Adaptive Cruise Control System Model

Your Name

June 24, 2023

Contents

Chapter 1

Part 1 (a)

1.1 Introduction

This part presents the derivation of a system model for an Adaptive Cruise Control (ACC) system. The model is based on the longitudinal dynamics of a vehicle and the dynamics of vehicle actuation. The model also considers the travel resistance which includes air drag, rolling resistance, acceleration resistance, and grading resistance.

1.2 Vehicle Dynamics Model

The longitudinal dynamics of a host vehicle is given by the equation:

$$m\dot{v}_h = ma_f - r_{travel} \tag{1.1}$$

where m is the vehicle mass, v_h is the vehicle speed, a_f is the traction force converted to acceleration, and r_{travel} is the travel resistance.

The dynamics of a vehicle actuation including engine, transmission, or brake has nonlinear characteristics. The input/output relationship of the actuation dynamics is described as an ordinary differential equation:

$$\dot{x}_f = f_{act}(x_f, u) \tag{1.2}$$

$$a_f = h_{act}(x_f) \tag{1.3}$$

where $x_f \in \mathbb{R}^{n_f}$, $u \in \mathbb{R}$ are respectively the state, the input of the actuation system. u is an acceleration command, i.e., a control input calculated by adaptive cruise controller. The output of the system is a_f.

The travel resistance r_{travel} contains several factors to resist its motion. A model of the resistive force is expressed as

$$r_{travel} = r_{air}v_h^2 + r_{roll}(v_h) + r_{accel}\dot{v}_h + r_{grad}(\theta) \tag{1.4}$$

1.3 State-Space Model for ACC System

To build a plant model for the ACC system design, two state variables are defined: inter-vehicle distance following error $\Delta d = d - d_r$ and velocity following error $\Delta v = v_p - v_h$. The d_r is determined based on the constant time headway policy given by

$$d_r = T_{hw} v_h + d_0 \tag{1.5}$$

where T_{hw} is the constant time headway and d_0 is the stopping distance for safety margin.

Let us define the state variables of the plant as $x = [x_1 \quad x_2 \quad x_3^T]^T \in \mathbb{R}^{2+n_f}$ with $x_1 = \Delta d$, $x_2 = \Delta v$ and $x_3 = x_f$. Then, the state-space model is formulated as

$$\dot{x} = f(x,u) + Gv + Hw \tag{1.6}$$

$$y = Cx + Jv \tag{1.7}$$

where

$$f(x,u) = \begin{bmatrix} x_2 - T_{hw} x_3 \\ -h_{act}(x_f) \\ f_{act}(x_f, u) \end{bmatrix},$$

$$G = \begin{bmatrix} T_{hw}/m \\ 1/m \\ 0 \end{bmatrix},$$

$$H = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$J = \begin{bmatrix} 0 \\ 0 \\ -1/m \end{bmatrix}$$

where $u \in \mathbb{R}$ and $y = [\Delta d \quad \Delta v \quad \dot{v}_h]^T \in \mathbb{R}^3$ are the input and output of the plant and $v = r_{travel}$ and $w = \dot{v}_p$ represents disturbances into the plant.

1.4 SS Model Representation

The plant model for MPC is simplified to reduce computational cost. The actuation system model is approximated to a switched linear time variant system as follows:

$$\begin{aligned} \dot{x}_f &= A_f(t) x_f + B_f(t) u \\ a_f &= C_f x_f \end{aligned} \tag{1.8}$$

where $x_f \in \mathbb{R}$, $n_f = 1$, and

$$A_f(t) = \begin{cases} -1/T_{eng}, & \text{if } u(t) \geq a_{thr_{off}} \\ -1/T_{brk}, & \text{if } u(t) < a_{thr_{off}} \end{cases}$$

$$B_f(t) = \begin{cases} K_{eng}(t)/T_{eng}, & \text{if } u(t) \geq a_{thr_{off}} \\ K_{brk}(t)/T_{brk}, & \text{if } u(t) < a_{thr_{off}} \end{cases}$$

and $C_f = 1$.

The steady-state gains can be changed to compensate the prediction accuracy. For example, $K_{eng}(t)$ is composed of constant value K_c and compensation signal $\Delta K(t)$ as below.

$$K_{eng}(t) = K_c + \Delta K(t) \tag{1.9}$$

where $\Delta K(t)$ is computed by the filter $\Delta K(s) = L(s)u(s)$ with $L(0) = 0$.

The prediction model can be simplified to the linear time variant system with three states below:

$$\begin{aligned} \dot{x} &= A(t)x + B(t)u + Gv + Hw \\ y &= Cx + Jv \end{aligned} \tag{1.10}$$

where $x \in \mathbb{R}^3$ and

$$A(t) = \begin{bmatrix} 0 & 1 & -T_{hw} \\ 0 & 0 & -1 \\ 0 & 0 & A_f(t) \end{bmatrix}, B(t) = \begin{bmatrix} 0 \\ 0 \\ B_f(t) \end{bmatrix}$$

The travel disturbance model can be written as

$$r_{travel} = r_{accel}\dot{v}_h + r_{travel} \tag{1.11}$$

where the first term is the acceleration resistance and the second term is other resistances $r_{travel} = r_{air}v_h^2 + r_{roll}(v_h) + r_{grad}(\theta)$.

Let us define a state transformation for x_3 as below.

$$\tilde{x}_3 = \frac{1}{1 + r_{accel}/m} x_3 \tag{1.12}$$

From the vehicle dynamics, the acceleration of the host vehicle becomes

$$\dot{v}_h = x_3 - \frac{1}{m} r_{travel} = (1 + r_{accel}/m)\tilde{x}_3 - \frac{1}{m_{travel}} \tag{1.13}$$

Substituting this into the travel disturbance model gives

$$r_{travel} = r_{accel}\tilde{x}_3 + \frac{1}{1 + r_{accel}/m} r_{travel}^2 \tag{1.14}$$

The travel resistance can be described using the new state $\tilde{x}_3$ instead of $\dot{v}_h$. Finally, the state transformation and the travel disturbance model yield the prediction model with the new state coordination below:

$$\begin{aligned}\dot{x} &= A(t)x + B(t)u + Gv + Hw \\ y &= Cx + Jv\end{aligned} \tag{1.15}$$

where

$$x = [\Delta d, \Delta v, \rho_{accel} x_f]^T \in \mathbb{R}^3, \quad v = r_{travel}, \quad w = \dot{v}_p$$

and

$$A(t) = \begin{bmatrix} 0 & 1 & -T_{hw} \\ 0 & 0 & -1 \\ 0 & 0 & A_f(t) \end{bmatrix}, B(t) = \begin{bmatrix} 0 \\ 0 \\ \rho_{accel} B_f(t) \end{bmatrix}$$

$$G = \begin{bmatrix} \rho_{accel} T_{hw}/m \\ \rho_{accel}/m \\ 0 \end{bmatrix}, H = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, J = \begin{bmatrix} 0 \\ 0 \\ -\rho_{accel}/m \end{bmatrix}$$

$$\rho_{accel} = \frac{1}{1 + r_{accel}/m}$$

Chapter 2

Suitability of MPC for the Control Problem

Model Predictive Control (MPC) is particularly suitable for solving the control problem due to several key features of the system:

2.1 Time-Varying Dynamics

The system dynamics are time-varying, which can be handled effectively by MPC because it uses a model of the system to predict future behavior.

2.2 Presence of Disturbances

The system is subject to disturbances, such as travel disturbance. MPC can account for these in its predictions and adjust control actions accordingly.

2.3 Constraints

The system has constraints on both the control inputs and the system states. MPC is well-suited to handle such constraints.

2.4 Need for Optimization

The system requires an optimization of a cost function, which could be related to energy consumption, system performance, etc. MPC operates by solving an optimization problem at each time step.

2.5 Requirement for Adaptability

The system parameters or dynamics can change over time. MPC can adapt to these changes by updating the prediction model at each sampling time.

Result

These features make MPC a powerful and flexible control strategy that can handle complex control problems in a systematic and efficient manner.

Chapter 3

Controller Design Formulation

3.1 Cost Matrices

In our Model Predictive Control (MPC) setup, we define the cost matrices $\mathbf{Q_t} = I$, $\mathbf{R_{t_{\Delta u}}} = I$, and $\mathbf{R_{t_u}} = I$, where I represents the identity matrix.

These matrices respectively penalize the tracking error, the rate of change, and the magnitude of the control input. We then iteratively adjust these matrices to optimize the system performance. Higher weights on the tracking error ($w_{y1}(t)$) or on the control input's change rate ($w_{\Delta u}(t)$) result in a more aggressive controller or smoother control inputs, respectively.

3.2 Constraints

In the MPC formulation, we apply constraints based on the physical limitations of the system. These constraints, which involve the control input u, its increment Δu, and the output y, are all subject to upper and lower bounds, as represented by the ”min” and ”max” suffixes. The specific values for these limits are determined by the actuation system's capabilities and the desired operational range of the system. We set the following requirements as constraints:

1. The relative velocity must not be excessively large, unless a preceding vehicle is initially detected outside of the range.
2. The acceleration of the host vehicle must be greater than or equal to -0.25 [G].
3. The acceleration change must be gradual, with the exception of traffic jam scenarios.

These requirements are carefully integrated into the MPC formulation to ensure the system operates within safe and efficient parameters.

3.3 Disturbances

The system is subject to disturbances, such as travel disturbance. These disturbances can cause prediction errors and degrade control performance. In the MPC formulation, the effect of these disturbances is explicitly considered in the prediction model. This allows the controller to compensate for the disturbances and maintain good performance. The specific effect of the disturbances on the feedback loop depends on their magnitude and frequency, and can be studied through simulation or experimental testing.

3.4 Dual Mode Prediction

The control input $u(t)$ guides the selection of the appropriate model, thereby creating a dual model system. Here are the matrices $A(t)$ and $B(t)$ for the two different cases:

$$A_{eng}(t) = \begin{bmatrix} 0 & 1 & -T_{hw} \\ 0 & 0 & -1 \\ 0 & 0 & -1/T_{eng} \end{bmatrix}, \quad B_{eng}(t) = \begin{bmatrix} 0 \\ 0 \\ K_{eng}(t)/T_{eng} \end{bmatrix},$$

$$A_{brk}(t) = \begin{bmatrix} 0 & 1 & -T_{hw} \\ 0 & 0 & -1 \\ 0 & 0 & -1/T_{brk} \end{bmatrix}, \quad B_{brk}(t) = \begin{bmatrix} 0 \\ 0 \\ K_{brk}(t)/T_{brk} \end{bmatrix},$$

If $u(t) \geq a_{thr_{off}}$, we use the $A_{eng}(t)$ and $B_{eng}(t)$ matrices, which reflect the engine dynamics. Conversely, if $u(t) < a_{thr_{off}}$, we switch to the $A_{brk}(t)$ and $B_{brk}(t)$ matrices, which encapsulate the brake dynamics. This dual model system allows the controller to adapt to the changing conditions of the vehicle's operation.

```
% Your system
A = [0 1 -1.3; 0 0 -1; 0 0 -2.17];
B = [0; 0; -1.5];
Q = eye(3);
R = 1;

% LQR
K = lqr(A, B, Q, R);

% Check stability by checking eigenvalues of (A-BK)
eig_A_BK = eig(A - B*K);
if all(real(eig_A_BK) < 0)
    disp('The system is asymptotically stable.')
else
    disp('The system is not asymptotically stable.')
end

>>> The system is asymptotically stable.
```

Note: I cannot add chain reaction into matlab.

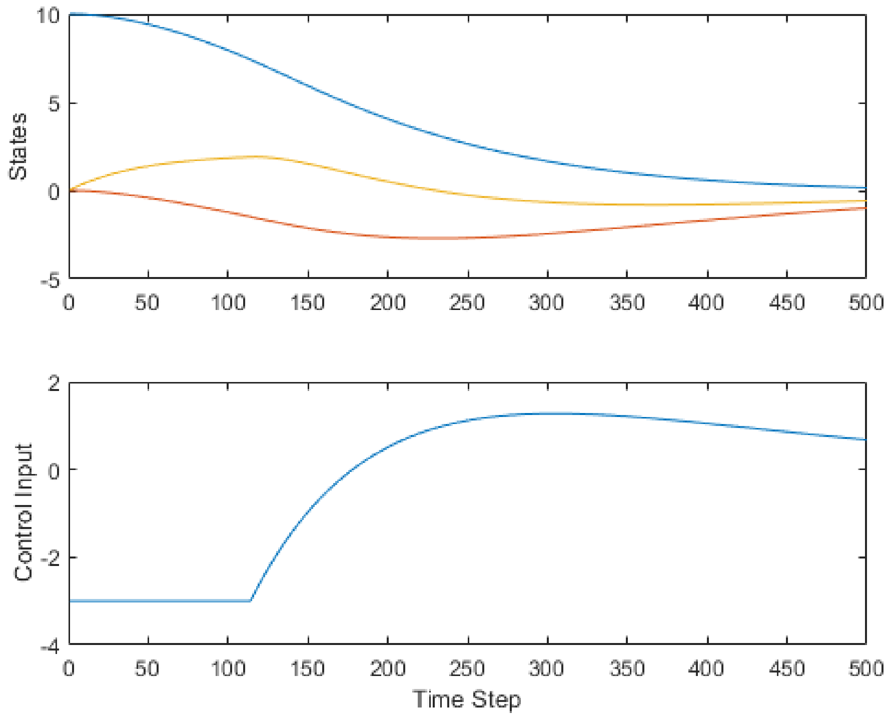

Figure 3.1: Behaviour when distance is relatively big

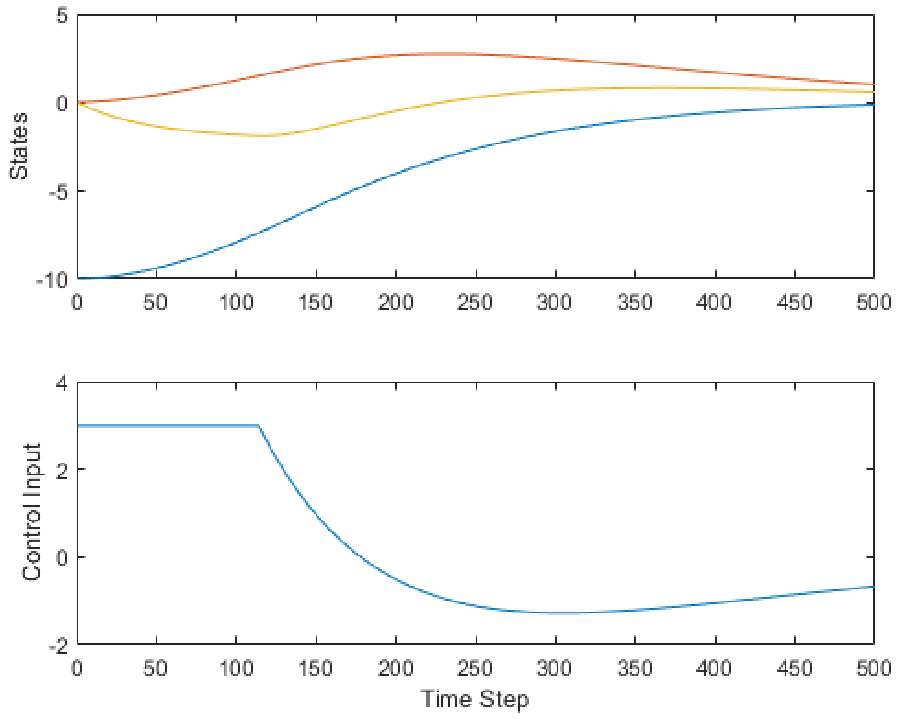

Figure 3.2: Behaviour when distance is relatively negative

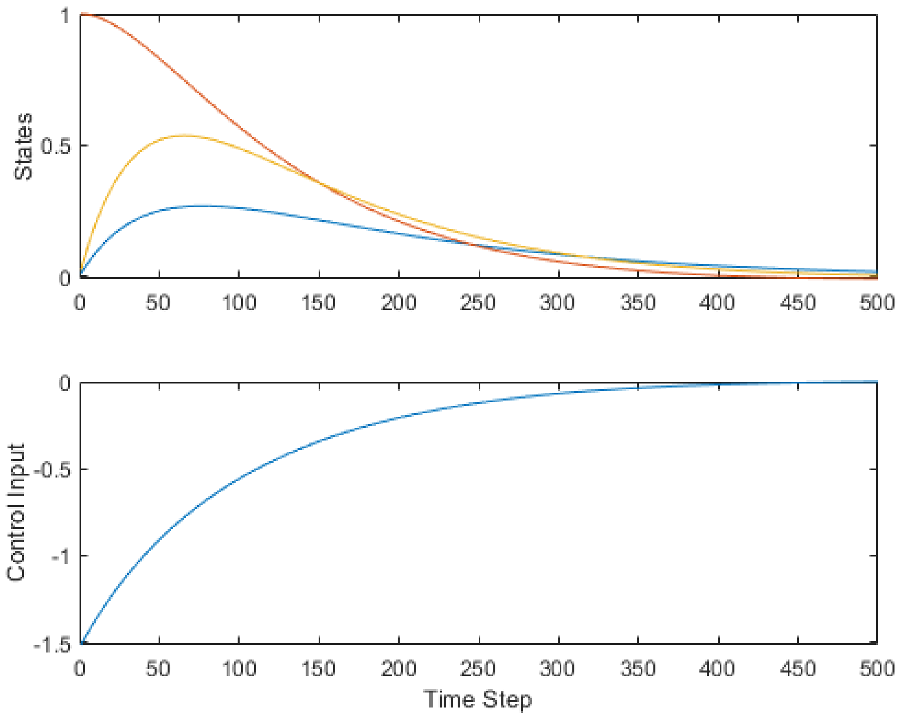

Figure 3.3: behavior when relative speed is high

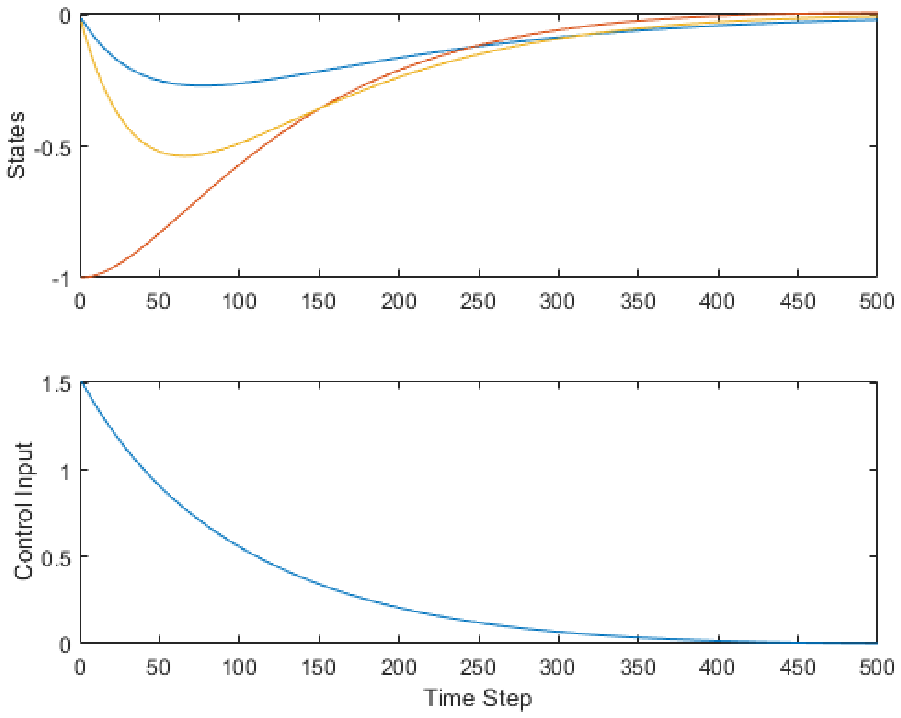

Figure 3.4: behavior when relative speed is slow

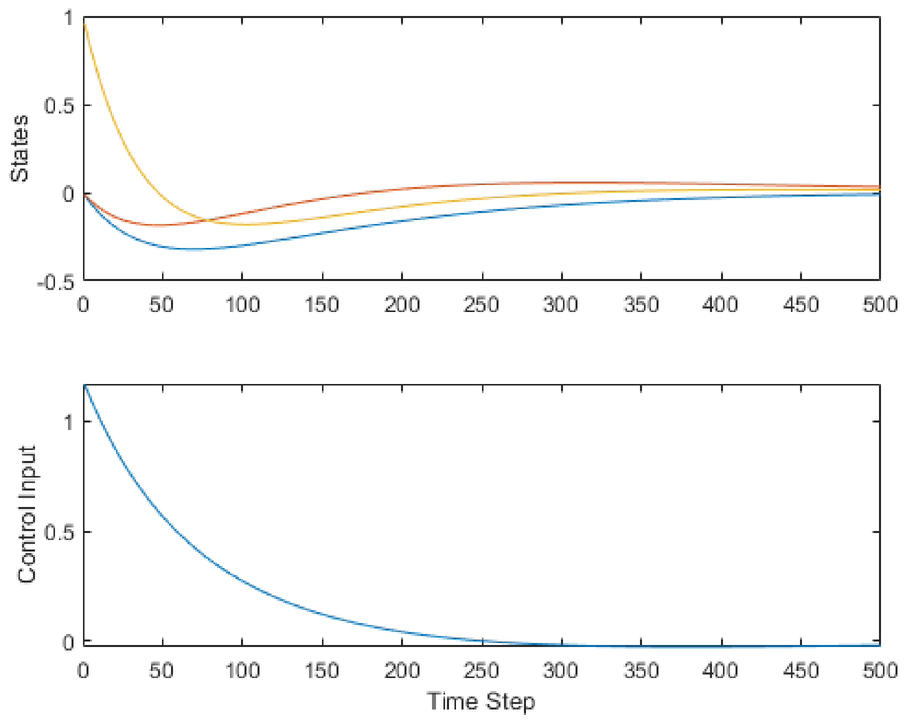

Figure 3.5: behaviour when relative acceleration is high

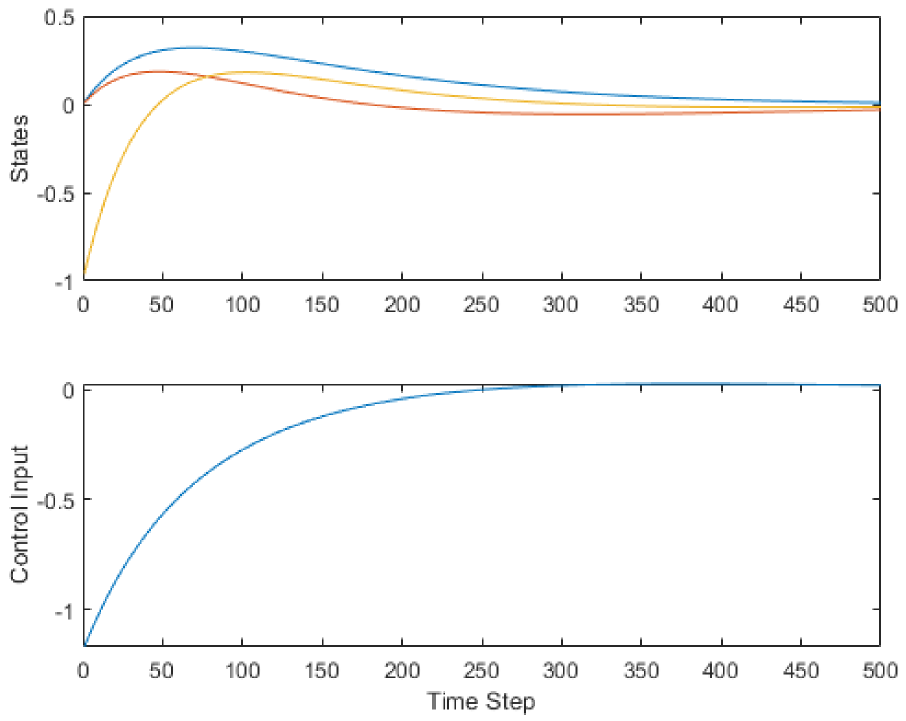

Figure 3.6: behaviour when relative acceleration is negative

Chapter 4

Part 2

4.1 a. Simulink model for Apollo satellite

All simulations are for 40 seconds starting from $t = 0$.

As can be observed from Figures 4.1 and 4.2, the step size decreases as the velocity of the satellite increases (in fact it is about 4. order derivative term). This enables providing more accurate data with higher resolution in areas where the satellite moves at high speeds.

The obtained trajectories with a step size of 0.001 are consistent and realistic. However, when we increase the step size to 0.01, the resulting trajectories are inaccurate. In the simulation performed with the Runge-Kutta method and a fixed step size, a successful result is also obtained.

However, similar to the variable step size case, when the step size increases, we obtain an erroneous simulation with Runge-Kutta as well. Additionally, while the fixed step size simulation was successful, the simulation with fixed step size took approximately 10 times longer. According to internet if system is complicated this speed difference can goes up to 100,000 times.

Variable step size simulations maintain the accuracy of calculations and improve performance, especially for systems with highly variable velocity and acceleration values.

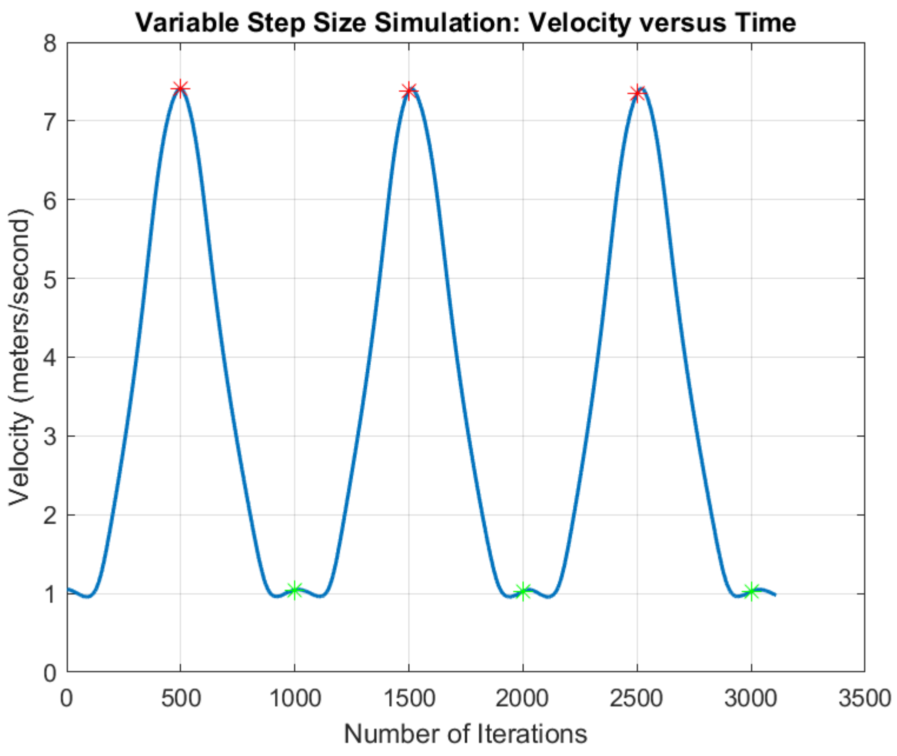

Figure 4.1: Satellite speed

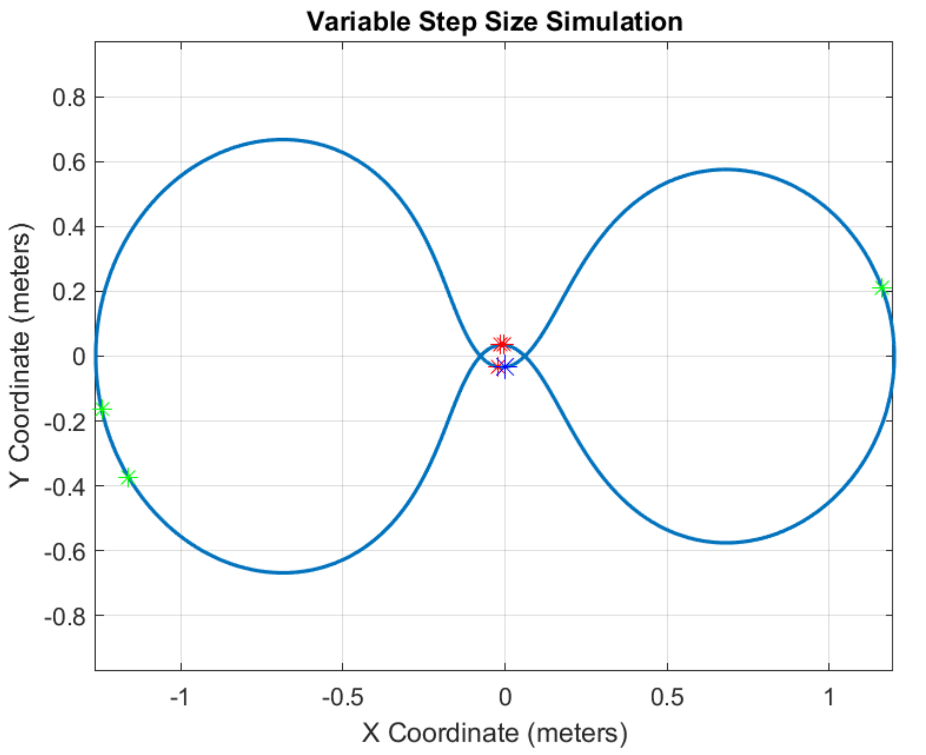

Figure 4.2: Orbit

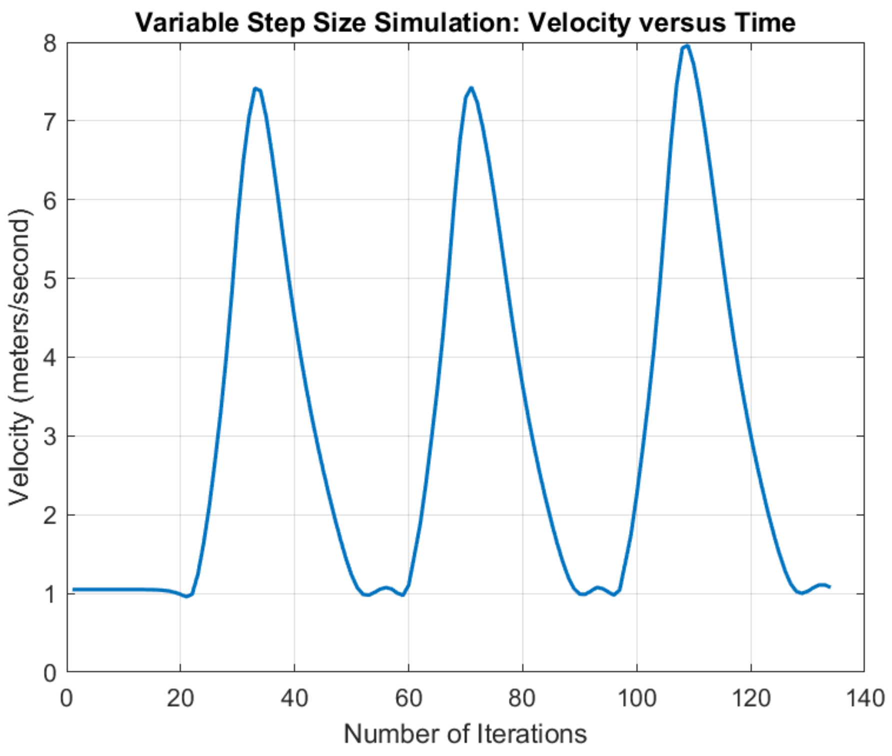

Figure 4.3: h = 0.01, variable step size simulation

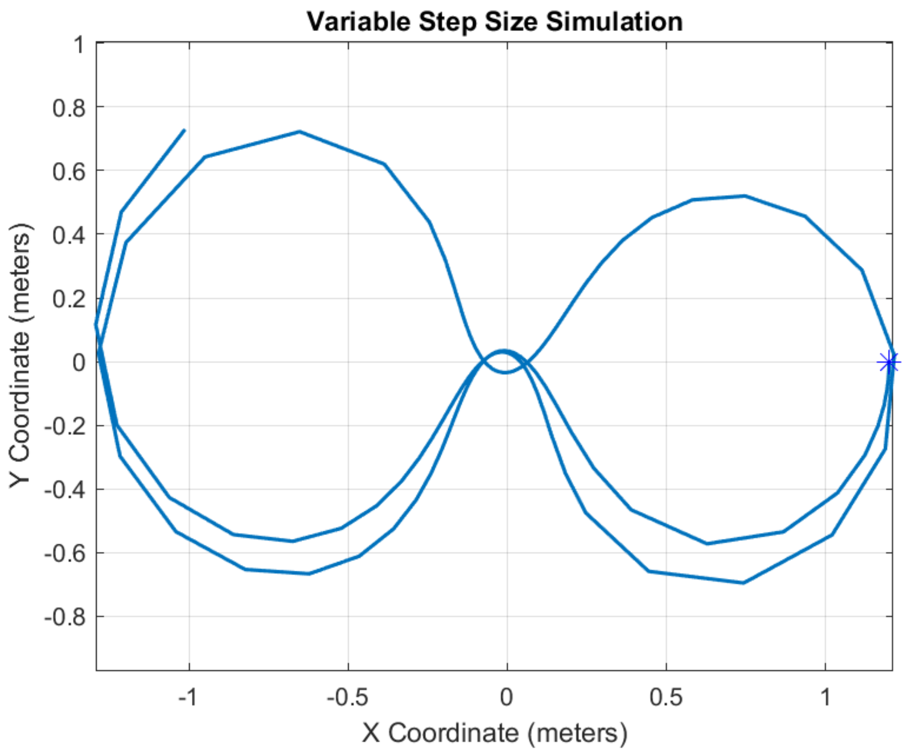

Figure 4.4: h = 0.01, variable step size simulation

4.2 Implementation and Testing of the Runge-Kutta Method

The Runge-Kutta method is a numerical technique used to solve ordinary differential equations (ODEs). The fourth-order Runge-Kutta method, often simply referred to as the ”RK4 method,” is commonly used due to its balance between computational efficiency and accuracy.

4.2.1 Method Implementation

The RK4 method approximates the solution to an ODE by taking a weighted average of four increments, where each increment is the estimated change over a small interval of time (or a ’step’). These increments are calculated at the beginning of the interval, at the midpoint, and at the end. The general formula for the RK4 method is as follows:

$$
\begin{aligned}
k_1 &= h \cdot f(t_n, y_n) \\
k_2 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= h \cdot f(t_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{1}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Where h is the step size, t_n is the current time, y_n is the current value, and $f(t, y)$ is the derivative of y.

4.2.2 Choosing a Suitable Step Size

The choice of step size h is crucial for the performance of the RK4 method. A smaller step size will generally result in a more accurate solution, but at the cost of requiring more computational resources. Conversely, a larger step size will be more efficient but less accurate.

A suitable step size can often be determined empirically. 0.001 in this case.

4.2.3 Testing the Method

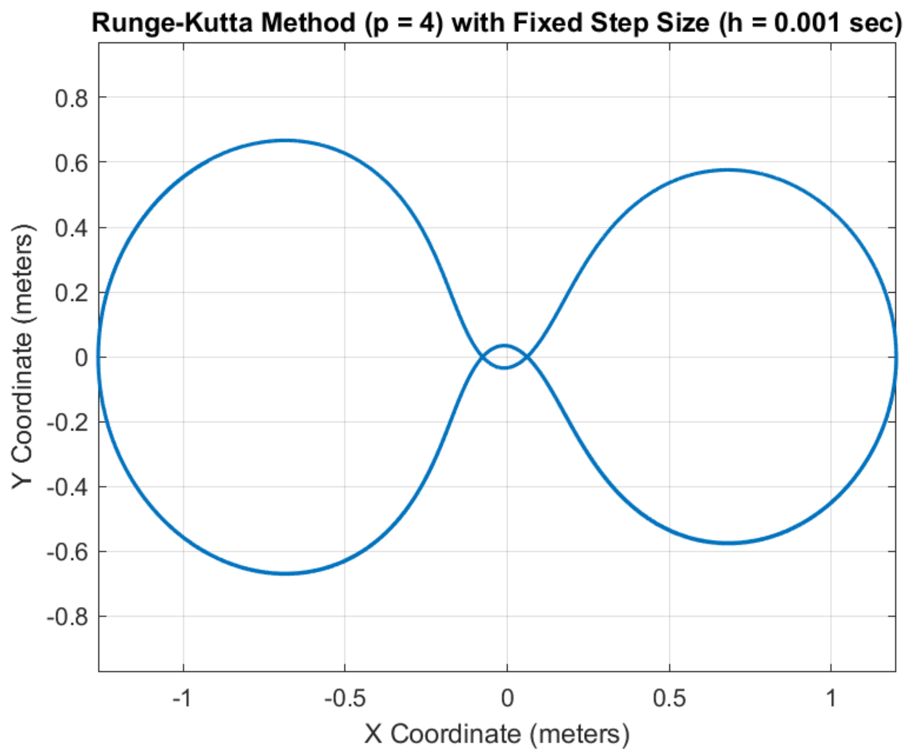

Figure 4.5: Orbit of Runge-Kutta fixed size p = 4, h = 0.001

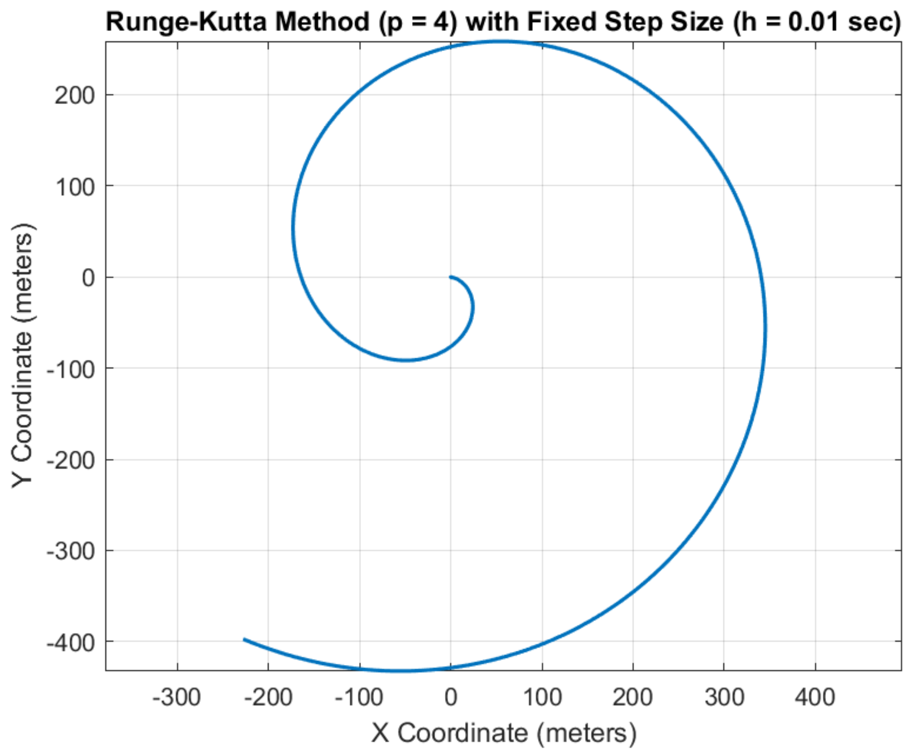

Figure 4.6: Orbit of Runge-Kutta fixed size p = 4, h = 0.01

4.3 Realization of Variable Step Size in Numerical Solvers

In numerical methods such as the Runge-Kutta for solving ordinary differential equations (ODEs), it's often advantageous to have a variable step size. This approach adjusts the step size dynamically based on the local behavior of the solution, allowing the method to take larger steps when the solution is changing slowly and smaller steps when it's changing rapidly.

4.4 Main Implementation Steps for a Variable Step Size Solver

1. **Initial Step Size Selection:** Start with a suitable initial step size. This could be a default small value, or it could be estimated based on the problem characteristics or initial error tolerance.
2. **Step Size Estimation:** After each step, estimate the local truncation error of the numerical method. This could be done, for instance, by taking a full step and two half steps and comparing the results.
3. **Step Size Adjustment:** Adjust the next step size based on the estimated error. If the error is below the tolerance, increase the step size; if it's above the tolerance, decrease the step size.
4. **Solution Update:** If the error is below the tolerance, accept the new solution and move to the next step. If it's above the tolerance, reject the step and repeat it with the smaller step size.
5. **Repeat:** Repeat steps 2-4 until the solution has been found for all desired values of the independent variable.
6. **Post Processing:** After the solution has been found, the step sizes used can be post processed for analysis. This can provide insight into the problem areas where smaller step sizes were needed, or regions where larger step sizes could be used.

RK4 System with adaptive step size

```
while t < t_end
    % RK4 step
    k1 = f(t, y);
    k2 = f(t + h/2, y + h/2 * k1);
    k3 = f(t + h/2, y + h/2 * k2);
    k4 = f(t + h, y + h * k3);
    y_new = y + h/6 * (k1 + 2*k2 + 2*k3 + k4);
    % Heun's method for error estimation
    y_star = y + h/2 * (k1 + k4);

    % Estimate error
    err = abs(y_new - y_star);

    % Update step size
    if err < absTol + relTol * abs(y_new)
        % Accept step
        t = t + h;
        y = y_new;
        T = [T, t];
        Y = [Y, y];
        H = [H, h];
        h = h * 1.2;
    else
        % Reject step
        h = h * 0.8;
    end
end
```

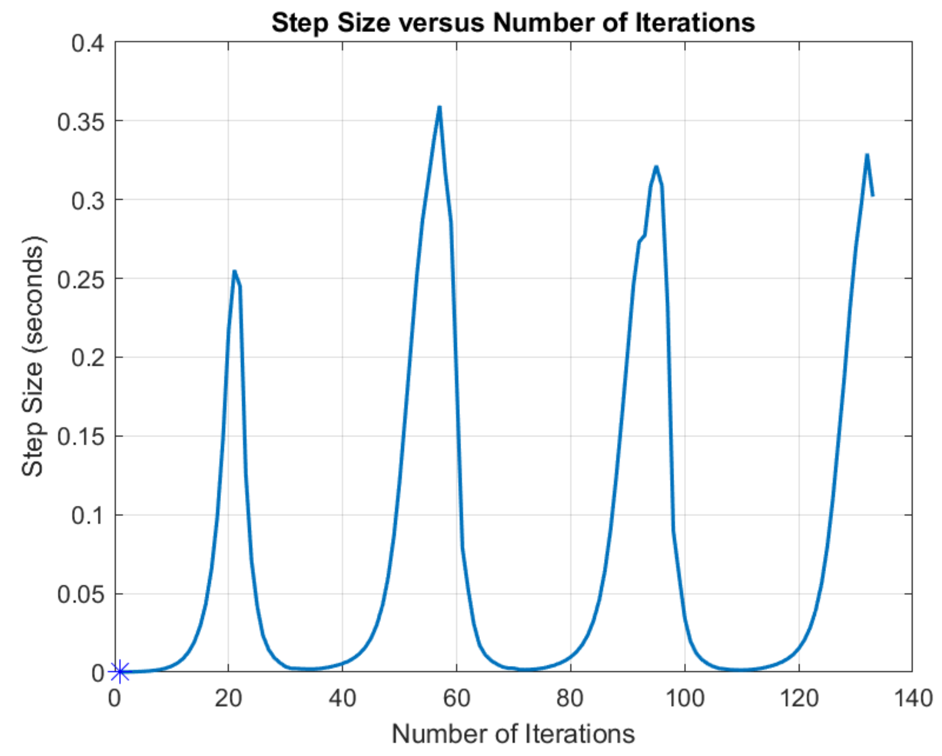

Figure 4.7: h = 0.01, variable step size simulation

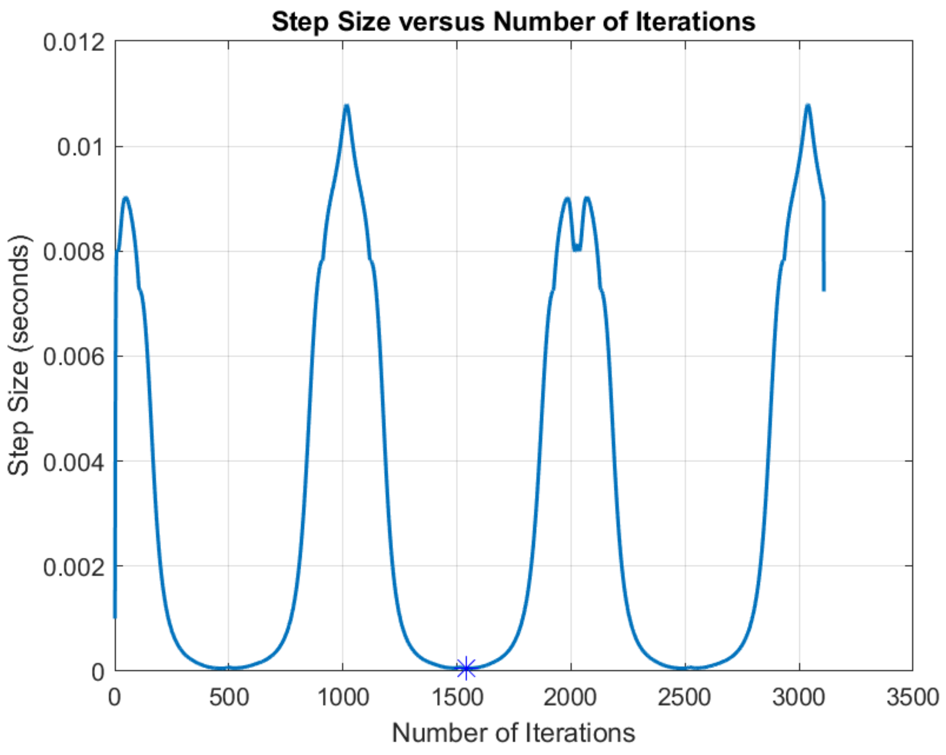

Figure 4.8: h = 0.001, variable step size simulation

The changes in the step sizes over the course of the simulation are a reflection of the complexity and non-linearity of the system's dynamics at different points in time.

When we look at the trajectory of the satellite and the associated step sizes, we observe that the step size decreases (i.e., the computation becomes more detailed) during periods when the satellite's trajectory is changing rapidly or in a complex manner. This could include moments when the satellite is making a sharp turn, accelerating rapidly, or passing close to a planet or other massive object that affects its trajectory.

Conversely, when the satellite's trajectory is relatively simple and predictable (for example, when it's moving in a straight line or following a steady curve), the solver might be able to take larger steps, because the error of approximation remains within acceptable bounds even over these longer intervals.