, which can be divided into eight different steps. Also, this SINS algorithm proved to be mathematically more stable in relation to others, in determining the quaternion position at the same sampling time. Namely, when integrating angular velocities in order to obtain the angular position, depending on the size of the integration step, the quaternion error increases over time, and in addition to renormalization, it also affects the overall error in position and velocity. This error does not occur with this algorithm.

2.1. Determination of angular increments and transformation matrix

The first step in determining the transformation matrix is the determination of angular inclusions, and as explained above, this process is repeated with the basic integration step which in this example is $t_s = 2$ ms:

$$\alpha_{x_b, y_b, z_b} = \int_{t_k}^{t_k + t_s} \omega_{x_b, y_b, z_b} dt \tag{6}$$

where $\omega_{x_b, yb, zb}$ is the gyroscope signals in the body coordinate frame.

The process of calculating the position quaternion or the transformation matrix is also divided into two parts.

The first part is the calculation of the quaternion between the navigation coordinate frame and the body frame, assuming that the navigation coordinate frame can be considered inert during one step of integration.

The second part is used for the quaternion correction due to the rotation of the navigation coordinate frame.

If we compare these two transformations, we can conclude that the first transformation is the rotation of "fast" motion. One of the reasons why this algorithm proved to be numerically more stable is the separation of the integration of the "fast" rotation from the integration of the "slow" rotation.

If we compare the angular rates of those two motions, we can conclude that the "slow" rotation rates are four or more times lower than the "fast" rotation rates which leads to numerical integral errors when these two rotations are combined.

In accordance with the above, the following relations apply:

$$\begin{aligned} \mathbf{q}_{n+1}^{I} &= \mathbf{q}_n \Delta \mathbf{q}_f \\ \mathbf{q}_{n+1} &= \Delta \mathbf{q}_s \mathbf{q}_{n+1}^{I} \end{aligned} \tag{7}$$

where $\mathbf{q}^I$ is the quaternion of rotation from the body to the inertial coordinate frame, $\mathbf{q}$ is the quaternion of rotation from the body to the navigational coordinate frame, $\Delta\mathbf{q}_f$ is the quaternion of fast rotation increment, $\Delta\mathbf{q}_s$ is the quaternion of slow rotation increment.

The quaternion of fast rotation can be represented in the form of a rotary vector as follows:

$$\Delta \mathbf{q}_f = \begin{bmatrix} \Delta q_{f0} \\ \Delta q_{f1} \\ \Delta q_{f2} \\ \Delta q_{f3} \end{bmatrix} = \begin{bmatrix} \cos \frac{\Delta\Phi}{2} \\ \frac{\Delta\Phi_{xb}}{\Delta\Phi} \sin \frac{\Delta\Phi}{2} \\ \frac{\Delta\Phi_{yb}}{\Delta\Phi} \sin \frac{\Delta\Phi}{2} \\ \frac{\Delta\Phi_{zb}}{\Delta\Phi} \sin \frac{\Delta\Phi}{2} \end{bmatrix} \tag{8}$$

The following relationship holds for small angles:

$$\Delta \mathbf{\Phi} = \int_{t_n}^{t_n+t_m} \boldsymbol{\omega} dt + \frac{1}{2}\int_{t_n}^{t_n+t_m} (\mathbf{\Phi} \times \boldsymbol{\omega}) dt \tag{9}$$

where $t_m = 8t_s$ is the slow integration step.

To solve the previous equation, a four-step algorithm will be used (Conning correction [5–7]):

$$\Delta\mathbf{\Phi} = \begin{bmatrix} \Delta\Phi_{xb} \\ \Delta\Phi_{yb} \\ \Delta\Phi_{zb} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{4}\alpha_{xb}(j) \\ \sum_{j=1}^{4}\alpha_{yb}(j) \\ \sum_{j=1}^{4}\alpha_{zb}(j) \end{bmatrix} + \frac{2}{3}\left(\mathbf{P}_1 \begin{bmatrix} \alpha_{xb}(2) \\ \alpha_{yb}(2) \\ \alpha_{zb}(2) \end{bmatrix} + \mathbf{P}_3 \begin{bmatrix} \alpha_{xb}(4) \\ \alpha_{yb}(4) \\ \alpha_{zb}(4) \end{bmatrix}\right)$$
$$+\frac{1}{2}(\mathbf{P}_1+\mathbf{P}_2)\left(\begin{bmatrix} \alpha_{xb}(3) \\ \alpha_{yb}(3) \\ \alpha_{zb}(3) \end{bmatrix} + \begin{bmatrix} \alpha_{xb}(4) \\ \alpha_{yb}(4) \\ \alpha_{zb}(4) \end{bmatrix}\right) + \frac{1}{30}(\mathbf{P}_1-\mathbf{P}_2)\left(\begin{bmatrix} \alpha_{xb}(3) \\ \alpha_{yb}(3) \\ \alpha_{zb}(3) \end{bmatrix} - \begin{bmatrix} \alpha_{xb}(4) \\ \alpha_{yb}(4) \\ \alpha_{zb}(4) \end{bmatrix}\right) \tag{10}$$

where

$$\alpha(j) = \alpha^k(t_s) + \alpha^{k-1}(t_s)$$

$$\mathbf{P}_j = \begin{bmatrix} 0 & -\alpha_{zb}(j) & \alpha_{yb}(j) \\ \alpha_{zb}(j) & 0 & -\alpha_{xb}(j) \\ -\alpha_{yb}(j) & \alpha_{xb}(j) & 0 \end{bmatrix}$$

If we return to the quaternion of slow rotation, the following relationship is valid:

$$\Delta\mathbf{q}_s = \begin{bmatrix} \cos\frac{\Omega t_m}{2} \\ -\frac{\Omega_x}{\Omega}\sin\frac{\Omega t_m}{2} \\ -\frac{\Omega_y}{\Omega}\sin\frac{\Omega t_m}{2} \\ -\frac{\Omega_z}{\Omega}\sin\frac{\Omega t_m}{2} \end{bmatrix} \tag{11}$$

where. $\Omega_x, \Omega_y, \Omega_z$ is the projections of the absolute angular velocity of the navigation coordinate frame on its axes.

If we neglect the rotation of the Earth, the following applies:

$$\begin{aligned} \Omega_x &= -\frac{V_y}{R_y} - \frac{V_x}{a}e^2 b_{13} b_{23} \\ \Omega_y &= \frac{V_x}{R_x} + \frac{V_y}{a}e^2 b_{13} b_{23} \\ \Omega_z &= 0 \\ \frac{1}{R_x} &= \frac{1}{a}\left(1 - e^2\frac{{b_{33}}^2}{2} + e^2 {b_{13}}^2 - \frac{h}{a}\right) \\ \frac{1}{R_y} &= \frac{1}{a}\left(1 - e^2\frac{{b_{33}}^2}{2} + e^2 {b_{23}}^2 - \frac{h}{a}\right) \end{aligned} \tag{12}$$

where b_{ij} are members of the transformation matrix from the Earth-coordinate frame (ECEF) into the navigation coordinate frame $\mathbf{B}_{ECEF}^{n}$.

The Poisson equation for the transformation matrix from the coordinate frame related to the Earth (ECEF) in the navigation coordinate frame can be written in the following form:

$$\begin{aligned}
\dot{\mathbf{B}}_{n}^{ECEF} &= \mathbf{B}_{n}^{ECEF}\Delta\boldsymbol{\omega}_{n-ECEF} \\
\mathbf{B}_{ECEF}^{n} &= \left(\mathbf{B}_{n}^{ECEF}\right)^{T} \\
\Delta\boldsymbol{\omega}_{n-ECEF} &= \begin{bmatrix} 0 & 0 & \Omega_y \\ 0 & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix}
\end{aligned} \tag{13}$$

The recursive solution of the Poisson equation can be represented in the following way:

$$\begin{aligned}
b_{12}(N) &= b_{12}(N-1) - \Omega_y b_{32}(N-1)t_m \\
b_{22}(N) &= b_{22}(N-1) + \Omega_x b_{32}(N-1)t_m \\
b_{32}(N) &= b_{32}(N-1) + \left(\Omega_y b_{12}(N-1) - \Omega_x b_{22}(N-1)\right)t_m \\
b_{13}(N) &= b_{13}(N-1) - \Omega_y b_{33}(N-1)t_m \\
b_{23}(N) &= b_{23}(N-1) + \Omega_x b_{33}(N-1)t_m \\
b_{33}(N) &= b_{33}(N-1) + \left(\Omega_y b_{13}(N-1) - \Omega_x b_{23}(N-1)\right)t_m \\
b_{31}(N) &= b_{12}(N)b_{23}(N) - b_{22}(N)b_{13}(N)
\end{aligned} \tag{14}$$

With the quaternion of fast and the quaternion of slow rotations defined above, on the basis of Eq. (7), the quaternion of total rotation can be determined and with its direct cosine matrix representing the transformation from the body to the navigation coordinate frame. This matrix will be updated with the time step of the slow integration:

$$\mathbf{C}_{b}^{n} = \begin{bmatrix} 1-2\left(q_2{}^2+q_3{}^2\right) & 2\left(q_1q_2-q_0q_3\right) & 2\left(q_0q_2+q_1q_3\right) \\ 2\left(q_1q_2+q_0q_3\right) & 1-2\left(q_1{}^2+q_3{}^2\right) & 2\left(q_2q_3-q_0q_1\right) \\ 2\left(q_1q_3-q_0q_2\right) & 2\left(q_0q_1+q_2q_3\right) & 1-2\left(q_1{}^2+q_2{}^2\right) \end{bmatrix} \tag{15}$$

2.2. Determination of speed and position in space

Previously defined method used for determining the angle increments based on measured gyroscope signals can now be used in the same way to define the speed increments based on signals from the accelerometer. These increments are also determined by the fast integration step t_s:

$$\Delta W_{x_b,y_b,z_b} = \int_{t_k}^{t_k+t_s} a_{x_b,y_b,z_b}dt \tag{16}$$

where $a_{x_b,yb,zb}$ is the signals from the accelerometer in the body coordinate frame.

The absolute acceleration can be written in the following form:

$$\left.\frac{d\mathbf{V}}{dt}\right|_I = \left.\frac{d\mathbf{V}}{dt}\right|_b + \boldsymbol{\omega}_b \times \mathbf{V} \tag{17}$$

where $\left.\frac{d\mathbf{V}}{dt}\right|_b$ is the total speed derivatives with respect to the body coordinate frame, $\left.\frac{d\mathbf{V}}{dt}\right|_I$ is the total speed derivatives with respect to the inertial coordinate frame, $\boldsymbol{\omega}_b$ is the absolute angular velocity of the body coordinate frame.

The specific force projections acting in the body coordinate frame are obtained from the accelerometer. Accordingly, the integration will be performed in the body coordinate frame, and the previous equation can be written like

$$\left.\frac{d\mathbf{V}}{dt}\right|_b = \left.\frac{d\mathbf{V}}{dt}\right|_I - \boldsymbol{\omega}_b \times \mathbf{V} \tag{18}$$

If we apply integration with the slow integration step to the previous equation, we obtain the following:

$$\begin{aligned}
\int_{t_k}^{t_k+t_m} \frac{d\tilde{V}_{x_b}}{dt}dt &= \int_{t_k}^{t_k+t_m} \frac{d\overline{V}_{x_b}}{dt}dt + \int_{t_k}^{t_k+t_m} \left(\omega_{z_b}V_{y_b} - \omega_{y_b}V_{z_b}\right)dt \\
\int_{t_k}^{t_k+t_m} \frac{d\tilde{V}_{y_b}}{dt}dt &= \int_{t_k}^{t_k+t_m} \frac{d\overline{V}_{y_b}}{dt}dt + \int_{t_k}^{t_k+t_m} \left(\omega_{x_b}V_{z_b} - \omega_{z_b}V_{x_b}\right)dt \\
\int_{t_k}^{t_k+t_m} \frac{d\tilde{V}_{z_b}}{dt}dt &= \int_{t_k}^{t_k+t_m} \frac{d\overline{V}_{z_b}}{dt}dt + \int_{t_k}^{t_k+t_m} \left(\omega_{y_b}V_{x_b} - \omega_{x_b}V_{y_b}\right)dt
\end{aligned} \tag{19}$$

The recursive solution of the previous equations is done in eight steps (sculling correction; see [5–7]) from which the step of slow integration was adopted as $t_m = 8t_s$:

$$\begin{aligned}
W_{x_b,k} &= W_{x_b,k-1} + W_{y_b,k-1}\alpha_{z_b,k} - W_{z_b,k-1}\alpha_{y_b,k} + \Delta W_{x_b,k} \\
W_{y_b,k} &= W_{y_b,k-1} + W_{z_b,k-1}\alpha_{x_b,k} - W_{x_b,k-1}\alpha_{z_b,k} + \Delta W_{y_b,k} \\
W_{z_b,k} &= W_{z_b,k-1} + W_{x_b,k-1}\alpha_{y_b,k} - W_{y_b,k-1}\alpha_{x_b,k} + \Delta W_{z_b,k} \\
W_{z_b,k} &= W_{z_b,k-1} + W_{x_b,k}\alpha_{y_b,k} - W_{y_b,k}\alpha_{x_b,k} + \Delta W_{z_b,k} \\
W_{y_b,k} &= W_{y_b,k-1} + W_{z_b,k}\alpha_{x_b,k} - W_{x_b,k}\alpha_{z_b,k} + \Delta W_{y_b,k} \\
W_{x_b,k} &= W_{x_b,k-1} + W_{y_b,k}\alpha_{z_b,k} - W_{z_b,k}\alpha_{y_b,k} + \Delta W_{x_b,k}
\end{aligned} \tag{20}$$

The initial values in each new step of slow integration are $W_{x_b} = W_{y_b} = W_{z_b} = 0$.

After calculating the velocity increments in the body coordinate frame, it is possible to determine the increment of the velocities in the navigation coordinate frame, since the matrix of transformation between the body and the navigational coordinate frame has already been defined:

$$\begin{bmatrix} \Delta W_x \\ \Delta W_y \\ \Delta W_z \end{bmatrix} = \mathbf{C}_b^n \begin{bmatrix} W_{x_b} \\ W_{y_b} \\ W_{z_b} \end{bmatrix} \tag{21}$$

The speed of the object relative to the Earth in the navigation coordinate frame can now be represented by the following relations, with the remark that the Earth's rotation that is neglected:

$$\begin{aligned} V_x &= W_x - \int_{t_0}^{t} V_z \Omega_y dt \\ V_y &= W_y + \int_{t_0}^{t} V_z \Omega_x dt \\ V_z &= W_z - \int_{t_0}^{t} \left(V_y \Omega_x - V_x \Omega_y + g\right) dt \end{aligned} \tag{22}$$

where Ω_x, Ω_y, Ω_z is the projections of the absolute angular velocity of the navigation coordinate frame on its axes, W_x, W_y, W_z is the sums of projections of velocity increments in the navigational frame.

The determination of the position in the navigation coordinate frame can be solved in two ways: by integration of the velocities, which is the case in determining the height, or by the relationship between the matrix defined by Poisson's equation and its definitions:

$$\mathbf{B}_{ECEF}^n = \begin{bmatrix} -\sin\varphi\cos\lambda\sin\varepsilon - \sin\lambda\cos\varepsilon & \sin\varphi\sin\lambda\sin\varepsilon + \cos\lambda\cos\varepsilon & \cos\varphi\sin\varepsilon \\ -\sin\varphi\cos\lambda\cos\varepsilon + \sin\lambda\sin\varepsilon & -\sin\varphi\sin\lambda\cos\varepsilon - \cos\lambda\sin\varepsilon & \cos\varphi\cos\varepsilon \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix} \tag{23}$$

where φ is the latitude, λ is the longitude, ε is the azimuth.

Geographical navigation parameters can be determined from the relation of the preceding equation and Eq. (14):

$$\begin{aligned} \phi &= \arctan\frac{b_{33}}{b_0} \quad [-90, +90] \\ \lambda &= \arctan\frac{b_{32}}{b_{31}} \quad [-180, 180] \\ \varepsilon &= \arctan\frac{b_{13}}{b_{23}} \quad [0, 360] \\ b_0 &= \sqrt{{b_{13}}^2 + {b_{23}}^2} \end{aligned} \tag{24}$$

As the azimuth is now defined, projections of speed in the ENUp coordinate frame can be determined:

$$\begin{aligned} V_N &= V_y\cos\varepsilon + V_x\sin\varepsilon \\ V_E &= -V_y\sin\varepsilon + V_x\cos\varepsilon \end{aligned} \tag{25}$$

The position in the ENUp frame can be determined as

$$
\begin{aligned}
E &= \frac{180}{\pi}(\lambda - \lambda_0)\cos(\varphi_0)a \\
N &= \frac{180}{\pi}(\varphi - \varphi_0)a \\
h &= \int_{t_0}^{t} V_z dt
\end{aligned}
\tag{26}
$$

Similarly, using the matrix definition from the navigation coordinate frame and the body frame, we can get to the relations for angular positions:

$$
\begin{aligned}
\psi &= \arctan\left(\frac{C_b{}^n(1,1)}{C_b{}^n(2,1)}\right) \\
\varphi &= \arctan\left(\frac{C_b{}^n(3,2)}{C_b{}^n(3,3)}\right) \\
\theta &= \arcsin(C_b{}^n(3,1))
\end{aligned}
\tag{27}
$$

3. Strapdown INS (SINS) algorithms

Three SINS algorithms based on previously defined mathematical model will be presented here.

The basic solution of SINS is forward Euler method applied to the main equations for rotation and translation. Block diagram of this method is presented in **Figure 1**. In this algorithm there is no division to the fast and the slow rotation, and all calculation is done in each step.

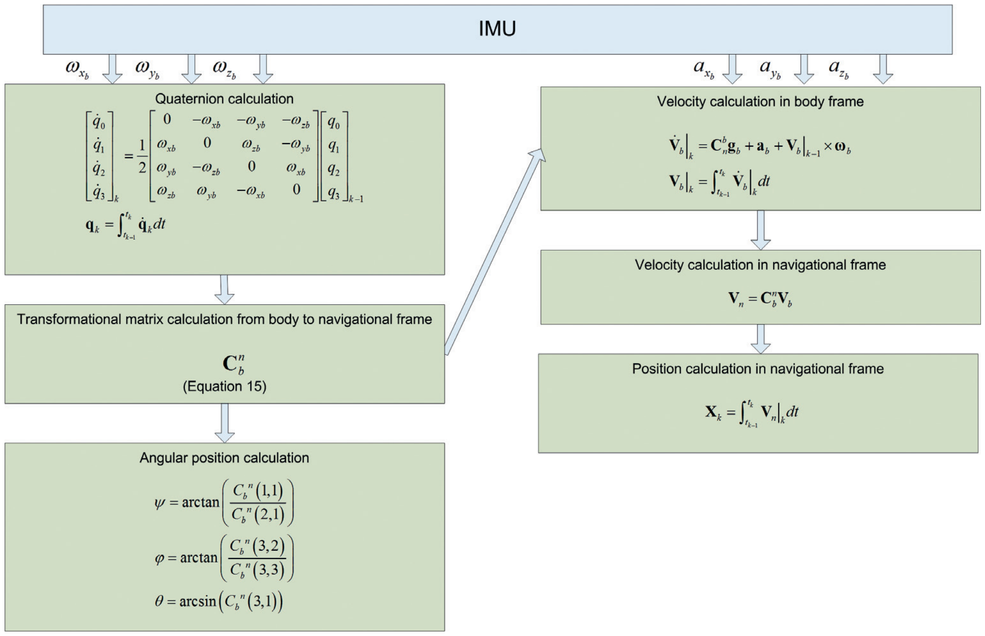
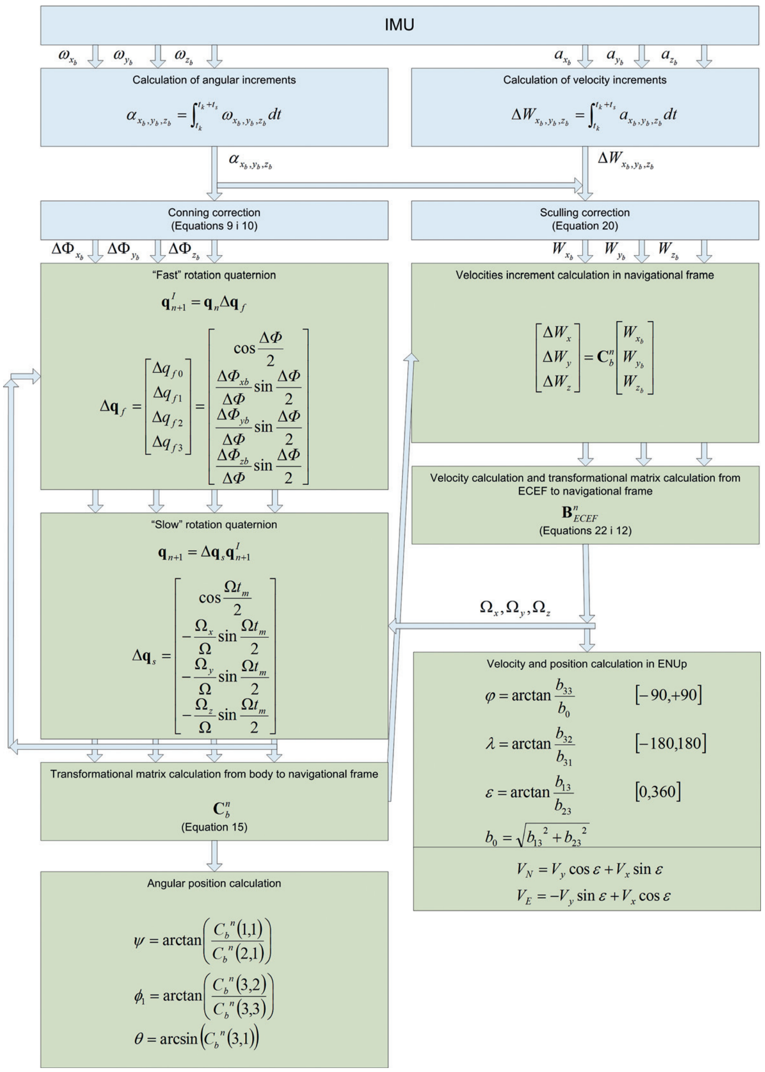

IMU
ω_{x_b}
ω_{y_b}
ω_{z_b}
a_{x_b}
a_{y_b}
a_{z_b}
Quaternion calculation
$\begin{bmatrix}\dot q_0\\ \dot q_1\\ \dot q_2\\ \dot q_3\end{bmatrix}_k = \frac{1}{2}\begin{bmatrix}0 & -\omega_{xb} & -\omega_{yb} & -\omega_{zb}\\ \omega_{xb} & 0 & \omega_{zb} & -\omega_{yb}\\ \omega_{yb} & -\omega_{zb} & 0 & \omega_{xb}\\ \omega_{zb} & \omega_{yb} & -\omega_{xb} & 0\end{bmatrix}\begin{bmatrix}q_0\\ q_1\\ q_2\\ q_3\end{bmatrix}_{k-1}$
$\mathbf{q}_k = \int_{t_{k-1}}^{t_k} \dot{\mathbf{q}}_k dt$
Transformational matrix calculation from body to navigational frame
$\mathbf{C}_b^n$
(Equation 15)
Angular position calculation
$\psi = \arctan\left(\frac{C_b{}^n(1,1)}{C_b{}^n(2,1)}\right)$
$\varphi = \arctan\left(\frac{C_b{}^n(3,2)}{C_b{}^n(3,3)}\right)$
$\theta = \arcsin\left(C_b{}^n(3,1)\right)$
Velocity calculation in body frame
$\dot{\mathbf{V}}_b\big|_k = \mathbf{C}_n^b\mathbf{g}_b + \mathbf{a}_b + \mathbf{V}_b\big|_{k-1} \times \boldsymbol{\omega}_b$
$\mathbf{V}_b\big|_k = \int_{t_{k-1}}^{t_k} \dot{\mathbf{V}}_b\big|_k dt$
Velocity calculation in navigational frame
$\mathbf{V}_n = \mathbf{C}_b^n\mathbf{V}_b$
Position calculation in navigational frame
$\mathbf{X}_k = \int_{t_{k-1}}^{t_k} \mathbf{V}_n\big|_k dt$

Figure 1. Forward Euler SINS algorithm block diagram.

The other solution of SINS algorithm—the regular SINS—based on mathematical model previously defined is presented in **Figure 2** as block diagram. The regular SINS algorithm calculates the velocity and angle increments eight times, and in the last step, Conning and Sculling corrections are implemented including all the other equations in **Figure 2**.



IMU
ω_{x_b} ω_{y_b} ω_{z_b}
a_{x_b} a_{y_b} a_{z_b}
Calculation of angular increments
$\alpha_{x_b,y_b,z_b} = \int_{t_k}^{t_k+t_s} \omega_{x_b,y_b,z_b} dt$
Calculation of velocity increments
$\Delta W_{x_b,y_b,z_b} = \int_{t_k}^{t_k+t_s} a_{x_b,y_b,z_b} dt$
α_{x_b,y_b,z_b}
$\Delta W_{x_b,y_b,z_b}$
Conning correction (Equations 9 i 10)
Sculling correction (Equation 20)
$\Delta\Phi_{x_b}$ $\Delta\Phi_{y_b}$ $\Delta\Phi_{z_b}$
W_{x_b} W_{y_b} W_{z_b}
"Fast" rotation quaternion
$\mathbf{q}_{n+1}^{I} = \mathbf{q}_n \Delta\mathbf{q}_f$
$\Delta\mathbf{q}_f = \begin{bmatrix}\Delta q_{f0}\\ \Delta q_{f1}\\ \Delta q_{f2}\\ \Delta q_{f3}\end{bmatrix} = \begin{bmatrix}\cos\frac{\Delta\Phi}{2}\\ \frac{\Delta\Phi_{xb}}{\Delta\Phi}\sin\frac{\Delta\Phi}{2}\\ \frac{\Delta\Phi_{yb}}{\Delta\Phi}\sin\frac{\Delta\Phi}{2}\\ \frac{\Delta\Phi_{zb}}{\Delta\Phi}\sin\frac{\Delta\Phi}{2}\end{bmatrix}$
Velocities increment calculation in navigational frame
$\begin{bmatrix}\Delta W_x\\ \Delta W_y\\ \Delta W_z\end{bmatrix} = \mathbf{C}_b^n \begin{bmatrix}W_{x_b}\\ W_{y_b}\\ W_{z_b}\end{bmatrix}$
"Slow" rotation quaternion
$\mathbf{q}_{n+1} = \Delta\mathbf{q}_s \mathbf{q}_{n+1}^{I}$
$\Delta\mathbf{q}_s = \begin{bmatrix}\cos\frac{\Omega t_m}{2}\\ -\frac{\Omega_x}{\Omega}\sin\frac{\Omega t_m}{2}\\ -\frac{\Omega_y}{\Omega}\sin\frac{\Omega t_m}{2}\\ -\frac{\Omega_z}{\Omega}\sin\frac{\Omega t_m}{2}\end{bmatrix}$
Velocity calculation and transformational matrix calculation from ECEF to navigational frame
$\mathbf{B}_{ECEF}^{n}$
(Equations 22 i 12)
$\Omega_x, \Omega_y, \Omega_z$
Velocity and position calculation in ENUp
$\varphi = \arctan\frac{b_{33}}{b_0}$ $[-90,+90]$
$\lambda = \arctan\frac{b_{32}}{b_{31}}$ $[-180,180]$
$\varepsilon = \arctan\frac{b_{13}}{b_{23}}$ $[0,360]$
$b_0 = \sqrt{b_{13}^2 + b_{23}^2}$
$V_N = V_y\cos\varepsilon + V_x\sin\varepsilon$
$V_E = -V_y\sin\varepsilon + V_x\cos\varepsilon$
Transformational matrix calculation from body to navigational frame
$\mathbf{C}_b^n$
(Equation 15)
Angular position calculation
$\psi = \arctan\left(\frac{C_b^n(1,1)}{C_b^n(2,1)}\right)$
$\phi_1 = \arctan\left(\frac{C_b^n(3,2)}{C_b^n(3,3)}\right)$
$\theta = \arcsin\left(C_b^n(3,1)\right)$

Figure 2. Regular SINS algorithm block diagram.

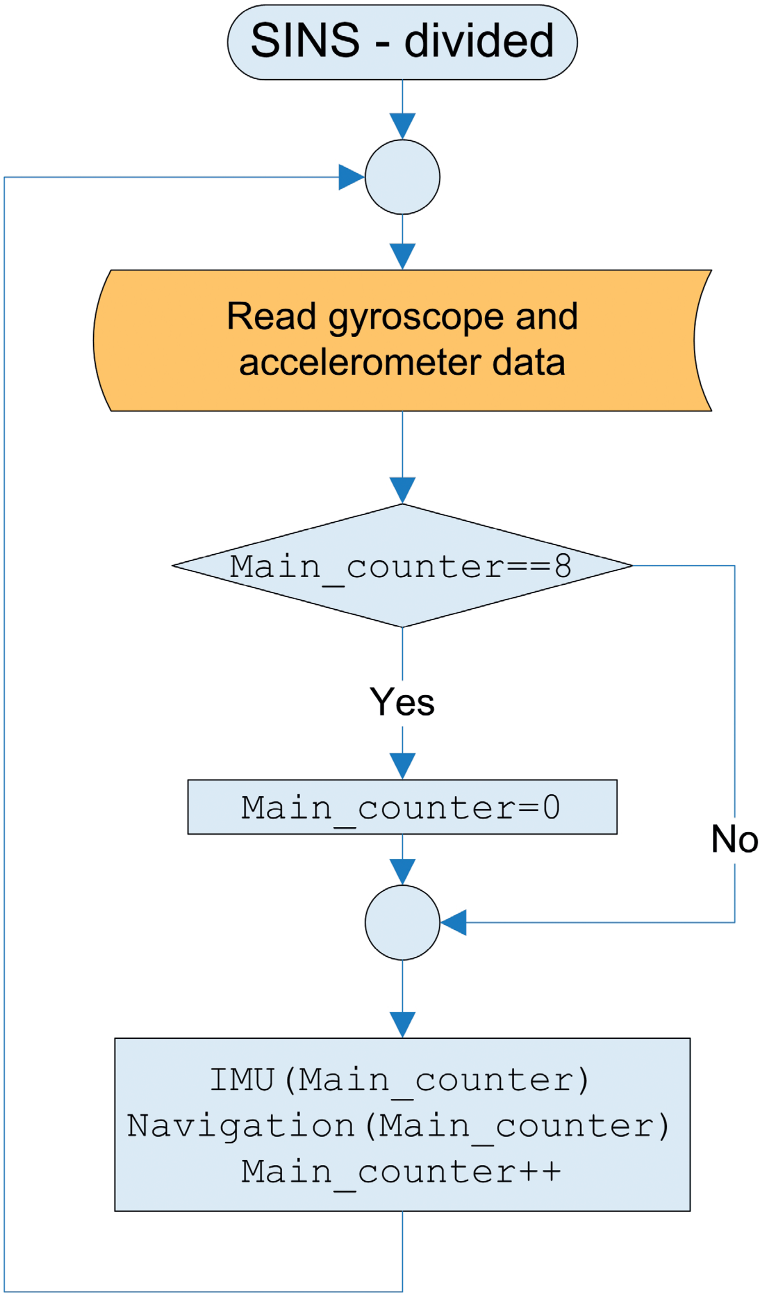
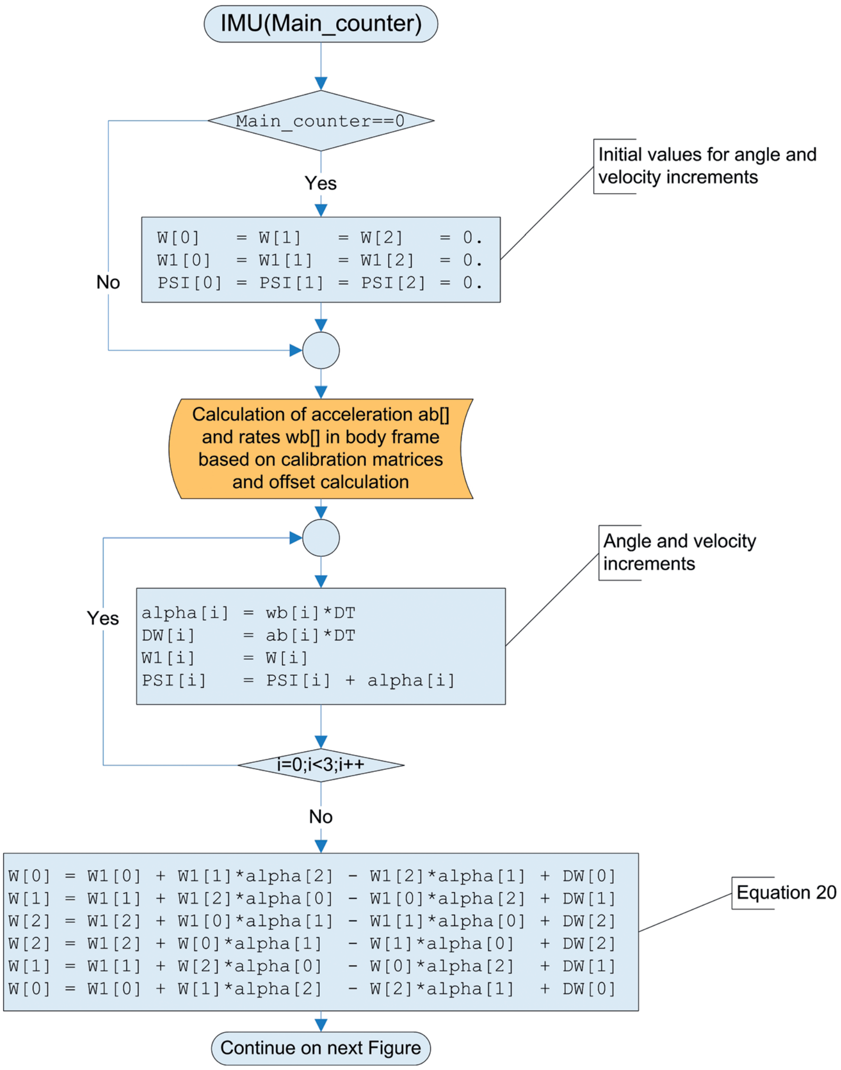


Figure 3. Divided SINS algorithm main flowchart.


IMU(Main_counter)
Main_counter==0
Yes
No
Initial values for angle and velocity increments
Calculation of acceleration ab[] and rates wb[] in body frame based on calibration matrices and offset calculation
Yes
Angle and velocity increments
i=0;i<3;i++
No
Equation 20
Continue on next Figure

```
W[0]   = W[1]   = W[2]   = 0.
W1[0]  = W1[1]  = W1[2]  = 0.
PSI[0] = PSI[1] = PSI[2] = 0.
```

```
alpha[i] = wb[i]*DT
DW[i]    = ab[i]*DT
W1[i]    = W[i]
PSI[i]   = PSI[i] + alpha[i]
```

```
W[0] = W1[0] + W1[1]*alpha[2] - W1[2]*alpha[1] + DW[0]
W[1] = W1[1] + W1[2]*alpha[0] - W1[0]*alpha[2] + DW[1]
W[2] = W1[2] + W1[0]*alpha[1] - W1[1]*alpha[0] + DW[2]
W[2] = W1[2] + W[0]*alpha[1]  - W[1]*alpha[0]  + DW[2]
W[1] = W1[1] + W[2]*alpha[0]  - W[0]*alpha[2]  + DW[1]
W[0] = W1[0] + W[1]*alpha[2]  - W[2]*alpha[1]  + DW[0]
```

Figure 4. IMU procedure flowchart—part one.

The last solution that is considered is SINS algorithm divided in eight steps. This eight-step algorithm naturally arose as a consequence of Conning equation, and it is presented in the following flowcharts.

From **Figure 3**, it can be seen that in each step, the main algorithm will call IMU and navigation procedures. This means that in each step, some part of calculation will be completed.

From **Figure 4**, it can be seen that sculling correction will be calculated in each step (**Figures 5** and **6**).

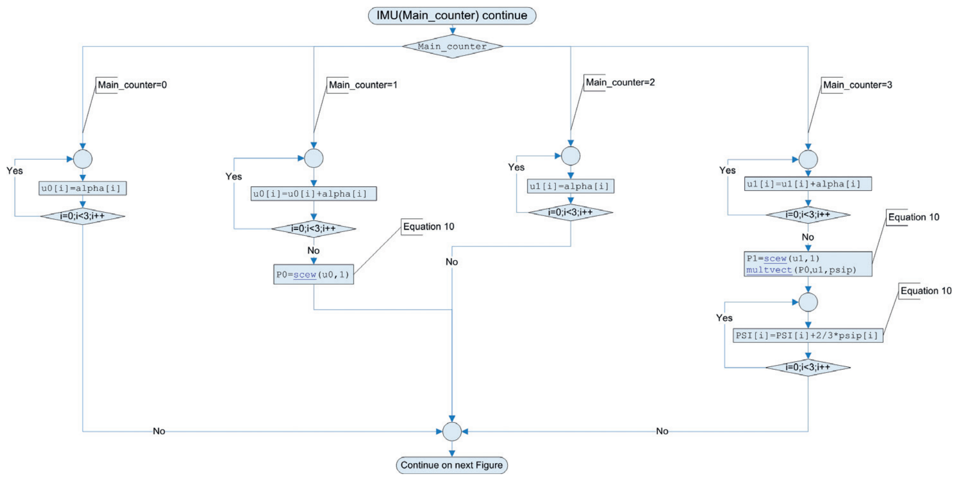


Figure 5. IMU procedure flowchart—part two.

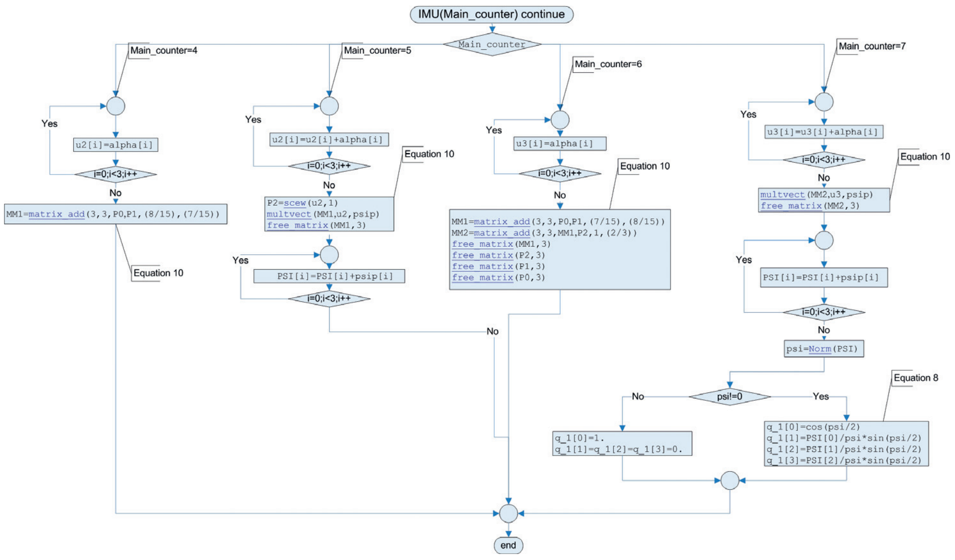
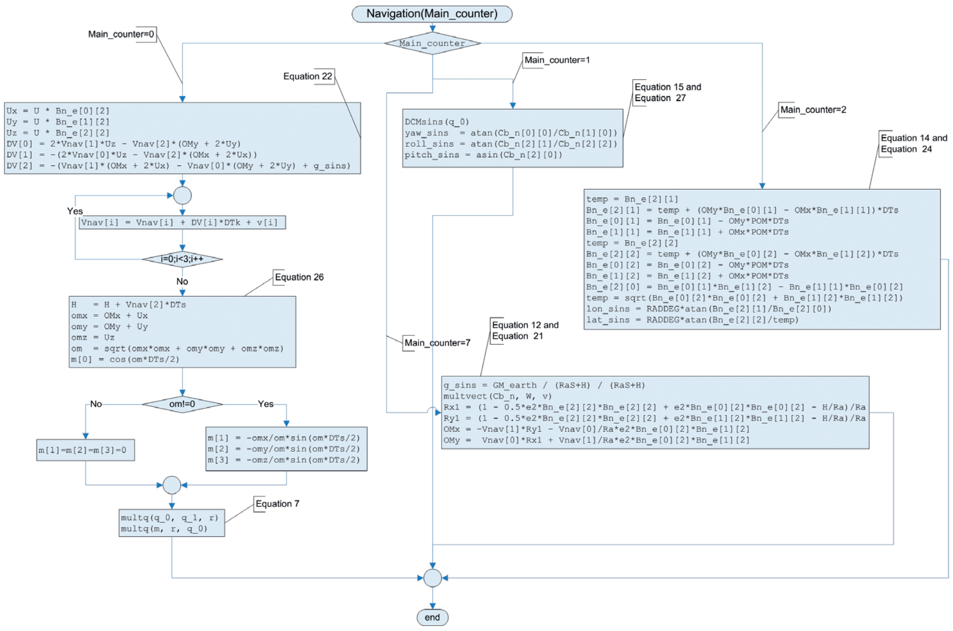


Figure 6. IMU procedure flowchart—part three.

Similar to the IMU algorithm, the navigation procedure is also divided into several steps shown in **Figure 7**.

Availability of output data calculated by all three SINS algorithms is presented in **Table 1**.

From **Table 1**, it can be seen that forward Euler algorithm provides all SINS output values in every step unlike regular and divided SINS which will provide outputs eight times slower.



```
Ux = U * Bn_e[0][2]
Uy = U * Bn_e[1][2]
Uz = U * Bn_e[2][2]
DV[0] = 2*Vnav[1]*Uz - Vnav[2]*(OMy + 2*Uy)
DV[1] = -(2*Vnav[0]*Uz - Vnav[2]*(OMx + 2*Ux))
DV[2] = -(Vnav[1]*(OMx + 2*Ux) - Vnav[0]*(OMy + 2*Uy) + g_sins)
```

```
Vnav[i] = Vnav[i] + DV[i]*DTk + v[i]
```

```
H   = H + Vnav[2]*DTs
omx = OMx + Ux
omy = OMy + Uy
omz = Uz
om  = sqrt(omx*omx + omy*omy + omz*omz)
m[0] = cos(om*DTs/2)
```

```
m[1]=m[2]=m[3]=0
```

```
m[1] = -omx/om*sin(om*DTs/2)
m[2] = -omy/om*sin(om*DTs/2)
m[3] = -omz/om*sin(om*DTs/2)
```

```
multq(q_0, q_1, r)
multq(m, r, q_0)
```

```
DCMsins(q_0)
yaw_sins  = atan(Cb_n[0][0]/Cb_n[1][0])
roll_sins = atan(Cb_n[2][1]/Cb_n[2][2])
pitch_sins = asin(Cb_n[2][0])
```

```
temp = Bn_e[2][1]
Bn_e[2][1] = temp + (OMy*Bn_e[0][1] - OMx*Bn_e[1][1])*DTs
Bn_e[0][1] = Bn_e[0][1] - OMy*POM*DTs
Bn_e[1][1] = Bn_e[1][1] + OMx*POM*DTs
temp = Bn_e[2][2]
Bn_e[2][2] = temp + (OMy*Bn_e[0][2] - OMx*Bn_e[1][2])*DTs
Bn_e[0][2] = Bn_e[0][2] - OMy*POM*DTs
Bn_e[1][2] = Bn_e[1][2] + OMx*POM*DTs
Bn_e[2][0] = Bn_e[0][1]*Bn_e[1][2] - Bn_e[1][1]*Bn_e[0][2]
temp = sqrt(Bn_e[0][2]*Bn_e[0][2] + Bn_e[1][2]*Bn_e[1][2])
lon_sins = RADDEG*atan(Bn_e[2][1]/Bn_e[2][0])
lat_sins = RADDEG*atan(Bn_e[2][2]/temp)
```

```
g_sins = GM_earth / (RaS+H) / (RaS+H)
multvect(Cb_n, W, v)
Rx1 = (1 - 0.5*e2*Bn_e[2][2]*Bn_e[2][2] + e2*Bn_e[0][2]*Bn_e[0][2] - H/Ra)/Ra
Ry1 = (1 - 0.5*e2*Bn_e[2][2]*Bn_e[2][2] + e2*Bn_e[1][2]*Bn_e[1][2] - H/Ra)/Ra
OMx = -Vnav[1]*Ry1 - Vnav[0]/Ra*e2*Bn_e[0][2]*Bn_e[1][2]
OMy =  Vnav[0]*Rx1 + Vnav[1]/Ra*e2*Bn_e[0][2]*Bn_e[1][2]
```

Figure 7. Navigation procedure flowchart.

Algorithm step	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Forward Euler method	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ
Regular SINS	—	—	—	—	—	—	—	V_{xyz}, $\mathbf{B}^n_{ECEF}$, H, lat, lon, $\mathbf{C}^n_b$, ϕ, θ, ψ
Divided SINS	V_x, V_y, V_z, H	$\mathbf{C}^n_b$, ϕ, θ, ψ	$\mathbf{B}^n_{ECEF}$, lat, lon	—	—	—	—	—

Table 1. Comparison of available data in each step for different SINS algorithms.

Generally, guidance and autopilot algorithms do not require inputs with such high frequency, and both regular and divided SINS will usually satisfy requirements; see [8]. On the other hand, if we compare the regular and the divided SINS algorithm, we can see that in the case of the divided SINS algorithm, the entire mission algorithm can be optimized in these eight steps.

The regular and the divided SINS algorithms are based on the same numerical integration, and the results of those two algorithms are equal in time. On the other side, we can compare quaternion stability of forward Euler integration and regular SINS algorithm in time. Quaternion norm which needs to be equal to one for quaternion of rotation is sensitive to the integration step for forward Euler integration. Both algorithms were implemented in MATLAB Simulink. Norm of quaternion is presented in **Figure 8** for the same integration step of 2 ms and for the same input data of gyroscopes presented in **Figure 9**. From **Figure 8** it can be seen that the quaternion norm will be affected whenever there is significant movement of the object.

Quaternion norm error will further affect all outputs of SINS algorithm, and that will lead to error accumulation over time. **Figure 10** represents angle errors for the same simulation.

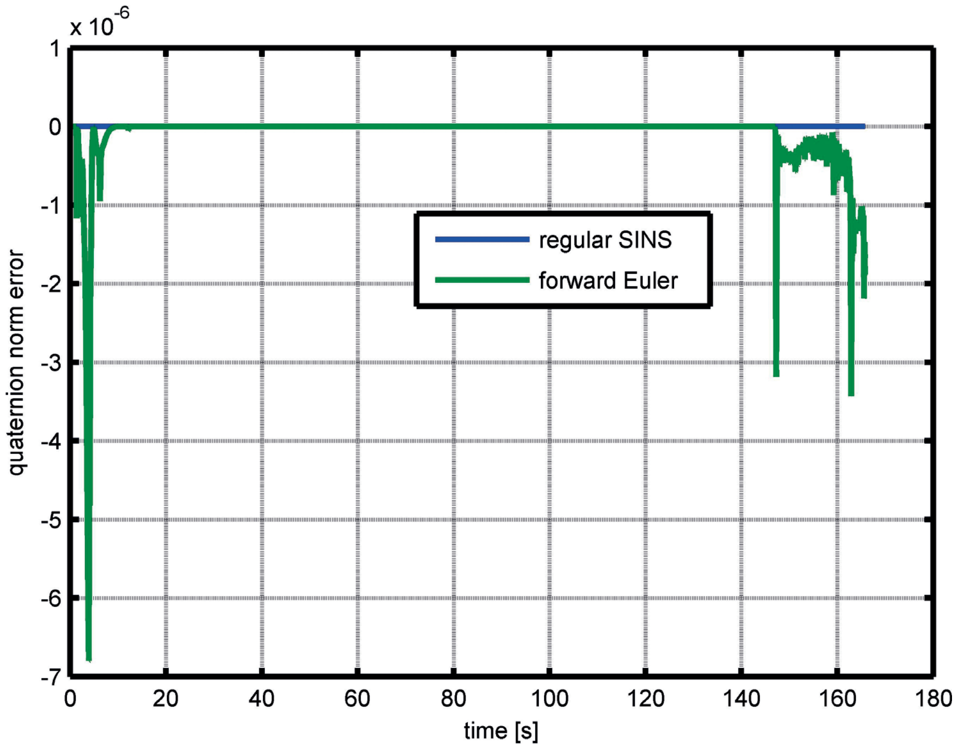


Figure 8. Quaternion norm error comparison.

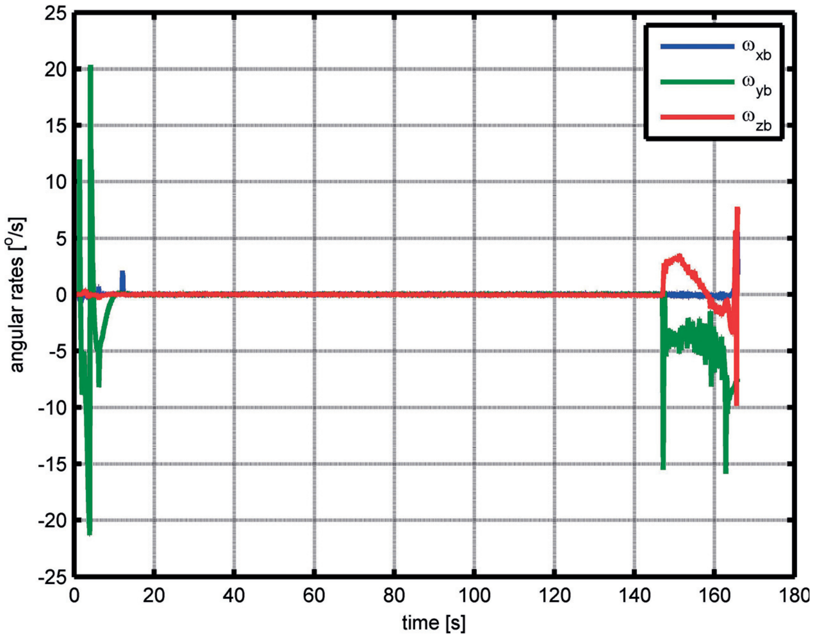


Figure 9. Input data from gyroscopes used for simulation.

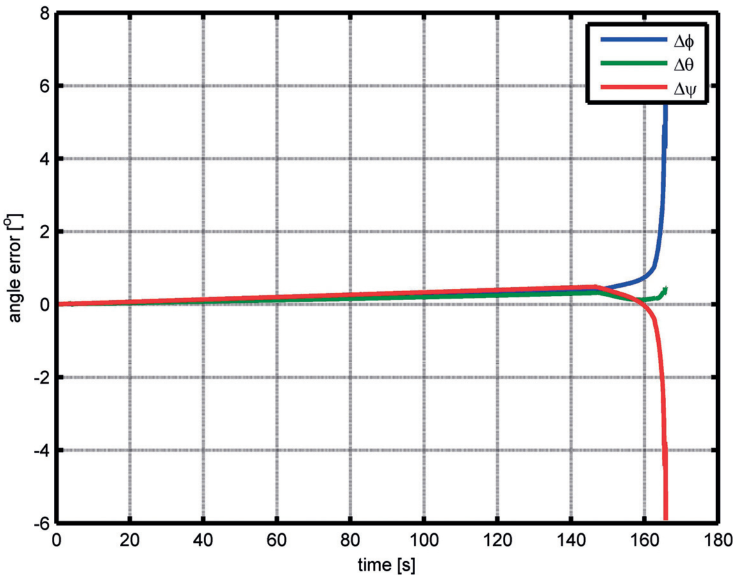


Figure 10. Angle error accumulation in time.

4. Time consumption and processor load comparison of the regular SINS, the divided SINS and the forward Euler algorithms

Forward Euler algorithm, regular SINS algorithm and SINS algorithm divided into eight different steps presented here were compared in terms of processor load and time it takes for all necessary calculations to complete. PC with Intel Core 2 Duo P8600 processor and 4 GB of RAM was used as a testbed for comparison of the three mentioned algorithms. Ubuntu 16.04 LTS operating system in real-time mode was used for time measurements and result generation.

Instead of using real sensors to feed the data to the algorithm, the data were read from the files that contained recorded sensor outputs from INS tests previously performed. All the data were memory mapped to avoid any loss of time due to IO operations, thus making the algorithm exclusively CPU bound. Real-time interval timer set to 2 ms was used as the time frame generator for the INS algorithm in order to mimic real-life operation. Every 2 ms, an interrupt would occur causing the next piece of data to be fed to the algorithm, and the next step of the algorithm would be performed. In the case of the regular SINS algorithm, the entire quaternion calculation will be performed in every eighth step. In the case of the divided SINS algorithm, a piece of that calculation will be calculated in all of those seven middle steps as well as in the final eighth step, thus optimizing processor load and dividing calculation time across all steps in the algorithm evenly.

Statistics that are compared after the completion of the two SINS algorithms are the total time spent in every eight steps of the algorithm, average amount of time spent in every step and average processor load in each of the steps of the algorithm. Total time spent in every step of the algorithm depends on the number of steps and as such is not important as a performance measure. Average time and average processor load in each step of the algorithm are used for performance comparison. In Linux, there are three distinct time measures of process execution. Those are wall clock time, user time and system time.

Wall clock time is the amount of calendar time that elapsed from starting the process or the stopwatch until moment “now”. Thus, wall clock time includes the time the process has spent waiting for its turn on the CPU besides the time it actually spent running on the CPU. User time is the time the process spent executing on the CPU in user mode, while system time is the time the process spent executing on the CPU in system or kernel mode. User and system time measure the actual time the process spent using the CPU, and total amount of time spent on the CPU is calculated as the sum of these two time measurements.

All of these considered, wall clock represents the time that would be measured using a stopwatch. Although wall clock time heavily relies on the operating system load, on the scheduling policy used by the operating system and on the number of cores the CPU has, it can be used as a measure of time since all versions of the algorithm are subjected to the same conditions during the testing procedure. Even though the wall clock time is measured, it is not actually used in time comparison of the two mentioned algorithms. Instead, user and system time are used for comparison, because they rely only on the performance of the CPU, and the actual time it takes for calculations in the algorithm is the sum of these two times.

All times mentioned are given in microseconds. Total amount of time spent in the step of the algorithm is calculated as the sum of user and system time. Average processor load is calculated as

$$PL = \frac{\overline{u} + \overline{s}}{t} \tag{28}$$

where $\overline{u}$ is the average user time, $\overline{s}$ is the average system time, t is the time sample duration.

The results obtained after time measurements of the regular SINS algorithm are presented in **Table 2**.

The results obtained after time measurements of the divided SINS algorithm are presented in **Table 3**.

Results presented in the tables are not comparable to the execution times on faster or slower processors. Even though exact times are not comparable when a switch to a different CPU is made, their ratio will still hold. Relative time gain and processor load gain of the divided SINS over the regular SINS are presented in **Table 4**.

For the sake of completeness, forward Euler version of the SINS algorithm that performs all calculations in each timer interrupt was also taken into consideration. Every 2 ms both quaternions are calculated as well as navigation parameters. Basically, this approach has no notable steps, so the previous method of time measurement is not applicable here. Instead, the average time necessary for the calculation of both quaternions and navigation parameters is taken as the performance measure.

On average, it takes 6.16023 μs for the forward Euler algorithm to perform all calculations. This translates to average processor load of 0.00308 which is significantly worse compared to

Algorithm step	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Elapsed time (μs)	2.07395	2.17831	3.11257	2.71898	2.04108	2.00986	2.06820	4.82662
Processor load	0.00104	0.00109	0.00156	0.00136	0.00102	0.00100	0.00103	0.00241

Table 2. Measured time of the regular SINS.

Algorithm step	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Elapsed time (μs)	2.08052	2.19571	3.09868	3.07395	2.07148	2.10682	2.27033	2.35067
Processor load	0.00104	0.00109	0.00154	0.00153	0.00103	0.00105	0.00113	0.00115

Table 3. Measured time of divided SINS.

Algorithm step	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Time (%)	+0.32	+0.80	−0.45	+13.05	+1.49	+4.82	+9.77	−51.30
Processor load (%)	+0.00	+0.00	−1.3	+12.5	+0.98	+5.00	+9.71	−52.28

Table 4. Regular and divided SINS time and processor load ratio.

Algorithm step	0	1	2	3	4	5	6	7
RSINS T (%)	−66.33	−64.64	−49.47	−55.86	−66.87	−67.37	−66.43	−21.65
RSINS PL (%)	−66.23	−64.61	−49.35	−55.84	−66.88	−67.35	−66.56	−21.75
DSINS T (%)	−66.23	−64.36	−49.70	−50.01	−66.37	−65.80	−63.14	−61.84
DSINS PL (%)	−66.23	−64.61	−50.00	−50.32	−66.56	−65.91	−63.31	−62.66

Table 5. Regular and divided SINS processor load and time gains over the forward Euler algorithm.

the regular SINS algorithm and even more so compared to the divided SINS algorithm. Processor load and time spent calculating in each step of the regular and the divided SINS algorithms vary from step to step, whereas the time it takes for the forward Euler algorithm to do its calculations can be considered as constant. Relative time gain (T) and processor load gain (PL) of the divided SINS (DSINS) and the regular SINS (RSINS) algorithm over the forward Euler algorithm are presented in **Table 5**.

5. Conclusion

In this chapter, navigation algorithm based on strapdown inertial navigation system algorithm optimized for coding in eight steps is presented. This algorithm proved to be a good option in situations where time and processor speed are limiting factors. Average time necessary for the regular SINS algorithm to complete all the steps and perform one full calculation is 21.02957 μs, whereas the divided SINS algorithm needs 19.24816 μs to perform the same operation, which scales to 8.47% improvement in time consumption. Even more important than time consumption improvement is the processor load in each timer interval, which is more uniformly distributed across all the steps in the divided SINS algorithm. Uniformly distributed processor load allows for easier design and development of multithreaded applications, as well as more free resources for the control computer to gather information about its surroundings and to issue commands to other devices in the control chain accordingly.

Also this algorithm proved to be mathematically more stable in term of quaternion norm, which mean that there is less error in angle computation and cumulatively in trajectory calculation.

Author details

Ivana Todić[1]* and Vladimir Kuzmanović[2]

*Address all correspondence to: itodic@mas.bg.ac.rs

1 Faculty of Mechanical Engineering, University of Belgrade, Belgrade, Republic of Serbia

2 Faculty of Mathematics, University of Belgrade, Belgrade, Republic of Serbia

References

[1] Titterton DH, Weston JL. Strapdown Inertial Navigation Technology. IEE Radar, Sonar, Navigation and Avionic Series. Vol. 17. 2nd ed. 2004. 576 p. ISBN: 0-86341-358-7

[2] Salychev O. Inertial Systems in Navigation and Geophysics. 1st ed. Moscow: Bauman MSTU Press; 1998. 352 p. ISBN: 5-7038-1346-8

[3] NIMA TR8350.2: Department of Defense World Geodetic System 1984, Its Definition and Relationship with Local Geodetic Systems. 3rd ed

[4] AIAA R-004: "Atmospheric and Space Flight Vehicle Coordinate Systems". American Institute of Aeronautics and Astronautics (AIAA); 1992. 69 p

[5] Salychev O. Applied Inertial Navigation: Problems and Solutions. Moscow: Bauman MSTU Press; 2004. 306 p. ISBN: 5-7038-2395-1

[6] Savage PG. Strapdown inertial navigation integration algorithm design Part1: Attitude algorithms. Journal of Guidance, Control, and Dynamics. 1998;**21**(1):19-28

[7] Savage PG. Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms. Journal of Guidance, Control, and dynamics. 1998;**21**(2):208-221

[8] Siouris GM. Missile Guidance and Control Systems. 1st ed. New York: Springer; 2004. 666 p. ISBN: 0-387-00726-1