Modeling and Simulation of Multibody Dynamics
MBS Library Report

Sai Tej Paruchuri (saitejp@vt.edu)

Contents

1 Introduction **4**

2 Methodology **4**
- 2.1 Explantion of Methodologies . . . 4
  - 2.1.1 Methodology 1 . . . 4
  - 2.1.2 Methodology 2 . . . 5
  - 2.1.3 Library Structure for Methodology 2 . . . 6
- 2.2 Difference between 3D and 2D libraries . . . 6
- 2.3 Matlab Code Execution Explnation . . . 7
- 2.4 Benefits of Function handle f_DAE . . . 7
- 2.5 2D Double Pendulum Solution Steps . . . 7
  - 2.5.1 Separation of bodies . . . 7
  - 2.5.2 Rotation matrix definition . . . 8
  - 2.5.3 Defining $\boldsymbol{s}'$ vectors . . . 8
  - 2.5.4 Defining $\boldsymbol{r}_A$ and $\boldsymbol{r}_B$ vectors . . . 9
  - 2.5.5 Defining the constraint equations . . . 9
- 2.6 2D Library DAE Equation . . . 9
- 2.7 Limitations of the 2D Library and DAE equation . . . 10

3 Results **12**
- 3.1 2D Library Results . . . 12
  - 3.1.1 Norm . . . 12
  - 3.1.2 Linear Displacements . . . 13
  - 3.1.3 Angular Displacements . . . 14
  - 3.1.4 Linear Velocities . . . 15
  - 3.1.5 Angular Velocities . . . 16
  - 3.1.6 Linear Acceleration . . . 17
  - 3.1.7 Angular Acceleration . . . 18
- 3.2 3D Library Results . . . 19
  - 3.2.1 Norm . . . 19
  - 3.2.2 Linear Displacements . . . 20
  - 3.2.3 Euler Parameters . . . 21
  - 3.2.4 Linear Velocities . . . 22
  - 3.2.5 Euler Parameter Derivatives . . . 23
- 3.3 Comparison of 2D and 3D library results . . . 24

4 Other Results **26**
- 4.1 Bipdeal Robot Simulation . . . 26
  - 4.1.1 Norm . . . 26
  - 4.1.2 Linear Displacements . . . 27
  - 4.1.3 Angular Displacements . . . 28
  - 4.1.4 Linear Velocities . . . 29
  - 4.1.5 Angular Velocities . . . 30
  - 4.1.6 Linear Acceleration . . . 31
  - 4.1.7 Angular Acceleration . . . 32

5 Conclusion **32**

1 Introduction

This report explains the methodology involved in the construction of a 2D as well as a 3D library for the Dynamical Analysis of a double pendulum. The primary motivation to develop the library is to ease the simulation of different systems with different parameters. For example, different codes have to be developed for the simulation of a double pendulum and an excavator. But, once the library is developed, both the systems can be simulated using the same library.

2 Methodology

2.1 Explantion of Methodologies

This section describes the methodology used to simulate the double pendulum with and without using a library.

2.1.1 Methodology 1

Methodology 1 is a direct method for computing the State Vector. No libraries are defined in this case.
When the Main function of Methodology 1 is run, the Matlab code executes as explained below:

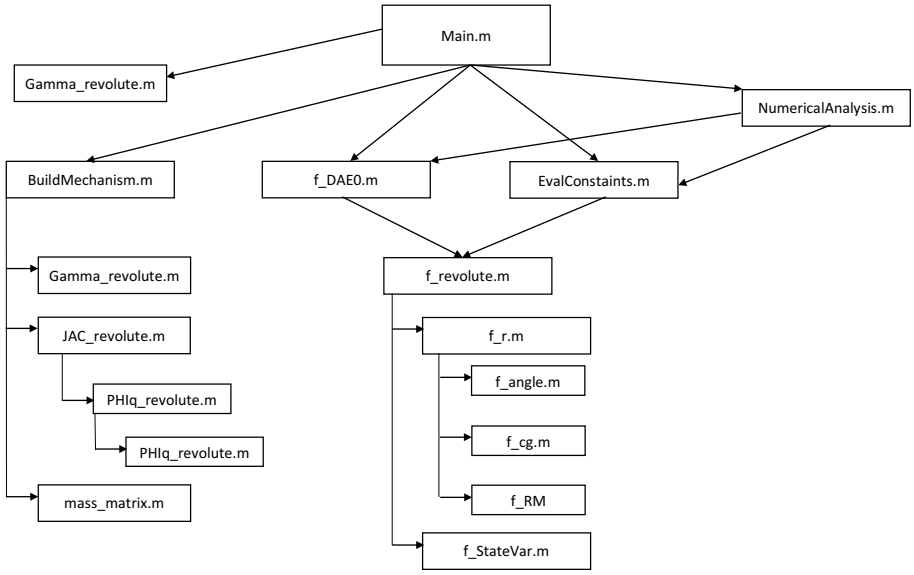
- The parameters such as number of state variables, number of constraints, sprime vectors, mass of bodies, moment of inertia of bodies, initial state vector and total time of simulation are stored in the workspace.
- The buildmechanism function is called for the initial state vector. The mass matrix, jacobian, PHIpq vector and Force vector are defined in the buildmechanism. These vectors are defined in the function based on hand based calculations.
- The function handle for f_DAE0 and Evalconstraints are defined. These functions return the $\dot{Y}$ - derivative of state vector and constraint vector respectively.
  - The DAE obtained from calculations on paper is defined inside the f_DAE0 function. It is important to call the BuildMechanism inside f_DAE0 in order to make sure the mass matrix, jacobian, PHIpq vector and Force vector are updated at each time step.
  - The constraint vector obtained from calculations on paper is defined in side the Evalconstraints function.
- NumericalAnalysis function is called. The function handles f_DAE, f_Evalconstraints are passed into the NumericalAnalysis function along with n,m,l,initial state vector, final time of simulation
  - Inside the NumericalAnalysis function, ode15s is used to solve for the state vector. The f_DAE function handle is given as the ode function to be solved by the ode15s solver since f_DAE0 function returns the derivative of the state vector.
  - The Evalconstraints functions is called and assigned to PHI vector at each time step. The norm of PHI at each time step is determined and returned to the main function.

2.1.2 Methodology 2

Methodology 2 uses libraries for computing the State Vector.
When the Main function of Methodology 2 is run, the Matlab code executes as explained below:

- The parameters such as number of state variables, number of constraints, sprime vectors, mass of bodies, moment of inertia of bodies, initial state vector and total time of simulation are stored in the workspace.
- The buildmechanism function is called for the initial state vector. The mass matrix, jacobian, PHIpq vector and Force vector are obtained by calling the following function.
  - The mass_matrix function returns the mass matrix for the DAE when the body number, mass and inertia are passed.
  - JAC_revolute function returns the Jacobian of revolute joint.
    - The PHIq_revolute function returns the parts of the jacobian corresponding to a particular body.
    - The f_statevar function returns the velocity, parameter and displacement vector when the state variable is passed.
  - The Gamma_revolute function returns the gamma vector used in the DAE.
    - The f_RM function returns the 'A' rotation matrix for a given angle.
  - The force vector Qa is defined directly.
- The function handle for f_DAE0 and Evalconstraints functions are defined. These functions return the $\dot{Y}$ - derivative of state vector and constraint vector respectively.
  - The DAE is defined inside the f_DAE0 function. It is important to call the BuildMechanism inside f_DAE0 in order to make sure the mass matrix, jacobian, PHIpq vector and Force vector are updated at each time step.
  - The constraint vector obtained by calling the f_revolute function twice inside the Evalconstraints function. f_revolute function returns the constraint vector for a given revolute joint.
- NumericalAnalysis function is called. The function handles f_DAE, f_Evalconstraints are passed into the NumericalAnalysis function along with n,m,l,initial state vector, final time of simulation
  - Inside the NumericalAnalysis function, ode15s is used to solve for the state vector. The f_DAE function handle is given as the ode function to be solved by the ode15s solver since f_DAE0 function returns the derivative of the state vector.
  - The Evalconstraints functions is called and assigned to PHI vector at each time step. The norm of PHI at each time step is determined and returned to the main function.

2.1.3 Library Structure for Methodology 2



2.2 Difference between 3D and 2D libraries

The 3d library differs from the 2D library in the following aspects:

- The 3D library used a structure similar to the 2D library but it was more complex (as expected) in the sense that it required more number of functions for its development.
- The 3D library took more computational time to simulate the double pendulum than the 2D library. This increase in computational time can be attributed to the increase in the number of equations and parameters.
- The 3D library used structures to pass the revolute joint parameters and body property parameters into the functions. More explanation on this has been given in the following section.
- The 3D DAE equation uses Euler parameters instead of angles in the state vector. Functions that relate to euler parameters had to be created for the 3D library.

2.3 Matlab Code Execution Explnation

This section explain how the parameters and the state vectors are passed through the code.

- The first step of the execution of the matlab code for 2D library is definition of parameters, initial state vector and functions handles for fDAE0, BuildMechanism and EvalConstraints.
- The NumericalAnalysis function is called. The functions handles for fDAE0, Evalconstraints and BuildMechanism are passed as parameters to the function.
- The ode15s inside the NumericalAnalysis functions calls the fDAE0 function handle to compute the value of the derivative of the state vector at a given time.
- The BuildMechanism function handle is called inside fDAE0 at each time step to compute the values required for the DAE.
- The ode15s integrates the output of fDAE0 function to get the state vector at each time step.
- The state vector obtained at each time step is passed to the EvalConstraints function to get the constraint vector at a given time step. The norm of constraint vector is then calculated.
- The state vector for the whole time span and the norm at each time step is passed to the main function. This data is used for plotting as well as animation.

2.4 Benefits of Function handle f_DAE

f_DAE is the function handle that stores the association to the function f_DAE0. The function handle is defined in such a way that the time and the state vector can be passed along with the function handle. For example when the function handle f_DAE is called by the ode15s for a given time step and state vector, these parameters are passed to the main file to the place where the function handle is defined. The values passed by the ode15s in the NumericalAnalysis function are passed into the function. The remaining values are passed from the main file.

2.5 2D Double Pendulum Solution Steps

2.5.1 Separation of bodies

The two bodies are separated and local coordinate systems are assigned as shown in Fig. 1.

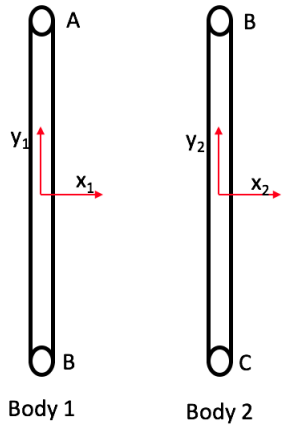


Figure 1: Individual bodies of double pendulum

The $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ matrices are defined for the two bodies.

$$\boldsymbol{q}_1 = [x_1, y_1, \varphi_1]^T \qquad \boldsymbol{q}_2 = [x_2, y_2, \varphi_2]^T$$

$$\boldsymbol{q} = [x_1, y_1, \varphi_1, x_2, y_2, \varphi_2]^T$$

2.5.2 Rotation matrix definition

The rotation matrices are defined for the two bodies. The rotation matrix can be expressed as

$$\boldsymbol{R} = \begin{bmatrix} cos\varphi & -sin\varphi \\ sin\varphi & cos\varphi \end{bmatrix}$$

2.5.3 Defining s' vectors

The position of points A, B and C with respect to the local coordinate systems are defined as follows,

$$\boldsymbol{s}'_{A1} = \begin{bmatrix} 0 \\ L \end{bmatrix} \qquad \boldsymbol{s}'_{B1} = \begin{bmatrix} 0 \\ -L \end{bmatrix}$$

$$\boldsymbol{s}'_{B2} = \begin{bmatrix} 0 \\ L \end{bmatrix} \qquad \boldsymbol{s}'_{C2} = \begin{bmatrix} 0 \\ -L \end{bmatrix}$$

The position of the local coordinate systems of the two bodies would be

$$\boldsymbol{r}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \qquad \boldsymbol{r}_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

2.5.4 Defining r_A and r_B vectors

The position of points A and B with respect to the global coordinate system are defined.

$$\boldsymbol{r}_O = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{r}_{A1} = \boldsymbol{r}_1 + R_1 s'_{A1} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} cos\varphi_1 & -sin\varphi_1 \\ sin\varphi_1 & cos\varphi_1 \end{bmatrix} \begin{bmatrix} 0 \\ L \end{bmatrix} = \begin{bmatrix} x_1 - Lsin\varphi_1 \\ y_1 + Lcos\varphi_1 \end{bmatrix}$$

$$\boldsymbol{r}_{B1} = \boldsymbol{r}_1 + R_1 s'_{B1} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} cos\varphi_1 & -sin\varphi_1 \\ sin\varphi_1 & cos\varphi_1 \end{bmatrix} \begin{bmatrix} 0 \\ -L \end{bmatrix} = \begin{bmatrix} x_1 + Lsin\varphi_1 \\ y_1 - Lcos\varphi_1 \end{bmatrix}$$

$$\boldsymbol{r}_{B2} = \boldsymbol{r}_2 + R_2 s'_{B2} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} + \begin{bmatrix} cos\varphi_2 & -sin\varphi_2 \\ sin\varphi_2 & cos\varphi_2 \end{bmatrix} \begin{bmatrix} 0 \\ L \end{bmatrix} = \begin{bmatrix} x_2 - Lsin\varphi_2 \\ y_2 + Lcos\varphi_2 \end{bmatrix}$$

2.5.5 Defining the constraint equations

The given system has two revolute joints which can be expressed as

$$\boldsymbol{r}_{A1} - \boldsymbol{r}_O = \boldsymbol{0}$$
$$\boldsymbol{r}_{B2} - \boldsymbol{r}_{B1} = \boldsymbol{0}$$

On expansion we get the constraint vector as,

$$\boldsymbol{\Phi}(\boldsymbol{q}) = \begin{bmatrix} x_1 - Lsin\varphi_1 \\ y_1 + Lcos\varphi_1 \\ x_2 - Lsin\varphi_2 - x_1 - Lsin\varphi_1 \\ y_2 + Lcos\varphi_2 - y_1 + Lcos\varphi_1 \end{bmatrix} = \boldsymbol{0}$$

2.6 2D Library DAE Equation

The velocity and acceleration vectors are

$$\dot{\boldsymbol{q}} = [\dot{x}_1, \dot{y}_1, \dot{\varphi}_1, \dot{x}_2, \dot{y}_2, \dot{\varphi}_2]^T$$
$$\ddot{\boldsymbol{q}} = [\ddot{x}_1, \ddot{y}_1, \ddot{\varphi}_1, \ddot{x}_2, \ddot{y}_2, \ddot{\varphi}_2]^T$$

The jacobian matrix is given by

$$\boldsymbol{\Phi}_{\boldsymbol{q}} = \begin{bmatrix} 1 & 0 & -Lcos\varphi_1 & 0 & 0 & 0 \\ 0 & 1 & -Lsin\varphi_1 & 0 & 0 & 0 \\ -1 & 0 & -Lcos\varphi_1 & 1 & 0 & -Lcos\varphi_2 \\ 0 & -1 & -Lsin\varphi_1 & 0 & 1 & -Lsin\varphi_2 \end{bmatrix}$$

$$\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} = \begin{bmatrix} 1 & 0 & -Lcos\varphi_1 & 0 & 0 & 0 \\ 0 & 1 & -Lsin\varphi_1 & 0 & 0 & 0 \\ -1 & 0 & -Lcos\varphi_1 & 1 & 0 & -Lcos\varphi_2 \\ 0 & -1 & -Lsin\varphi_1 & 0 & 1 & -Lsin\varphi_2 \end{bmatrix} \begin{Bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\varphi}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{\varphi}_2 \end{Bmatrix} = \begin{bmatrix} \dot{x_1} - \dot{\varphi}_1 Lcos\varphi_1 \\ \dot{y_1} - \dot{\varphi}_1 Lsin\varphi_1 \\ -\dot{x_1} - \dot{\varphi}_1 Lcos\varphi_1 + \dot{x_2} - \dot{\varphi}_2 Lcos\varphi_2 \\ -\dot{y_1} - \dot{\varphi}_1 Lsin\varphi_1 + \dot{y_2} - \dot{\varphi}_2 Lsin\varphi_2 \end{bmatrix}$$

$$\gamma \equiv \boldsymbol{\Phi_q}\ddot{\boldsymbol{q}} = -(\boldsymbol{\Phi_q}\dot{\boldsymbol{q}})_{\boldsymbol{q}} = \begin{bmatrix} 0 & 0 & -\dot{\varphi_1}Lsin(\varphi_1) & 0 & 0 & 0 \\ 0 & 0 & \dot{\varphi_1}Lcos(\varphi_1) & 0 & 0 & 0 \\ 0 & 0 & -\dot{\varphi_1}Lsin(\varphi_1) & 0 & 0 & -\dot{\varphi_2}Lsin(\varphi_2) \\ 0 & 0 & \dot{\varphi_1}Lcos(\varphi_1) & 0 & 0 & \dot{\varphi_2}Lcos(\varphi_2) \end{bmatrix}$$

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3, \lambda_4]^T$$

$$\boldsymbol{Q_A} = [0, -mg, 0, 0, -mg, 0]^T$$

The equation equation can be expressed as

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\Phi_q^T} & \boldsymbol{0} \\ \boldsymbol{\Phi_q} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q_A} \\ \boldsymbol{\gamma} \\ \dot{\boldsymbol{q}} \end{bmatrix}$$

The DAE equation is solved using Matlab ode15s.

2.7 Limitations of the 2D Library and DAE equation

Careful analysis of the norm graph given below in 2D library methodologies shows that the error in the displacement vector is accumulated over time. This implies that the DAE equation used and the solution of the ode15s obtained will have significant error when simulated for very long time span.

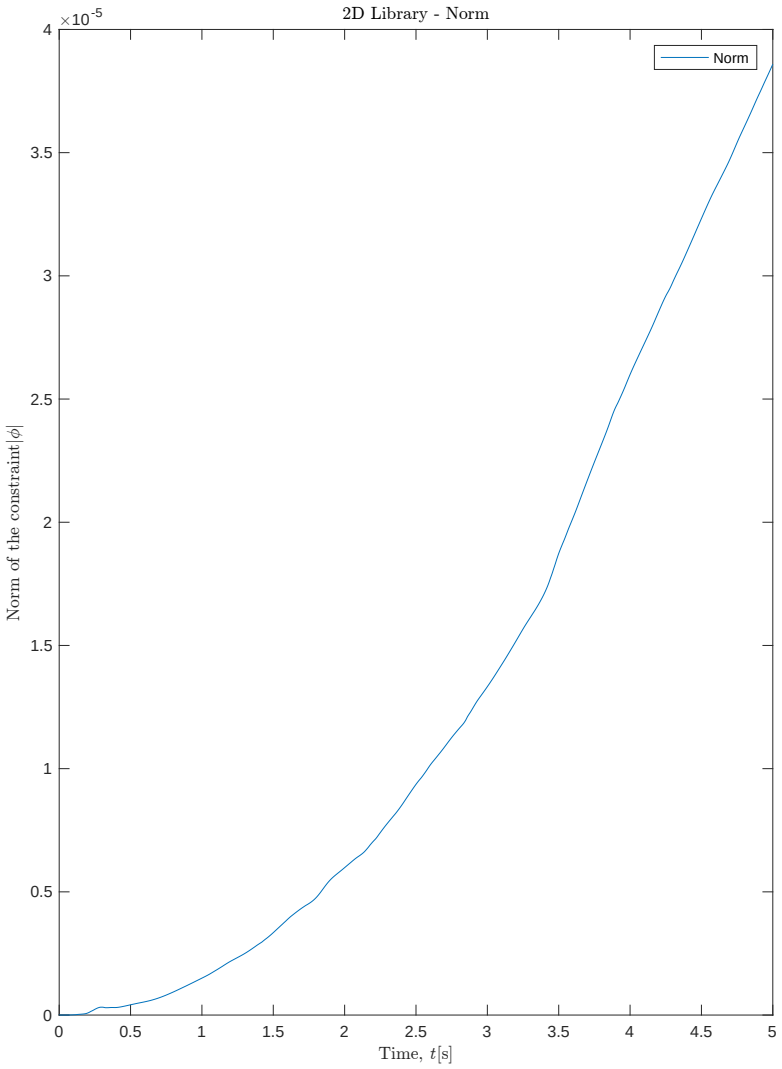


Figure 2: Time vs 2D Norm

3 Results

3.1 2D Library Results

3.1.1 Norm

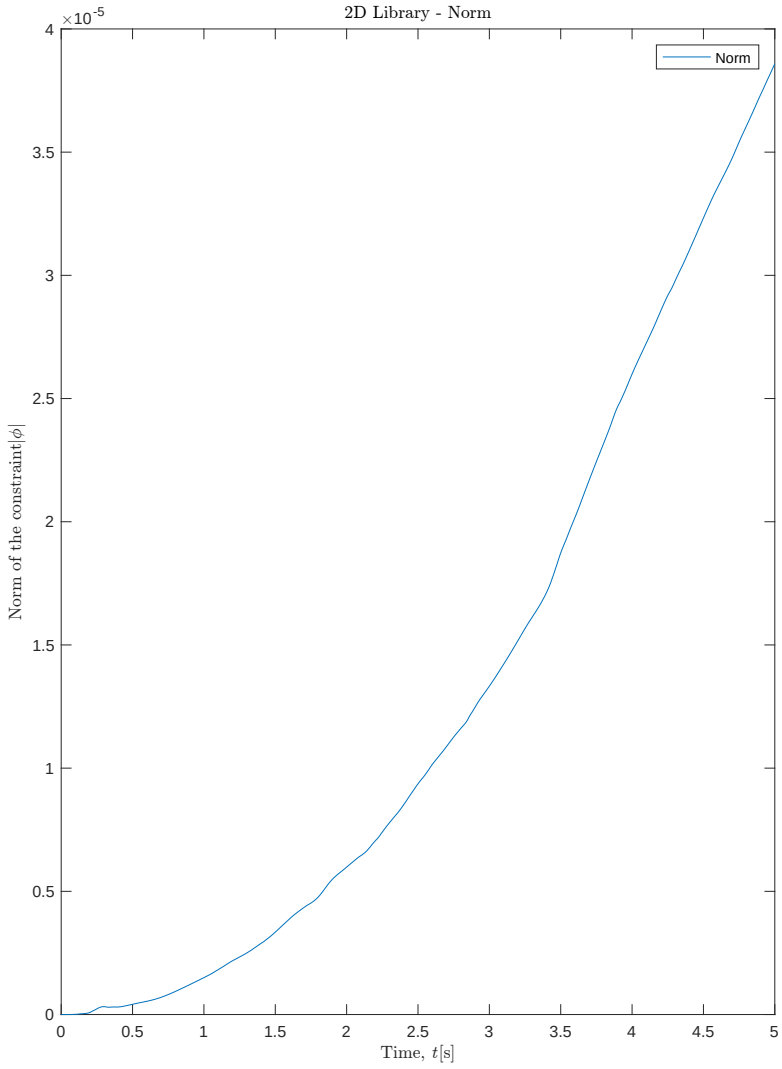


Figure 3: Time vs Norm

3.1.2 Linear Displacements

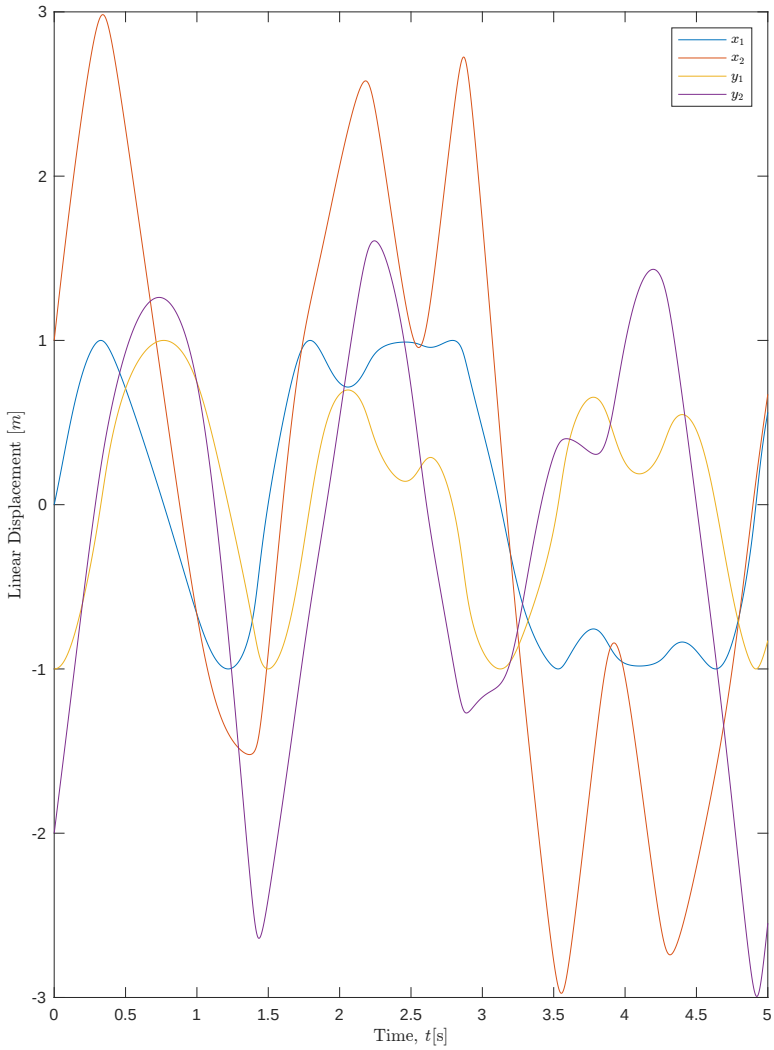


Figure 4: Time vs Linear Displacements

3.1.3 Angular Displacements

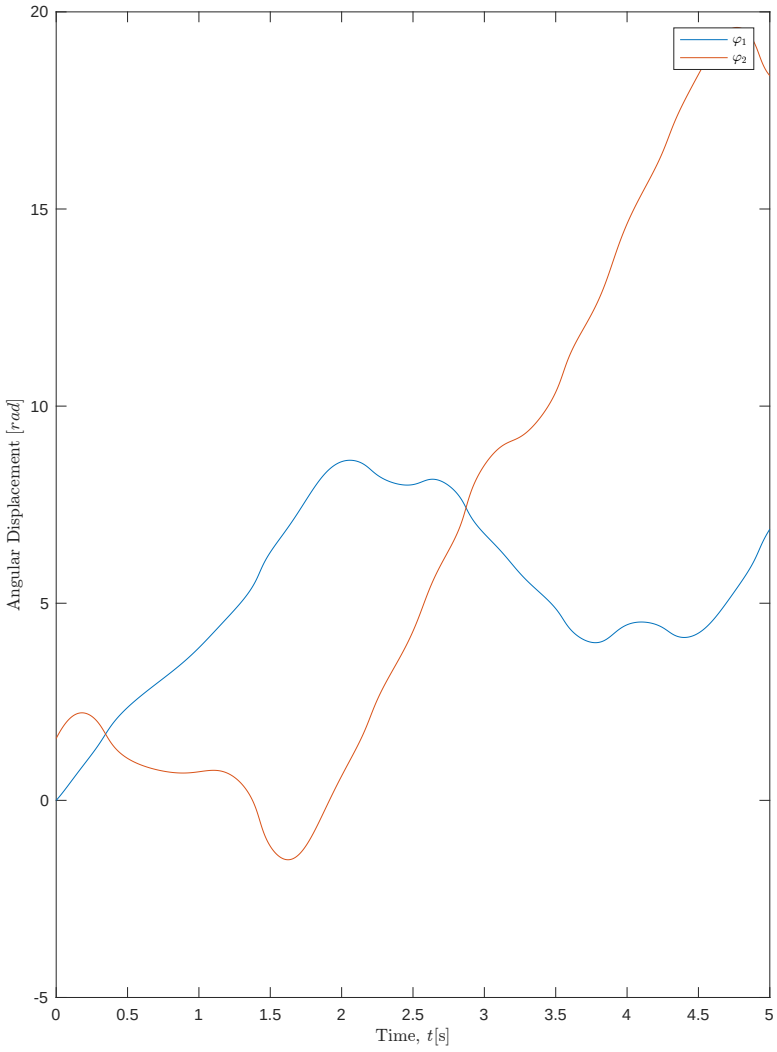


Figure 5: Time vs Angular Displacements

3.1.4 Linear Velocities

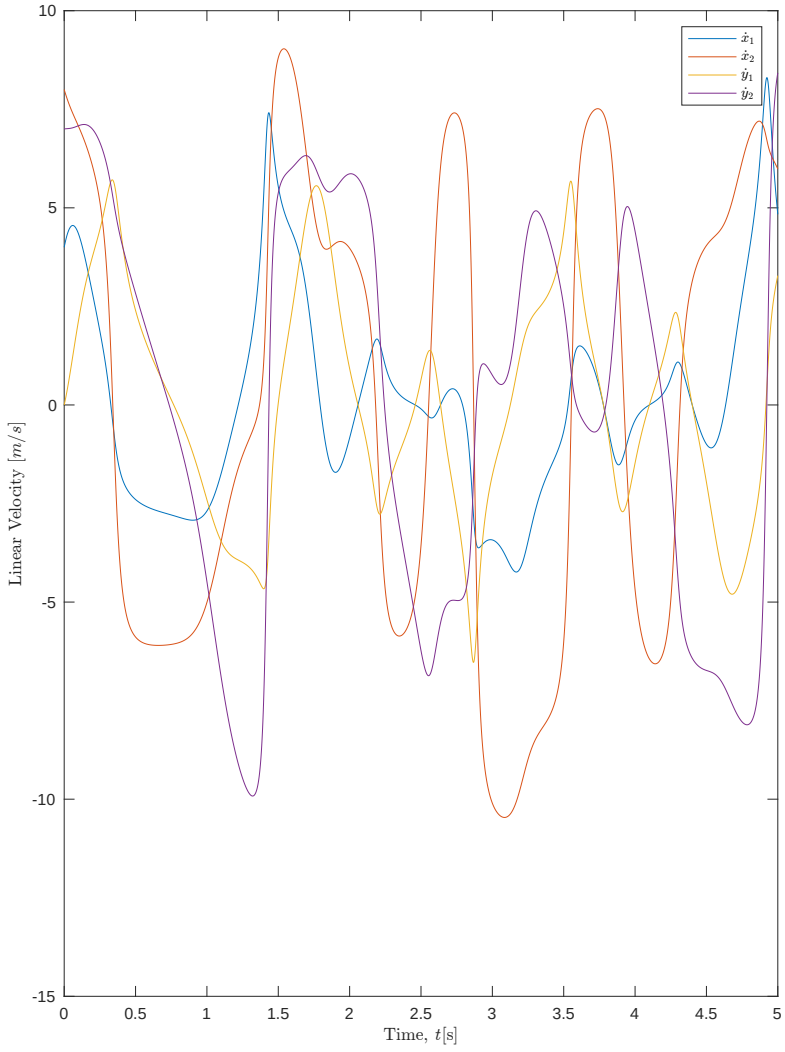


Figure 6: Time vs Linear Velocities

3.1.5 Angular Velocities

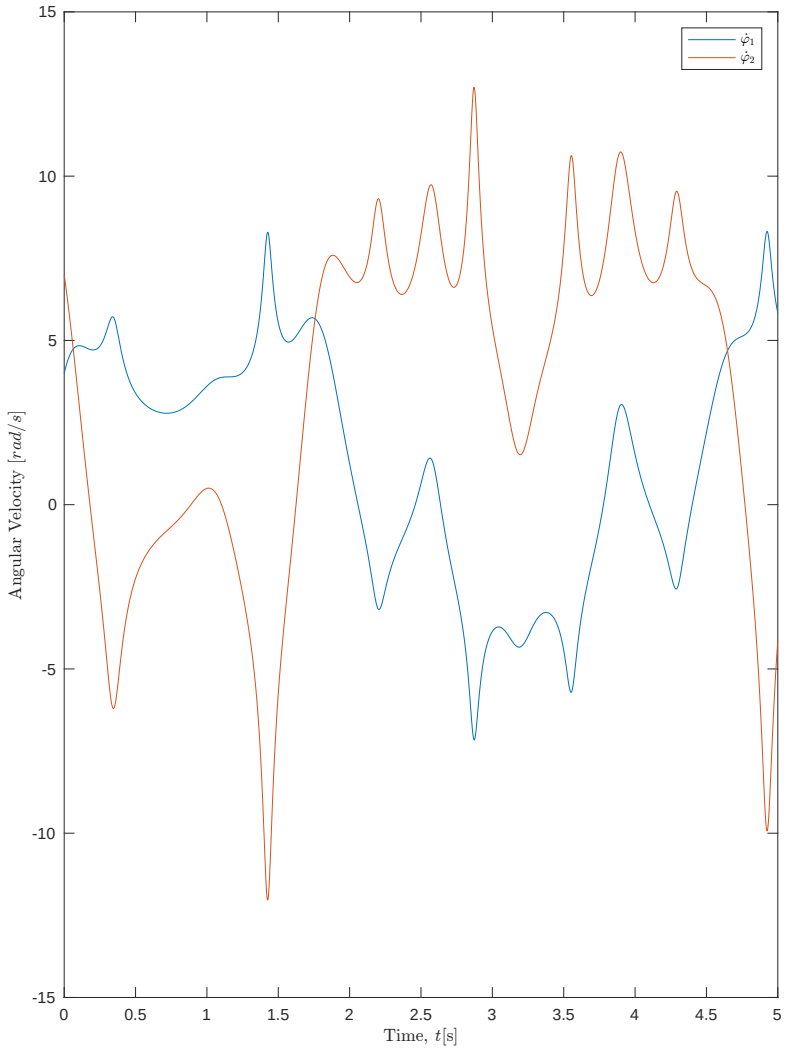


Figure 7: Time vs Angular Velocities

3.1.6 Linear Acceleration

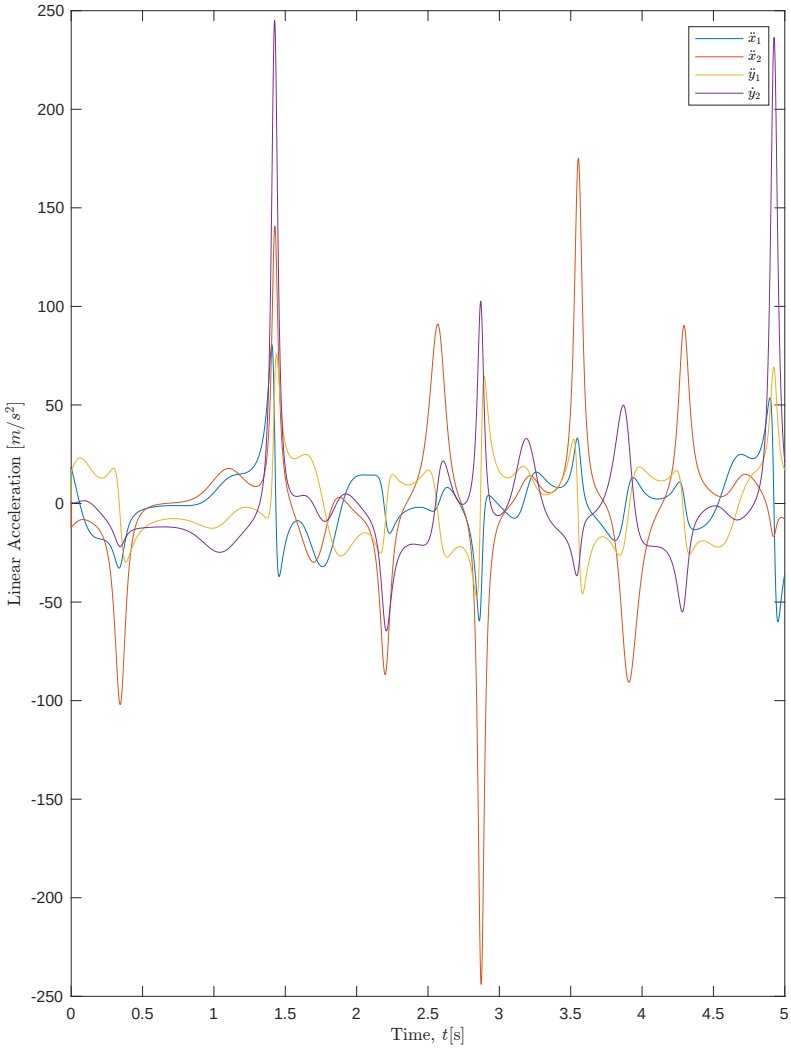


Figure 8: Time vs Linear Acceleration

3.1.7 Angular Acceleration

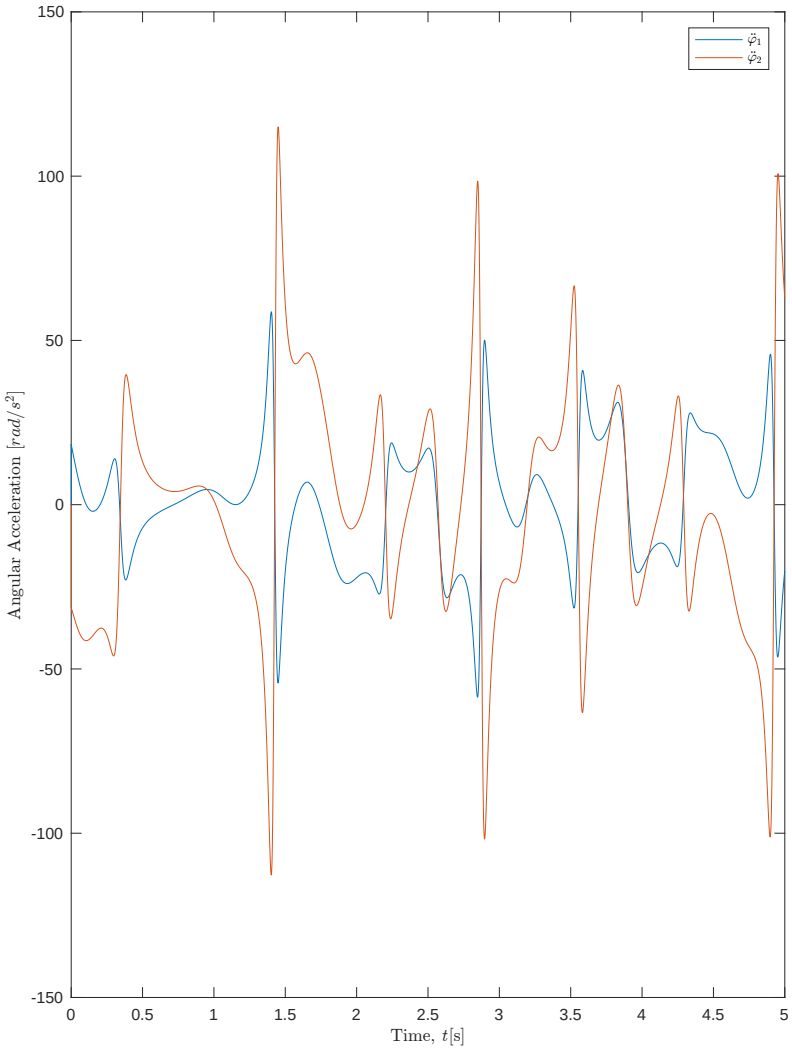


Figure 9: Time vs Angular Acceleration

3.2 3D Library Results

3.2.1 Norm

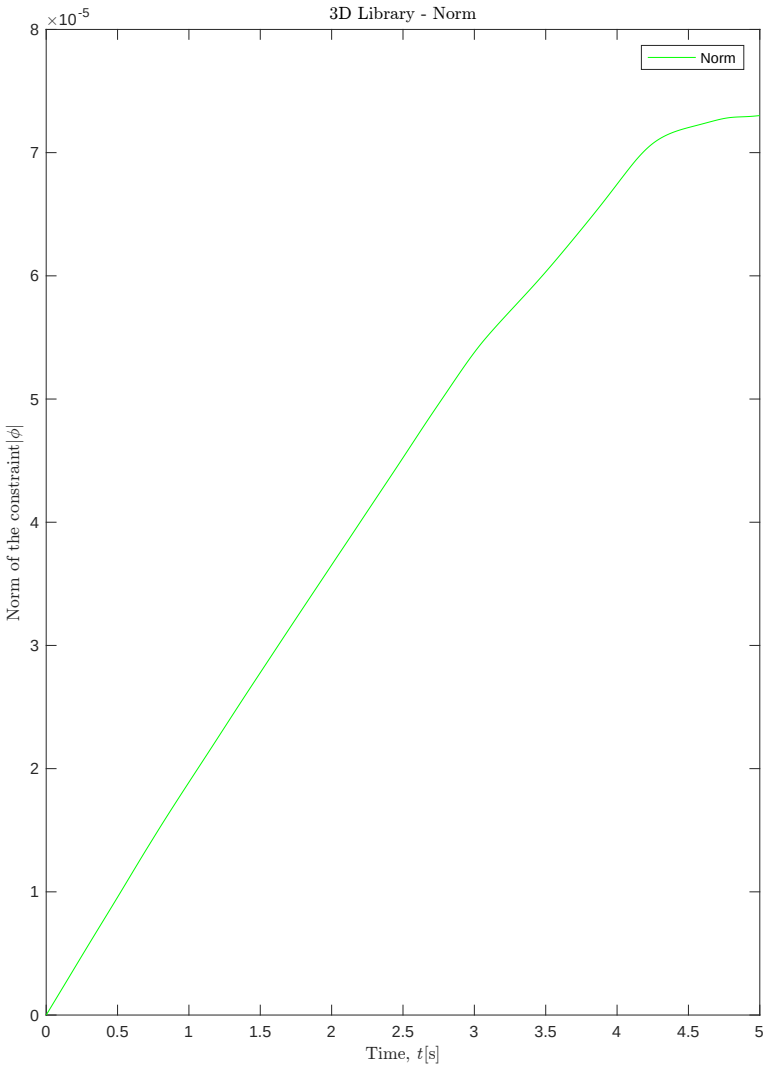


Figure 10: Time vs Norm

3.2.2 Linear Displacements

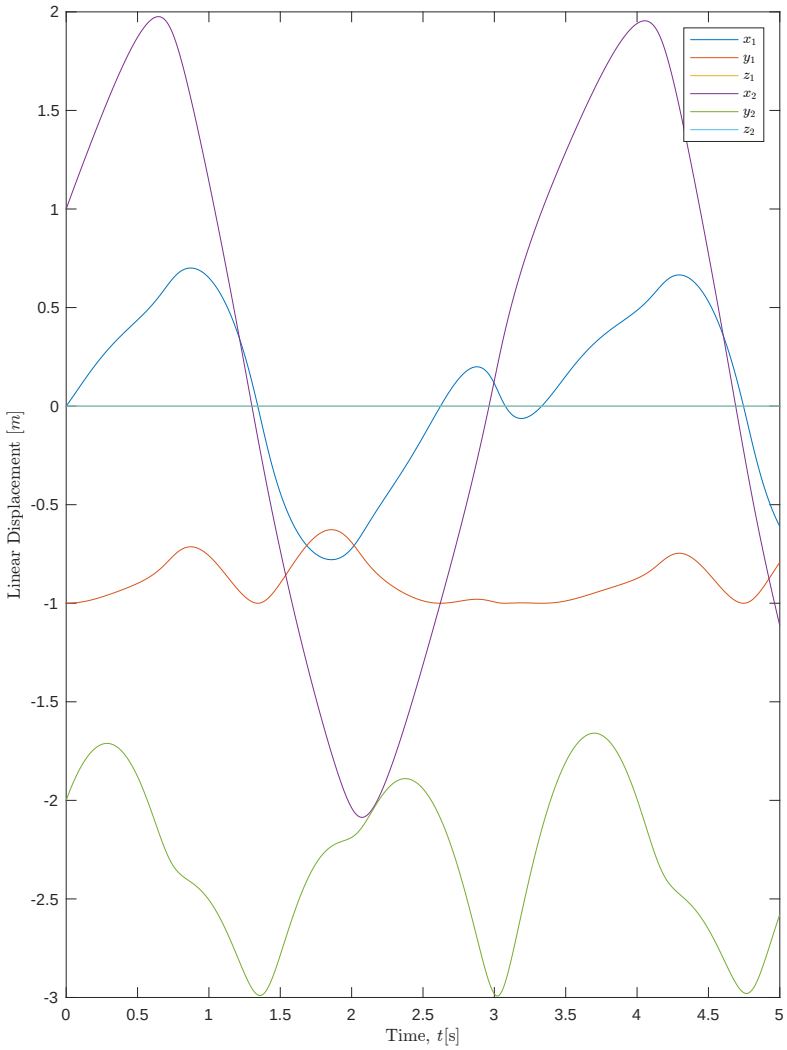


Figure 11: Time vs Linear Displacements

3.2.3 Euler Parameters

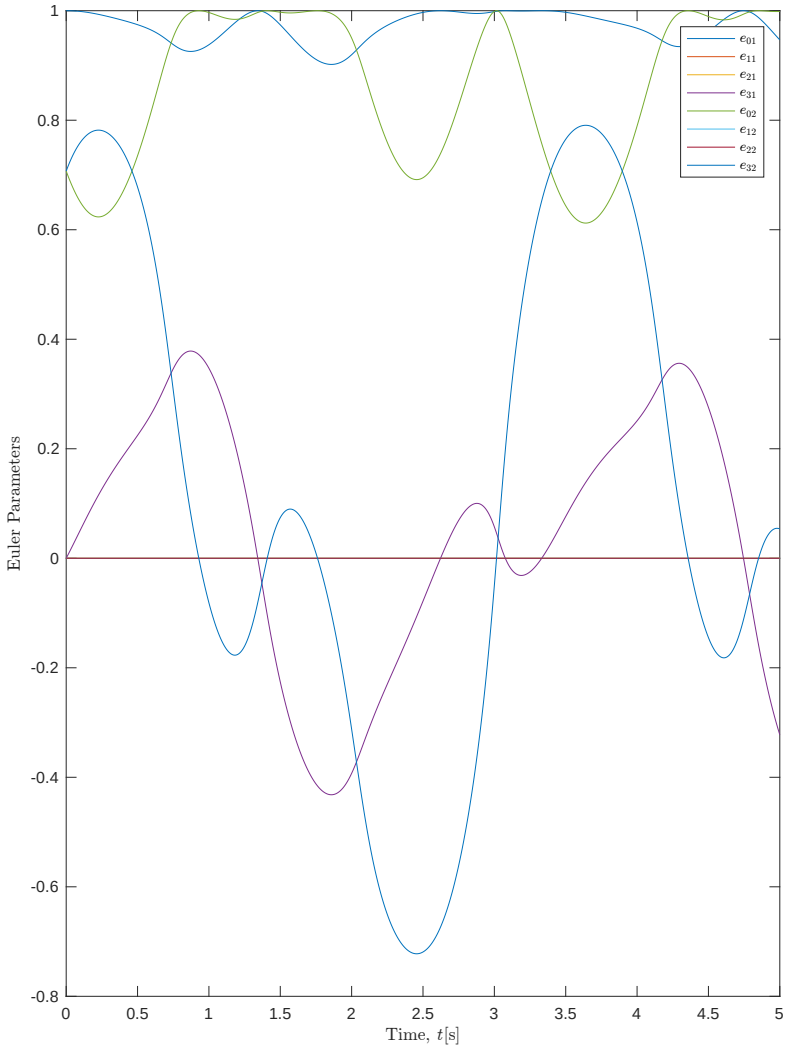


Figure 12: Time vs Euler parameters

3.2.4 Linear Velocities

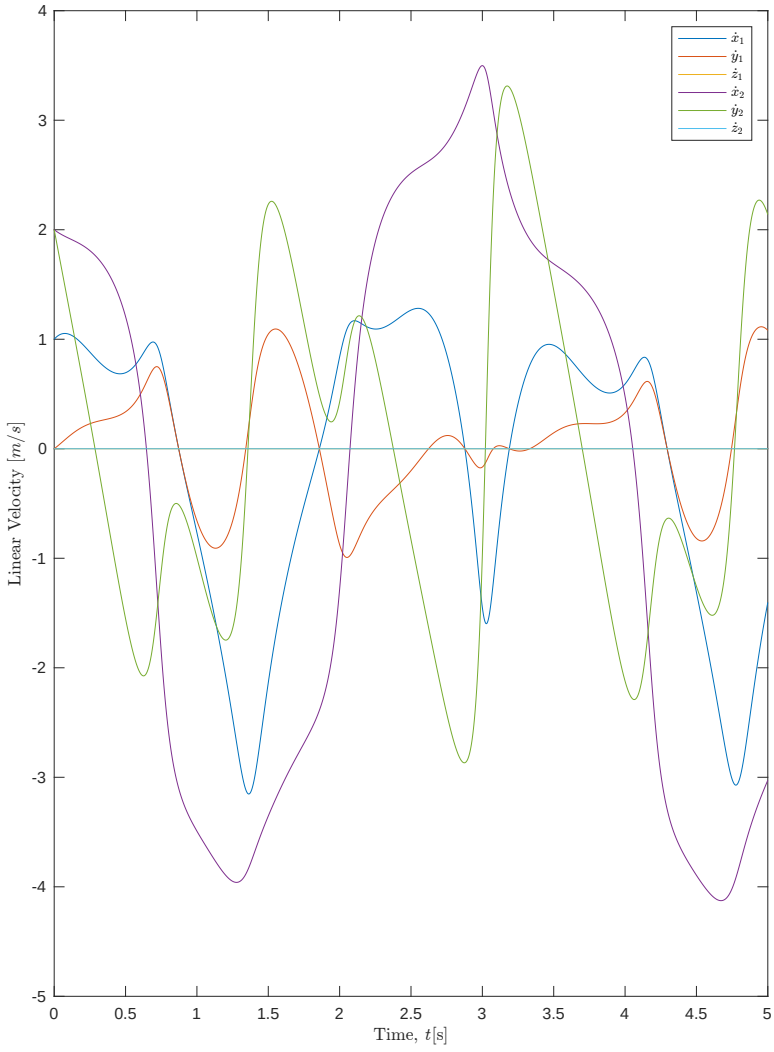


Figure 13: Time vs Linear Velocities

3.2.5 Euler Parameter Derivatives

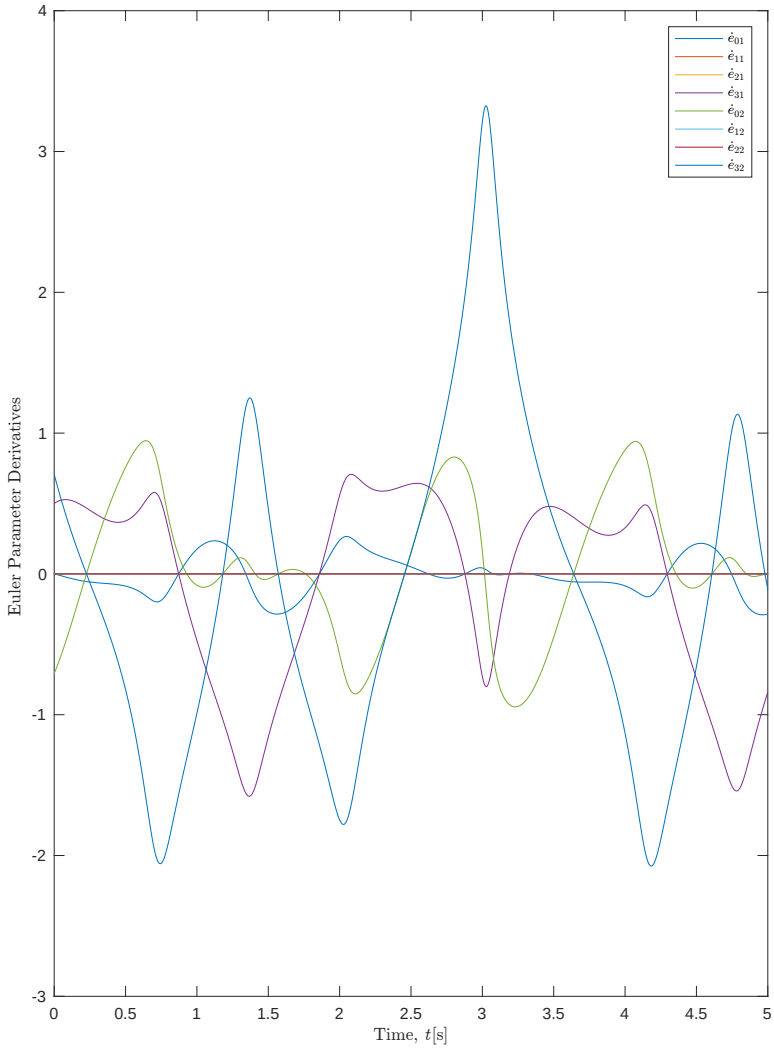


Figure 14: Time vs Euler Parameter Derivatives

3.3 Comparison of 2D and 3D library results

The plot below compares the results obtained from 2D and 3D library with zero initial velocities. Since both the plots match, the libraries are validated.

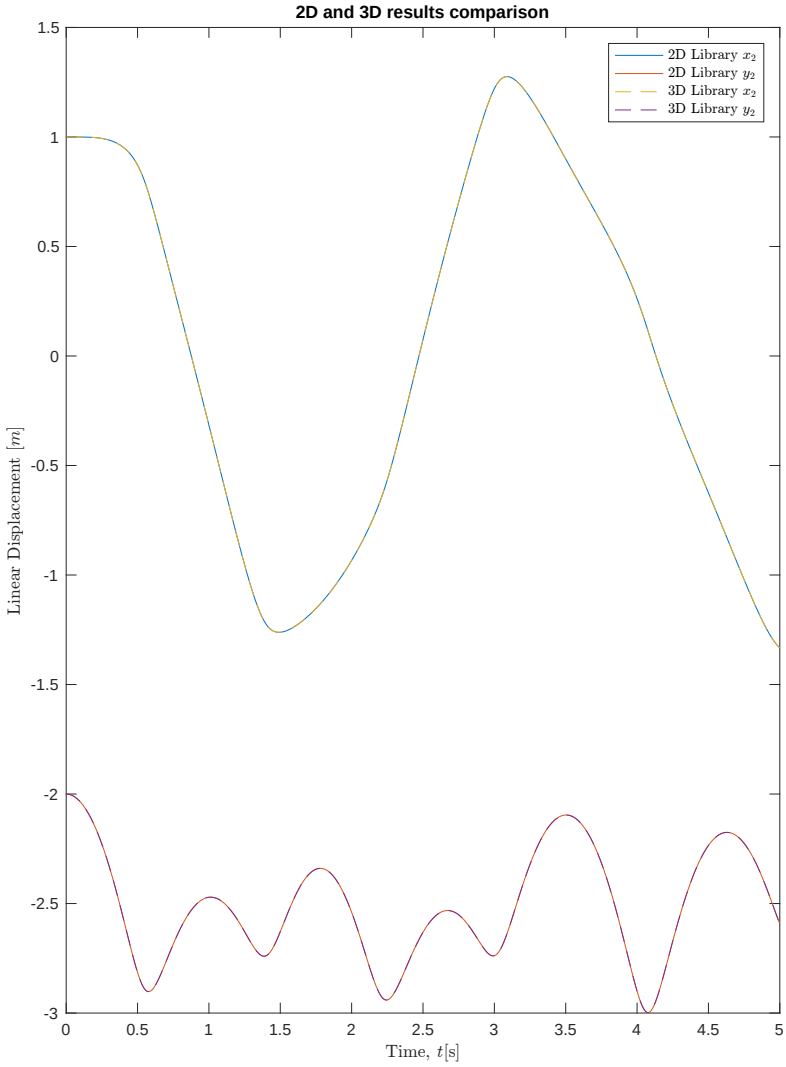


Figure 15: Time vs Linear Displacements

4 Other Results

4.1 Bipdeal Robot Simulation

The double pendulum was converted to a bipedal robot by changing the initial conditions. The program was simulated to the event where point C hit a point located $\frac{L}{3}$ below the ground. The following results were obtained.

4.1.1 Norm

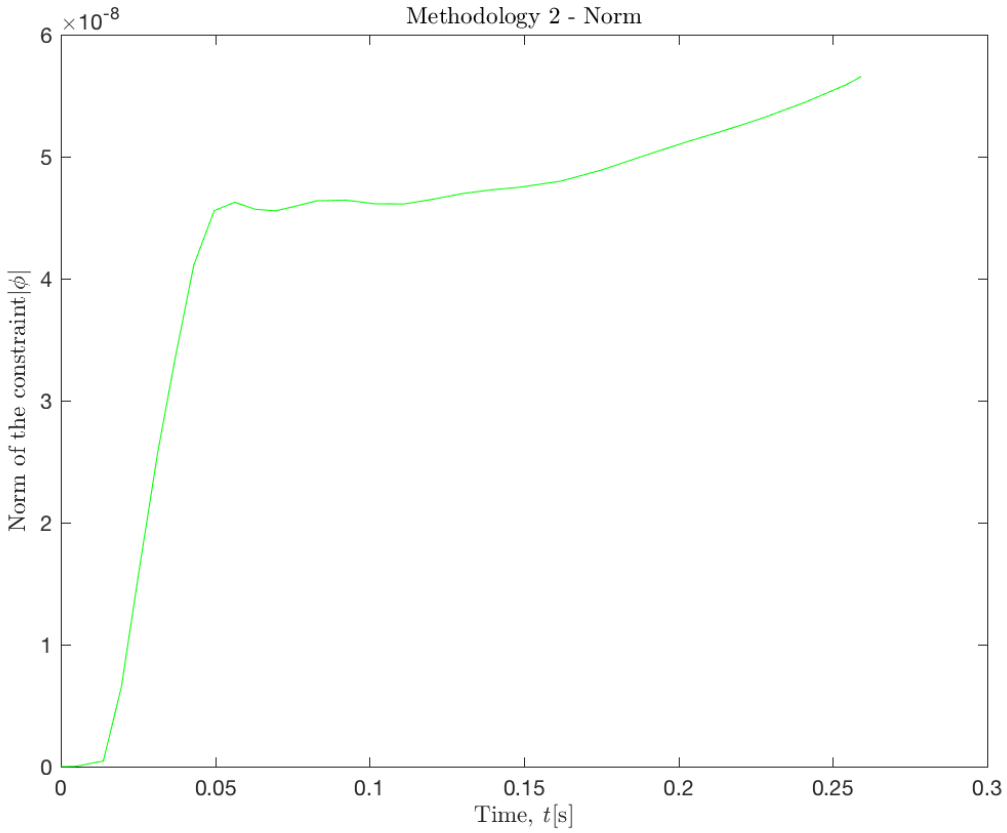


Figure 16: Time vs Norm

4.1.2 Linear Displacements

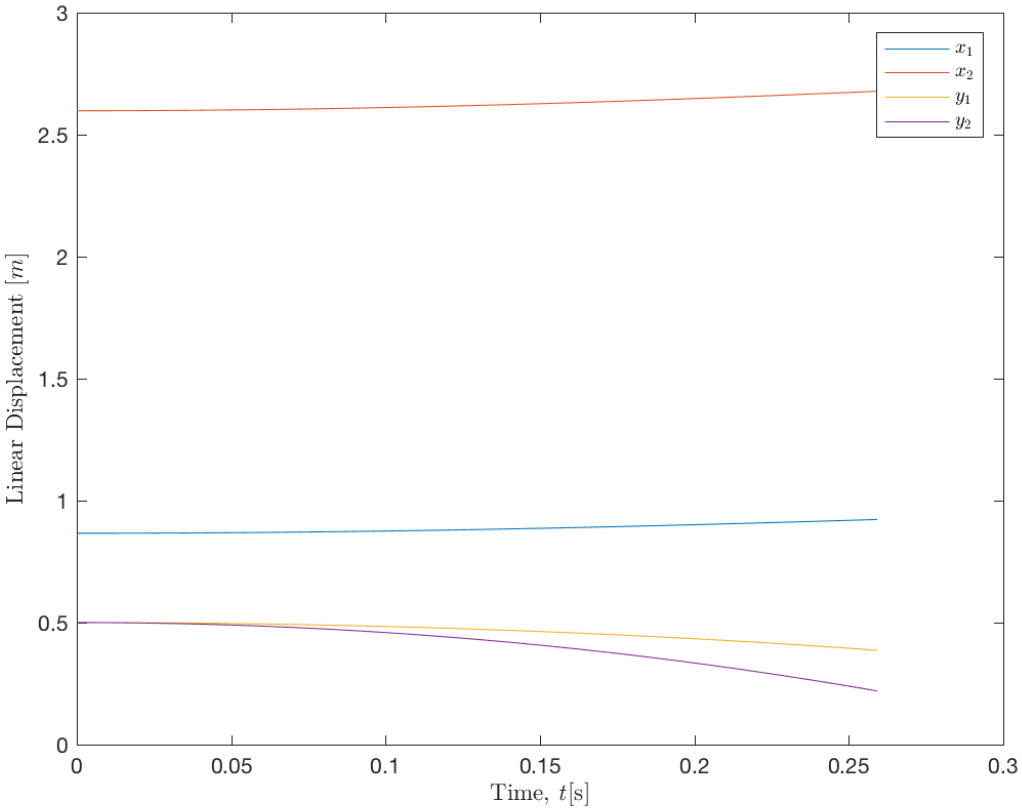


Figure 17: Time vs Linear Displacements

4.1.3 Angular Displacements

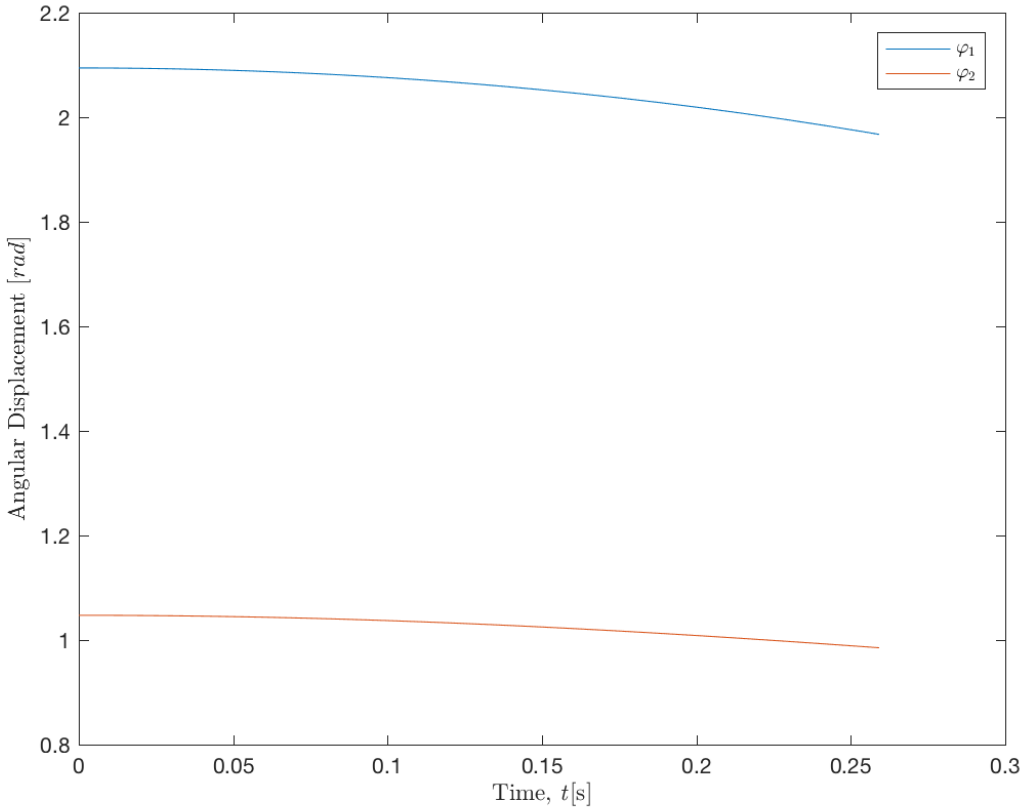


Figure 18: Time vs Angular Displacements

4.1.4 Linear Velocities

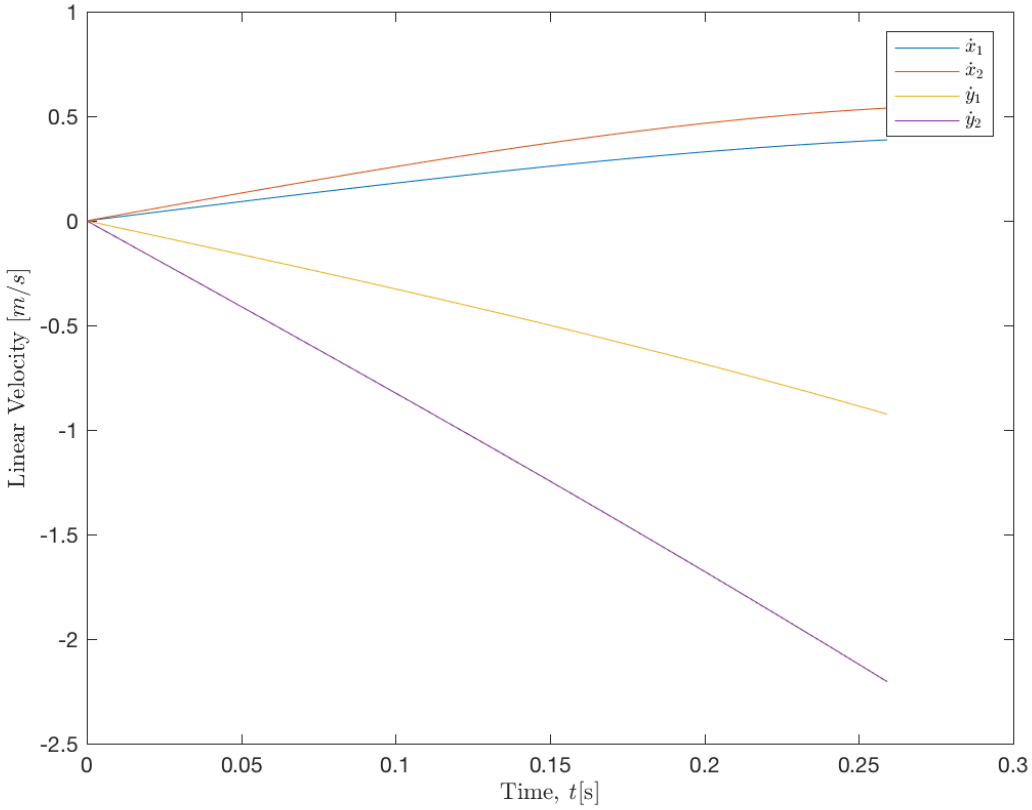


Figure 19: Time vs Linear Velocities

4.1.5 Angular Velocities

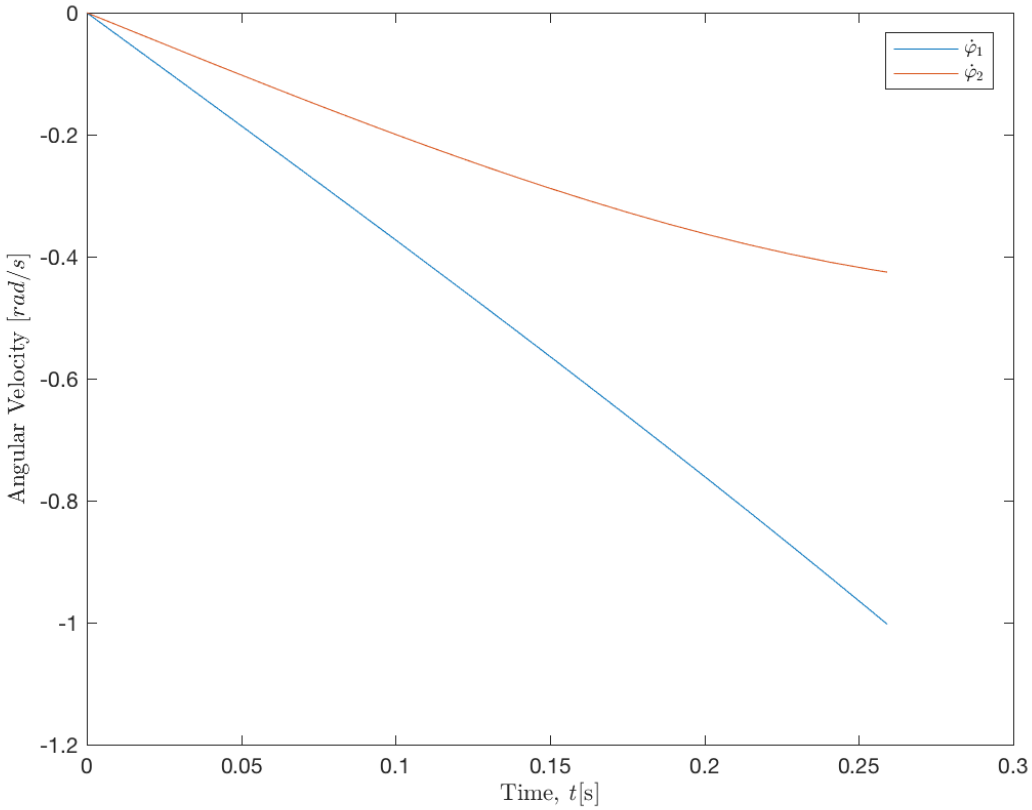


Figure 20: Time vs Angular Velocities

4.1.6 Linear Acceleration

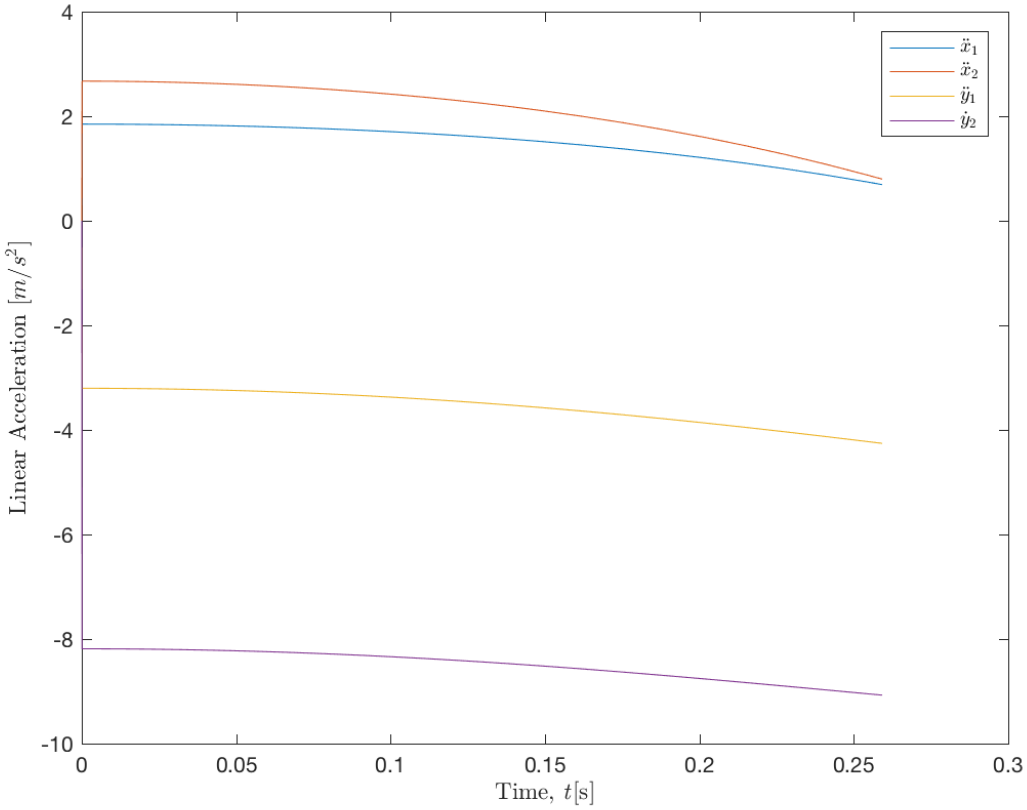


Figure 21: Time vs Linear Acceleration

4.1.7 Angular Acceleration

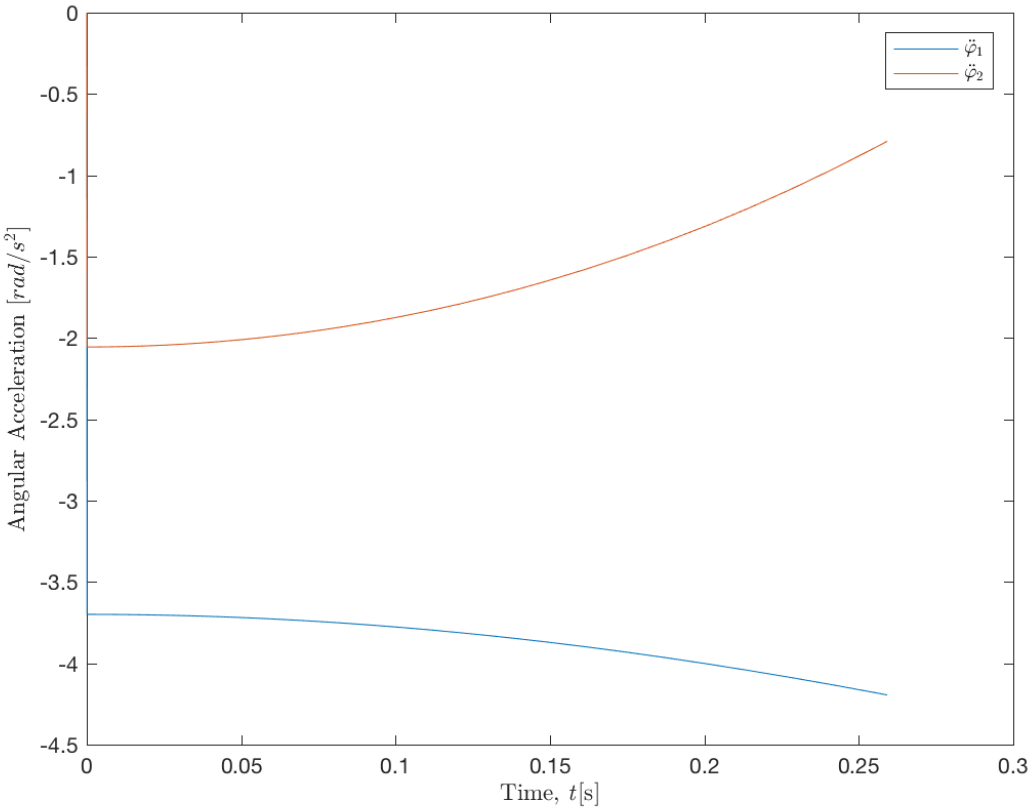


Figure 22: Time vs Angular Acceleration

5 Conclusion

The Multibody Dynamic System library was successfully created and validated for for both 2 dimensional and 3 dimensional systems. After working on this library I am confident I will be able to simulate complex multibody systems in a systematic way. This experience helped me gain knowledge on how multibody softwares work. My ability to solve problems that require rigorous coding has improved a lot. My recommendation for future classes would be to allow students to work in teams for library related assignments as I believe it will help them develop the library faster. Also it is easy to point out the bugs in the code build by one student while working as a team.