Extended Kalman Filter for ECEF Satelite

January, 2025

Author:
Pawel Wiktor

Email:
pawelwiktor.kontakt@gmail.com

GitHub:
https://github.com/pawelwiktor-github

Table of Contents

1 Introduction

The aim of this project is to present the implementation of the Extended Kalman Filter (EKF) and analyze its effectiveness in estimating the position and velocity of a satellite based on measurements from the GPS system and the mission control center. The Extended Kalman Filter is an extension of the classical Kalman Filter, which enables state estimation in nonlinear systems. Its application in space is particularly important as it allows for precise tracking of satellite trajectories, even in the presence of noise and other disturbances.

The equations of satellite motion in the ECEF (Earth-Centered, Earth-Fixed) frame describe changes in the satellite's position and velocity, taking into account both gravitational forces and the influence of satellite engines. Additionally, the model incorporates Wiener process noise, which is typical for this type of dynamic system. Satellite position measurements are performed by the GPS system, whose accuracy is limited by noise, posing a challenge for the estimation process.

In the first phase of the project, the goal is to implement the Extended Kalman Filter, which will perform both the prediction and correction steps in real-time, based on available measurements. The state estimation process will be based on solving the system of differential equations describing changes in the satellite's state. In particular, equations related to the dynamic correction of state estimation based on measurement data will be adopted, taking into account the Jacobian matrix and noise processes.

In the later stages of the project, an analysis of estimation quality will be conducted based on the computation of the average estimation error for the satellite's position and velocity, as well as a comparison of the results obtained using the Extended Kalman Filter with actual data. The project aims not only to present the theoretical foundations of the Kalman Filter but also to carry out its practical implementation and perform simulation tests in the context of satellite motion monitoring.

2 Mathematical Equations

This section presents the mathematical equations describing the motion of the satellite and its observation. These equations take into account gravitational forces, the influence of Earth's motion, and measurement errors. Based on these, the Extended Kalman Filter was designed for satellite state estimation.

2.1 Satellite Motion Equations

The equations of satellite motion in the ECEF coordinate system describe the relationships between the satellite's position and velocity, considering factors of circular and Earth-orbital motion.

$$\dot{x}_1 = x_4\,dt, \tag{1}$$
$$\dot{x}_2 = x_5\,dt, \tag{2}$$
$$\dot{x}_3 = x_6\,dt, \tag{3}$$
$$\dot{x}_4 = (2\omega x_5 + \omega^2 x_1 + F_1(x))\,dt + F_{t,1} + g_1 dw_1, \tag{4}$$
$$\dot{x}_5 = (-2\omega x_4 + \omega^2 x_2 + F_2(x))\,dt + F_{t,2} + g_2 dw_2, \tag{5}$$
$$\dot{x}_6 = F_3(x)\,dt + F_{t,3} + g_3 dw_3, \tag{6}$$

where:

x_1 – satellite position along the X-axis $[km]$
x_2 – satellite position along the Y-axis $[km]$
x_3 – satellite position along the Z-axis $[km]$
x_4 – satellite velocity along the X-axis $[\frac{km}{s}]$
x_5 – satellite velocity along the Y-axis $[\frac{km}{s}]$
x_6 – satellite velocity along the Z-axis $[\frac{km}{s}]$
ω – angular velocity of Earth's rotation $[\frac{rad}{s}]$
w_i – standard Wiener processes (where $i = 1, 2, 3$)
g_i – noise intensity (where $i = 1, 2, 3$)
$F_i(x)$ – gravitational acceleration at a given point (where $i = 1, 2, 3$)
$F_{t,i}$ – acceleration produced by thrusters (where $i = 1, 2, 3$)

2.2 Equations Describing Accelerations in Earth-Orbital Motion

The gravitational acceleration at the point $x = (x_1, x_2, x_3)^T$ is:

$$F(x) = F_1(x) + F_2(x) + F_3(x) \tag{7}$$

where:

$$F_{1,i}(x) = -\frac{\mu x_i}{r^3}, \quad i = 1, 2, 3, \tag{8}$$

$$F_{2,1}(x) = J_2 \frac{x_1}{r^7} \left(6x_3^2 - 3(x_1^2 + x_2^2)\right), \tag{9}$$

$$F_{2,2}(x) = J_2 \frac{x_2}{r^7} \left(6x_3^2 - 3(x_1^2 + x_2^2)\right), \tag{10}$$

$$F_{2,3}(x) = J_2 \frac{x_3}{r^7} \left(3x_3^2 - 9(x_1^2 + x_2^2)\right), \tag{11}$$

$$F_{3,1}(x) = J_3 \frac{x_1 x_3}{r^9} \left(10x_3^2 - 15(x_1^2 + x_2^2)\right), \tag{12}$$

$$F_{3,2}(x) = J_3 \frac{x_2 x_3}{r^9} \left(10x_3^2 - 15(x_1^2 + x_2^2)\right), \tag{13}$$

$$F_{3,3}(x) = J_3 \frac{1}{r^9} \left(4x_3^2(x_3^2 - 3(x_1^2 + x_2^2)) + 3(x_1^2 + x_2^2)^2\right), \tag{14}$$

where:

- μ – gravitational parameter $[km^3/s^2]$
- r – distance from Earth's center at a given point $[km]$
- J_2 – Earth's quadrupole flattening
- J_3 – Earth's octupole flattening

2.3 Observation Equation Including Measurement Errors

The satellite's position is observed at times t_k (not necessarily equidistant), with a measurement error of $\sigma_v = 3$ m. The observation equation is:

$$y(t_k) = Cx(t_k) + v_k, \tag{15}$$

where:

- $C = [I_3, 0_{3\times3}]$
- $v_k \sim N(0, S_{v,k})$
- $S_{v,k} = \sigma_v^2 I_3$

2.4 Prediction Step for the Extended Kalman Filter

Equations (1-6), used for designing the Extended Kalman Filter, can be rewritten as:

$$dx = f(t, x)\, dt + g\, dw, \tag{16}$$

where functions f and g directly follow from the satellite motion equations (1-6).

It is assumed that measurements from the Mission Control Center (MCC) are performed with high accuracy. Therefore, the initial position m_0 is taken from the first position measurement, and S_0 is set to a value close to zero (but non-zero).

The system of equations for the prediction step in the time interval $[t_{k-1}, t_k)$ is:

$$\dot{m}(t) = f(m(t)), \quad m(t_{k-1}) = m_{k-1}, \quad t \in [t_{k-1}, t_k), \tag{17}$$

$$\dot{S}(t) = A(m(t))S(t) + S(t)A(m(t))^T + gg^T, \quad S(t_{k-1}) = S_{k-1}, \tag{18}$$

where:

- t_{k-1} – end time of estimation
- $\dot{m}(t)$ – state estimate at a given time
- $\dot{S}(t)$ – error covariance matrix at a given time
- A – Jacobian matrix
- gg – noise intensity matrix

2.5 Jacobian Matrix

The Jacobian matrix A is defined as:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \omega^2 + \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \frac{\partial F_1}{\partial x_3} & 0 & 2\omega & 0 \\ \frac{\partial F_2}{\partial x_1} & \omega^2 + \frac{\partial F_2}{\partial x_2} & \frac{\partial F_2}{\partial x_3} & -2\omega & 0 & 0 \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & \frac{\partial F_3}{\partial x_3} & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

2.6 Correction Step for the Extended Kalman Filter

At the moment of measurement, the state estimate and the error covariance matrix at time t_k are:

$$m_k^- = m(t_k^-), \quad S_k^- = S(t_k^-), \tag{20}$$

and the correction step is performed as follows:

$$\Sigma_k = S_{v,k} + C S_k^- C^T, \tag{21}$$

$$L_k = S_k^- C^T \Sigma_k^{-1}, \tag{22}$$

$$S_k = S_k^- - L_k \Sigma_k L_k^T, \tag{23}$$

$$m_k = m_k^- + L_k(y_k - C m_k^-). \tag{24}$$

where:

Σ_k – measurement error covariance
L_k – Kalman gain matrix
S_k – updated error covariance matrix
m_k – updated state estimate

2.7 Quality Indicators

The average position estimation error is:

$$\sigma_x^2 = \frac{1}{N} \sum_{k=1}^{N} s_k \left\| C m_k - y_k \right\|^2, \tag{25}$$

where:

N – number of measurements
s_k – measurement weight (=1 for MCC measurements)
C – observation matrix
m_k – state estimate at time t_k
y_k – actual position measurement

The average velocity estimation error is:

$$\sigma_v^2 = \frac{1}{N} \sum_{k=1}^{N} s_k \left\| C_v m_k - v_k \right\|^2, \tag{26}$$

where:

v_k – actual satellite velocity at time t_k

3 Auxiliary Functions

To enhance the clarity of the project, the program has been divided into auxiliary functions that perform specific tasks to improve the operation of the Extended Kalman Filter.

3.1 Function: matrix_to_vector

```
function x_mtv = matrix_to_vector(x, S)

    % Function rewrites a matrix into a vector

    % Arguments:
    % x - state vector
    % S - covariance matrix

    % Outputs:
    % x_mtv - combined state vector

    % Description:
    % Function rewrites a matrix into a vector. Input matrix is a
    % symetrical matrix relative to the diagonal and it is rewrited with
    % diagonal values as well as components under the diagonal.
    %
    % Dimensions of the input values:
    % x = [pX;pY;pZ;vX;vY;vZ] (6x1)
    % p - position value, v - velocity value in specified axis
    % S = (6x6)
    % symetrical matrix relative to the diagonal consisting of prediction error
    %
    % Dimensions of the output values:
    % x_mtv = (27x1)
    % first 6 rows stands for state vector, remaining 21 rows stand for
    % rewrited covariance matrix
    %
    % Wiktor Pawel, 01.17.2025

    %% Execution
    % Vector initialisation
    x_mtv = zeros(27, 1);

    x_mtv(1:6) = x; % Transcription of estimates
    ind = 7; % First index for matrix in vector
        for i = 1:6
            for j = i:6
                x_mtv(ind) = S(i, j);
                ind = ind + 1;
            end
        end
end
```

Function 1. Transcription of a matrix into a vector

3.2 Function: vector_to_matrix

```
function [x_vtm, S_vtm] = vector_to_matrix(x)

    % Function rewrites a vector into a matrix

    % Arguments:
    % x - combined state vector

    % Outputs:
    % x_vtm - state vector
    % S_vtm - covariance matrix

    % Description:
    % Function rewrites a vector into a matrix. Input vector contains 21
    % rows omitting the state vector, and it stands for diagonal values as
    % well as components under the diagonal for a covariance matrix.
    %
    % Dimensions of the input values:
    % x = (27x1)
    % first 6 rows stands for state vector, remaining 21 rows stand for
    % covariance matrix elements
    %
    % Dimensions of the output values:
    % x_vtm = [pX;pY;pZ;vX;vY;vZ] (6x1)
    % p - position value, v - velocity value in specified axis
    % S_vtm = (6x6)
    % symetrical matrix relative to the diagonal consisting of prediction error
    %
    % Wiktor Pawel, 01.17.2025

    %% Execution
    x_vtm = x(1:6); % Transcription of estimates
    S_vtm = zeros(6, 6); % Matrix initialisation

    % Filling the matrix
    ind = 7; % First index from vector
    for i = 1:6
        for j = i:6
            S_vtm(i, j) = x(ind);
            ind = ind + 1;
        end
    end

    % Matrix symmetrization
    S_vtm = S_vtm + S_vtm' - diag(diag(S_vtm));
end
```

Function 2. Transcription of a vector into a matrix

3.3 Function: rhs

```
function dx=rhs(t,x)
  % The function calculates the derivatives of the state equations

  % Arguments:
  % t - time
  % x - current state vector

  % Outputs:
  % dx - derivative of a state vector

  % Description:
  % Function calculates the derivatives of the state equations according
  % to the equations in the attached documentation.
  %
  % Dimensions of the input values:
  % x = [pX;pY;pZ;vX;vY;vZ] (6x1)
  % p - position value, v - velocity value in specified axis
  % t = [t] (1x1)
  % current time stamp
  % Dimensions of the output values:
  % dx = (6x1)
  % derivatives by every state vector element
  %
  % Wiktor Pawel, 01.17.2025

  %% Execution
  % Constans definition
  mu=398600.4415; % Gravitational parameter [km^3*s^-2]
  Re=6378.1363; % Earth's radius [km]
  Te=86164.09092; % Period of Earth's rotation [s]
  J_2=-0.1082635854*1e-2; % Calculation factor
  J_3=0.2532435346*1e-5; % Calculation factor
  J2=-mu*J_2*Re^2; % Quadrupole flattening of the Earth
  J3=-mu*J_3*Re^3; % Octupole flattening of the Earth
  omega=2*pi/Te; % Angular velocity of the Earth's rotation [rad/s].
  omega_square=omega*omega; % Squared angular velocity of the Earth's rotation

  % State derivative initialization
  dx=zeros(6,1);

  % Auxiliary calculations
  % Gravitational acceleration at a point
  r2=x(1:3)'*x(1:3);
  r=sqrt(r2);
  r3=r2*r;
  r7=r^7;
  r9=r7*r2;
  F1=-mu*x(1:3)/r3;
  d=x(1)^2+x(2)^2;
  x32=x(3)^2;
  u=6*x32-1.5*d;
  F2=J2*x(1:3)/r7;
  F2=F2.*[u;u;3*x32-4.5*d];
  F3=J3*[x(1)*x(3);x(2)*x(3);1]/r9;
  u=10*x32-7.5*d;
  F3=F3.*[u;u;4*x32*(x32-3*d)+1.5*d^2];
  F=F1+F2+F3;

  % Drag forces (in this example not taken into calcualtions)
  Cd=0.0;
  v=x(4:6);
  av=sqrt(v'*v);
  Fd=-Cd*av*v;

  % State equations of a satellite motion
  dx(1:3)=x(4:6);
  dx(4)=2*omega*x(5)+omega_square*x(1)+F(1)+Fd(1);
  dx(5)=-2*omega*x(4)+omega_square*x(2)+F(2)+Fd(2);
  dx(6)=F(3)+Fd(3);
end
```

Function 3. Calculation of the derivatives of the equations of state

3.4 Function: get_jacob

```
function A=get_jacob(t,x)
  % The function calculates the Jacobi matrix for the state vector

  % Arguments:
  % t - time
  % x - state vector

  % Outputs:
  % A - Jacobi matrix

  % Description:
  % Function calculates the Jacobi matrix coefficients and performs its
  % filling according to the equations in the attached documentation.
  %
  % Dimensions of the input values:
  % x = [pX;pY;pZ;vX;vY;vZ] (6x1)
  % p - position value, v - velocity value in specified axis
  % t = [t] (1x1)
  % current time stamp
  % Dimensions of the output values:
  % A = (6x6)
  % with coefficients filled according to the doc
  %
  % Wiktor Pawel, 01.17.2025

  %% Execution
  % Constans definition
  mu=398600.4415; % Gravitational parameter [km^3*s^-2]
  Re=6378.1363; % Earth's radius [km]
  Te=86164.09092; % Period of Earth's rotation [s]
  J_2=-0.1082635854*1e-2; % Calculation factor
  J_3=0.2532435346*1e-5; % Calculation factor
  J2=-mu*J_2*Re^2; % Quadrupole flattening of the Earth
  J3=-mu*J_3*Re^3; % Octupole flattening of the Earth
  omega=2*pi/Te; % Angular velocity of the Earth's rotation [rad/s].
  db_omega=2*omega; % Doubled angular velocity of the Earth's rotation
  omega_square=omega*omega; % Squared angular velocity of the Earth's rotation

  % Filling right half of the matrix
  A=zeros(6);
  A(1,4)=1;
  A(2,5)=1;
  A(3,6)=1;
  A(4,5)=db_omega;
  A(5,4)=-db_omega;

  % Auxiliary calculations for r
  r2=x(1:3)'*x(1:3);
  r=sqrt(r2);
  r3=r2*r;
  r7=r^7;
  r9=r7*r2;

  % Auxiliary calculations for x
  x12=x(1)^2;
  x22=x(2)^2;
  x32=x(3)^2;
  x13=x(1)*x(3);
  x23=x(2)*x(3);

  % Calculations for auxiliary variables
  % d
  d=x12+x22;
  % u
  u=6*x32-1.5*d;
  u1=10*x32-7.5*d;

  % q
  q2=[x(1)*u;x(2)*u;x(3)*(3*x32-4.5*d)];
  q3=[x13*u1;x23*u1;4*x32*(x32-3*d)+1.5*d^2];
  q2m(1,1)=u-3*x12;q2m(1,2)=-3*x(1)*x(2);q2m(1,3)=12*x13;
  q2m(2,1)=q2m(1,2);q2m(2,2)=u-3*x22;q2m(2,3)=12*x23;
  q2m(3,1)=-9*x13;q2m(3,2)=-9*x23;q2m(3,3)=9*x32-4.5*d;
  q3m(1,1)=x(3)*u1-15*x12*x(3);q3m(1,2)=-15*x(1)*x23;q3m(1,3)=x(1)*u1+20*x(1)*x32;
  q3m(2,1)=-15*x(1)*x23;q3m(2,2)=x(3)*u1-15*x22*x(3);q3m(2,3)=x(2)*u1+20*x(2)*x32;
  q3m(3,1)=6*x(1)*(d-4*x32);q3m(3,2)=6*x(2)*(d-4*x32);q3m(3,3)=16*x(3)*(x32-1.5*d);

  % Matrix F initialization (gravitational acceleration at point x)
  F=zeros(3,3);

  % Calculation of coefficients for subsequent values of F(i,j)
  c1r=-mu/r3;
  c2r=J2/r7;
  c3r=J3/r9;

  % Calculation of matrix F
  for i=1:3
      for j=1:3
          del_ij=0;if i==j, del_ij=1;end
          F(i,j)=c1r*(del_ij-3*x(i)*x(j)/r2);
          F(i,j)=F(i,j)+(q2m(i,j)-7*q2(i)*x(j)/r2)*c2r;
          F(i,j)=F(i,j)+(q3m(i,j)-9*q3(i)*x(j)/r2)*c3r;
      end
  end
  F(1,1)=F(1,1)+omega_square;
  F(2,2)=F(2,2)+omega_square;

  % Jacobi matrix completion
  A(4:6,1:3)=F;
end
```

Function 4. Calculation of the Jacobi matrix for the state vector

3.5 Function: rhs_ekf

```
function dx = rhs_ekf(t, x)

    % The function returns the derivative of the combined state vector

    % Arguments:
    % x - combined, current state vector (27x1) [m(:); S(:)]
    % t - time

    % Outputs:
    % dx - derivative of a state vector

    % Description:
    % Function calculates the derivatives of the state vector and matrix covariance according
    % to the equations in the attached documentation.
    %
    % Dimensions of the input values:
    % x = (27x1)
    % first 6 rows stands for state vector, remaining 21 rows stand for
    % covariance matrix elements
    % t = [t] (1x1)
    % current time stamp
    % Dimensions of the output values:
    % dx = (27x1)
    % derivatives by every state vector element as well as every coefficient at diagonal
    % and components under the diagonal covariance matrix
    %
    % Wiktor Pawel, 01.17.2025

    %% Execution
    % Constants definition
    g = 4.52*1e-5; % Noise intensity
    D = g*g'; % Diffusion matrix

    % Transcription of vector to matrix
    [m, S] = vector_to_matrix(x);

    % Calculation for state vector (m)
    dm = rhs(0, m);

    % Calculation for covariance matrix (S)
    A = get_jacob(0, m);
    dS = A * S + S * A' + D;

    % Transcription of matrix to vector
    dx = matrix_to_vector(dm, dS);
end
```

Function 5. Calculation of the derivative of a connected state vector

3.6 Function: rk4

```
function [t,x]=rk4(x0,tf,SPTu)

    % The function performs RK4 integration

    % Arguments:
    % x0 - current state vector
    % tf - time window over which the integration will be calculated
    % SPTu - number of steps per time unit

    % Outputs:
    % x - integrated state vector
    % t - time of integration

    % Description:
    % Function calculates the integral of state vector based on
    % Rungego-Kutty method (4th row).
    %
    % Dimensions of the input values:
    % x0 = (27x1)
    % first 6 rows stands for state vector, remaining 21 rows stand for
    % covariance matrix elements
    % t = [t] (1x1)
    % current time stamp
    % Dimensions of the output values:
    % t = [t] (1x1)
    % time of integration
    % x = (27x1)
    % integral by every state vector element as well as every coefficient at diagonal
    % and components under the diagonal covariance matrix
    %
    % Wiktor Pawel, 01.17.2025

    %% Execution
    % Basic calculation
    n = length(x0);
    nt = 1 + floor(tf * SPTu);
    h = tf / nt;
    h_2 = h / 2; h_6 = h / 6; h_26 = 2 * h_6;

    % Parameters initialization
    x0=x0(:);
    x=zeros(nt+1,n);
    t=zeros(nt+1,1);
    x(1,:)=x0';
    tmp=zeros(n,1);
    xtmp=x0;tt=0;
    dx1=zeros(n,1);
    dx2=zeros(n,1);
    dx3=zeros(n,1);
    dx4=zeros(n,1);

    for i=1:nt
        dx1=rhs_ekf(tt,xtmp);tmp=xtmp+h_2*dx1;tt=tt+h_2;
        dx2=rhs_ekf(tt,tmp);tmp=xtmp+h_2*dx2;
        dx3=rhs_ekf(tt,tmp);tmp=xtmp+h*dx3;tt=tt+h_2;
        dx4=rhs_ekf(tt,tmp);
        xtmp=xtmp+h_6*(dx1+dx4)+h_26*(dx2+dx3);
        x(i+1,:)=xtmp';t(i+1)=tt;
    end
end
```

Function 6. Numerical calculation with the RK4 method/Prediction

3.7 Function: correction

```
function [m_c, S_c] = correction(m, S, yv, Svk, C)

    % The function performs a correction step based on the last GPS measurement

    % Arguments:
    % m - current value of the state vector prediction
    % S - current value of the covariance matrix
    % yv - the measurement on the basis of which the correction is performed
    % Svk - measurement noise matrix
    % C - unit matrix identifying the state variable / state output matrix

    % Outputs:
    % m_c - updated state vector
    % S_c - updated covariance matrix

    % Description:
    % Function performs a correction update after predictions based on
    % numerical integration (RK4 method). It corrects the latest state
    % vector and covariance matrix in main loop.
    %
    % Dimensions of the input values:
    % m = [pX;pY;pZ;vX;vY;vZ] (6x1)
    % p - position prediction, v - velocity prediction in specified axis
    % S = (6x6)
    % symetrical matrix relative to the diagonal consisting of prediction error
    % yv = [mpX;mpY;mpZ;mvX;mvY;mvZ] (6x1)
    % mp - measured position, v - measured velocity in specified axis
    % Svk = [pErr * eye(3), zeros(3); zeros(3), eye(3) * vErr] (6x6)
    % pErr - measurement noise for position, vErr - measurement noise for velocity
    % C = [eye(3), zeros(3); zeros(3), eye(3)] (6x6)
    % diagonal unit matrix
    %
    % Dimensions of the output values:
    % m_c = [pX;pY;pZ;vX;vY;vZ] (6x1)
    % p - position prediction, v - velocity prediction in specified axis
    % S_c = (6x6)
    % diagonal unit matrix consisting of prediction error
    %
    % Wiktor Pawel, 01.17.2025

    %% Execution
    sigma_k = Svk + C * S * C'; % Covariance matrix of observation prediction error
    L_k = S * C' * inv(sigma_k); % EKF gain
    S_c = S - L_k * sigma_k * L_k';S_c = diag(diag(S_c)); % Update of covariance matrix based on correction
    m_c = m + L_k * (yv - C * m); % Update of state vector based on correction
end
```

Function 7. Estimate correction

4 Measurement Data Analysis

First, the satellite trajectory in the ECEF coordinate system was plotted (Figure 1).

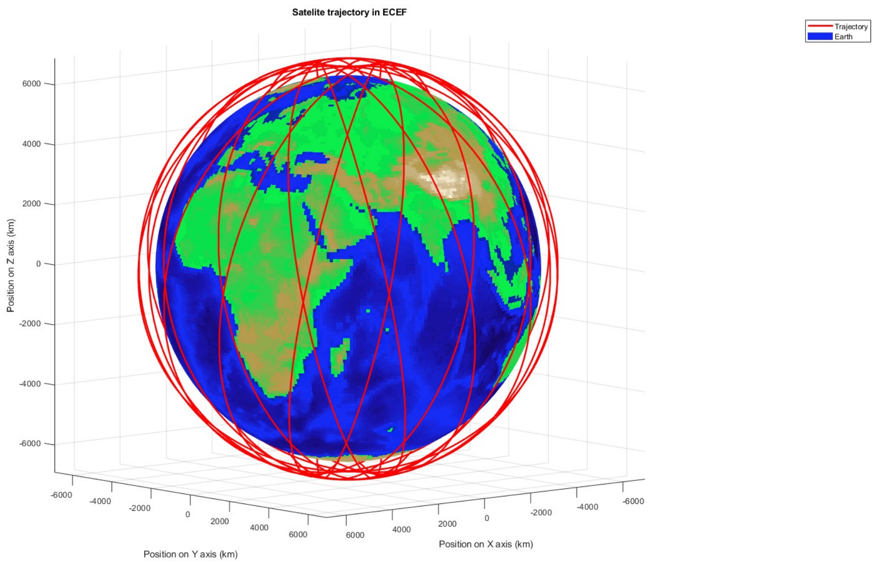

Fig. 1. Satellite trajectory in the ECEF coordinate system

Next, the complete measurement data was analyzed (Figures 2 and 3).

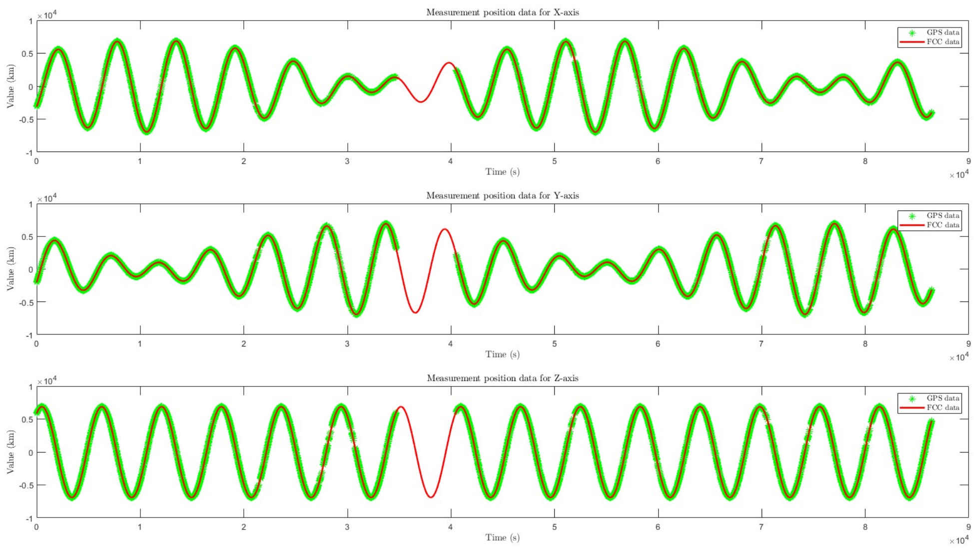

Fig. 2. Satellite position measurement data

The measurement data plot shows the change in the satellite's position over a 24-hour period and the frequency of GPS data availability (green color). A gap in GPS data was also observed between 9:38:17 [hh/mm/ss] and 11:15:01 [hh/mm/ss].

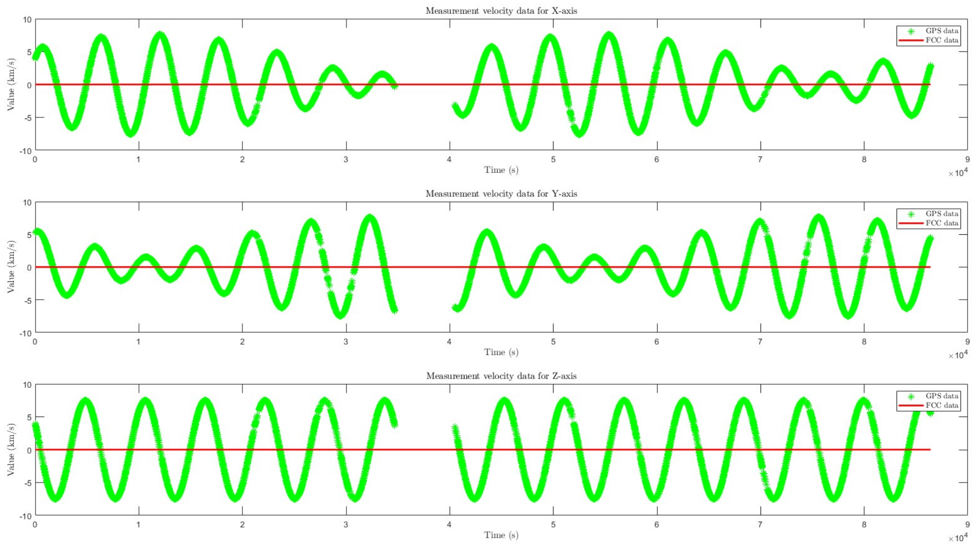

Fig. 3. Satellite velocity measurement data

The velocity measurements originate exclusively from GPS data, as confirmed by Figure 3. Therefore, the correction of the state estimate is performed only after receiving velocity data from the GPS system.

5 Test of the Main Kalman Loop

Chapter five describes the testing of the main Kalman loop on a limited dataset.

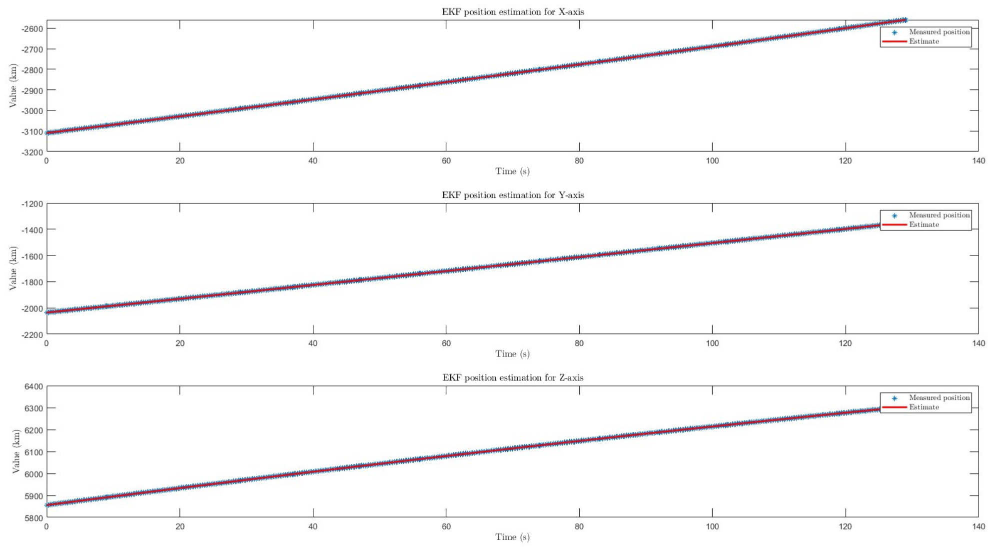

Fig. 4. Testing position estimation using EKF

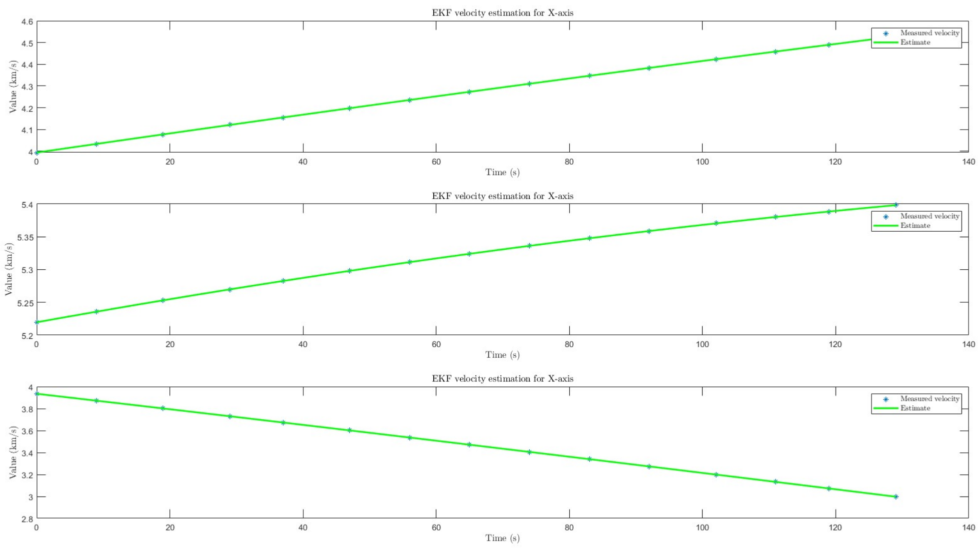

Fig. 5. Testing velocity estimation using EKF

The Extended Kalman Filter was designed as described below:
It performs the prediction step over the time interval $[t_{k-1}, t_k)$ *during measurements from the mission control center, while performing the correction step upon receiving a GPS measurement, based on the latest integration estimate.*

Based on the plots, the estimation performance of the filter can be considered correct.

6 Main Loop of the Extended Kalman Filter

This section describes the implementation of the main loop of the Extended Kalman Filter (EKF), which iteratively estimates the state of the system based on measurements. The process consists of two stages: prediction, where the state is calculated based on previous estimates, and correction, where new measurement data is incorporated. Below is the code implementation of this algorithm.

```
% Extended Kalman Filter implementation

clear; clc;
load 'pos_vel_data.mat'

% Constants definition
Svk = [9e-06 * eye(3), zeros(3); zeros(3), eye(3) * 1e-8]; % Output noise
SPTu = 100; % Number of integration steps per unit time
C = [eye(3), zeros(3); zeros(3), eye(3)]; % State output matrix

m0 = [y(1,:), v(1,:)]; % Initial conditions for state variables
S0 = Svk; % Initial condition for covariance matrix

% Transfer of matrix to vector
x0 = matrix_to_vector(m0, S0);

%% EKF main loop

for i = 2:length(t)
    % Integration time constant
    tf = t(i)-t(i-1);

    % Numerical integration RK4 / Prediction
    [tt,x] = rk4(x0,tf,SPTu);

    % Save last estimate
    x_last = x(end, :)';

    % PTransfer of vector to matrix
    [m, S] = vector_to_matrix(x_last);

    if s(i) == 0
        yv = [y(i, :), v(i, :)]'; % Measurement
        % Correction
        [m_c, S_c] = correction(m, S, yv, Svk, C);
        % Update state vector after correction
        x_last = matrix_to_vector(m_c, S_c);
    end

    % Reset the current state vector for RK4 method
    x0 = x_last;

    % Results
    results(i, :) = x_last(1:6)';

    % Display of EKF progress
    if mod(i, 5000) == 0
        fprintf('Iteration %d completed. There are %d left.\n', i, length(t)-i);
    end
end

% Rewriting the first measurement
results(1, :) = [y(1,:), v(1,:)];

fprintf('EKF parameter estimation completed...\n');
```

Function 8. Extended Kalman Filter

The complete process of data estimation was performed. The results are presented in Figures 6 and 7.

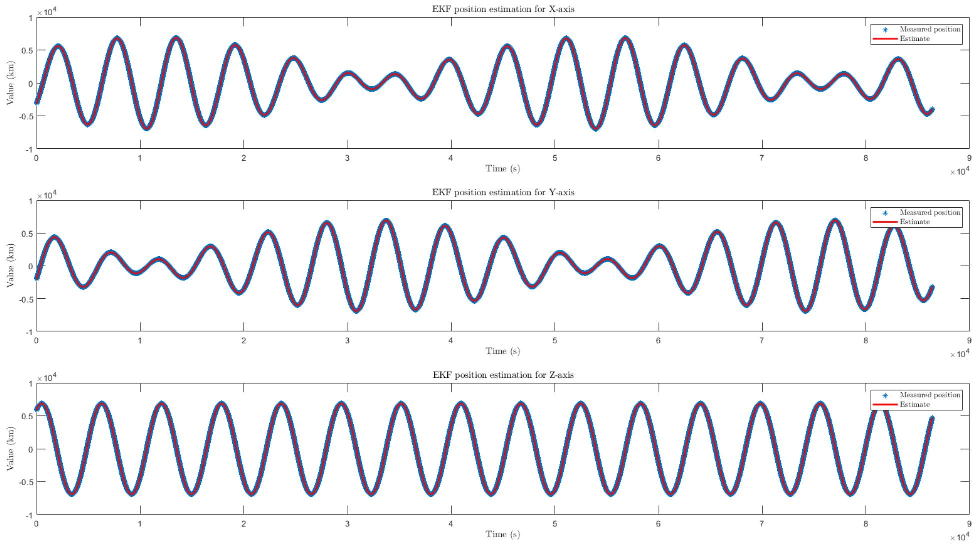

Fig. 6. Position estimation using EKF

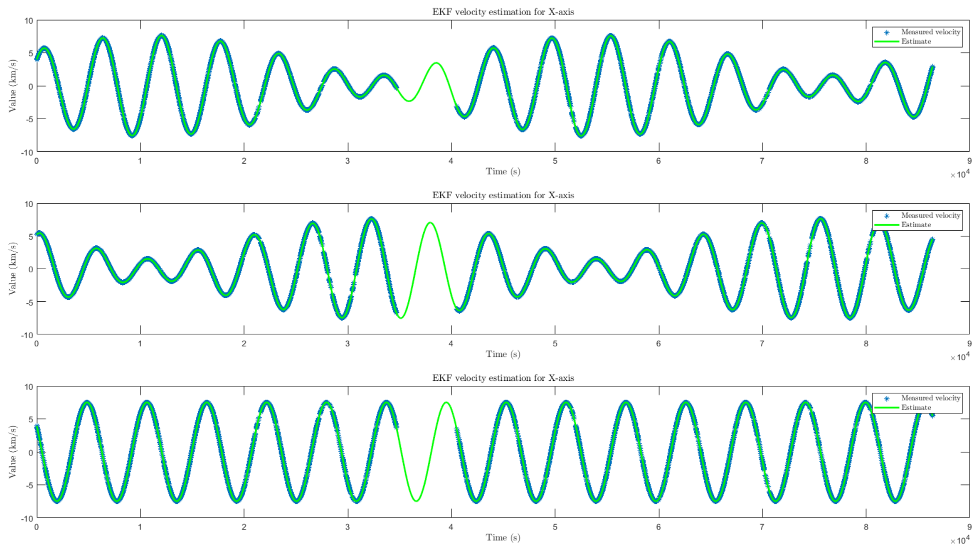

Fig. 7. Velocity estimation using EKF

Next, the quality indicators for position estimation were calculated according to Equation 25. The results are presented in Tables 1 and 2. The differences in position estimation relative to measurement data are shown in Figure 8.

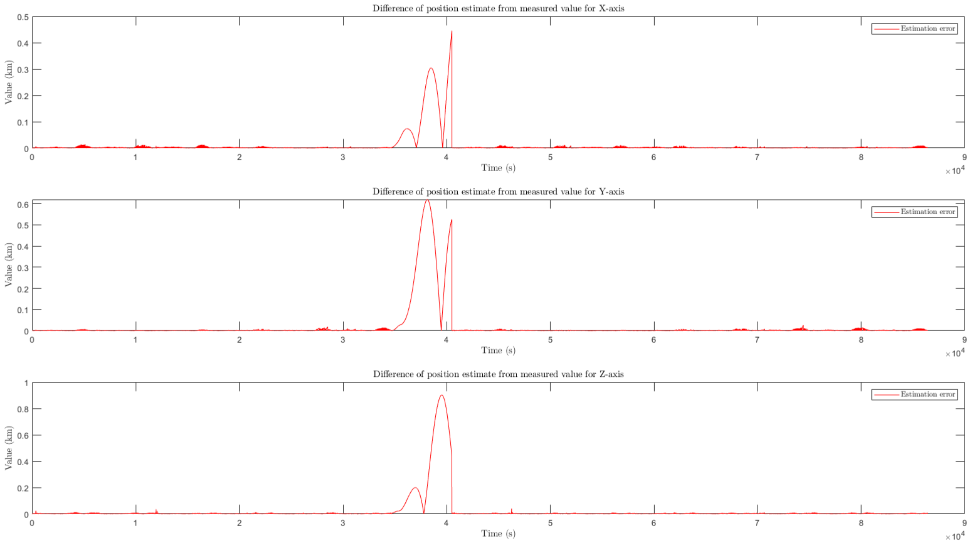

Fig. 8. Difference in position estimation relative to measurement data

Table 1. Average Standard Deviation

Quality Indicator	Value
Average standard deviation of position estimation	$66.104[m]$

Table 2. Standard Deviation

Quality Indicator	Axis	Value
Standard Deviation	X	$35.746[m]$
Standard Deviation	Y	$67.860[m]$
Standard Deviation	Z	$94.704[m]$

In evaluating the quality of the velocity estimation performed by the designed filter, Equation 26 was applied. However, the reference data (from the mission control center) does not include velocity measurements (velocity measurements from the mission control center are zero). Consequently, the results do not represent an assessment of quality because the estimation (a numerical value) is compared to a reference (a zero value).

7 Conclusions

The implementation of the EKF demonstrated high effectiveness in estimating the satellite's position in the ECEF coordinate system, even in the presence of measurement noise and interruptions in GPS data.

The quality of the filter's estimation was evaluated using quality indicators. The designed Extended Kalman Filter maintains the difference between the estimated and reference values at no more than $10[m]$. During GPS data interruptions, the filter is unable to perform state correction, resulting in an increased difference between the estimated and reference values, as shown in Figure 8. The average standard deviation of position estimation was $66.104[m]$.

Data analysis revealed that after estimation correction (upon receiving a measurement), the filter maintains a stable prediction that does not differ from the reference by more than $10[m]$. The more frequently corrections are performed, the more accurately the EKF is able to estimate the satellite's position in the ECEF coordinate system.

Despite its high effectiveness, the model's accuracy depends on input parameters. Further research focusing on improving disturbance suppression through more precise tuning, Jacobian matrix calculations, or numerical integration could enhance the quality of the estimation.