TMMS30 Lab 1

Eric Moringe (erimo668), David Wiman (davwi279)

February 7, 2023

Task 1

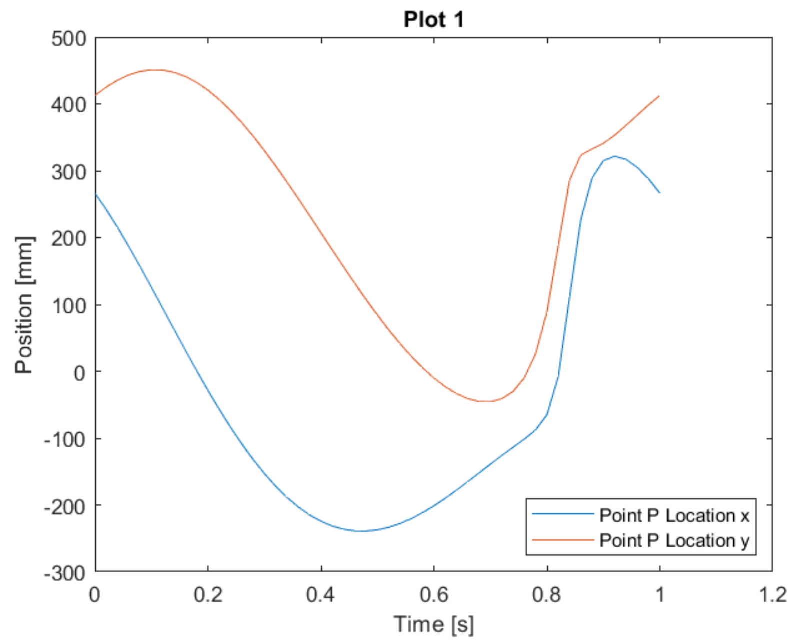

Figure 1: Location of point P as a function of time.

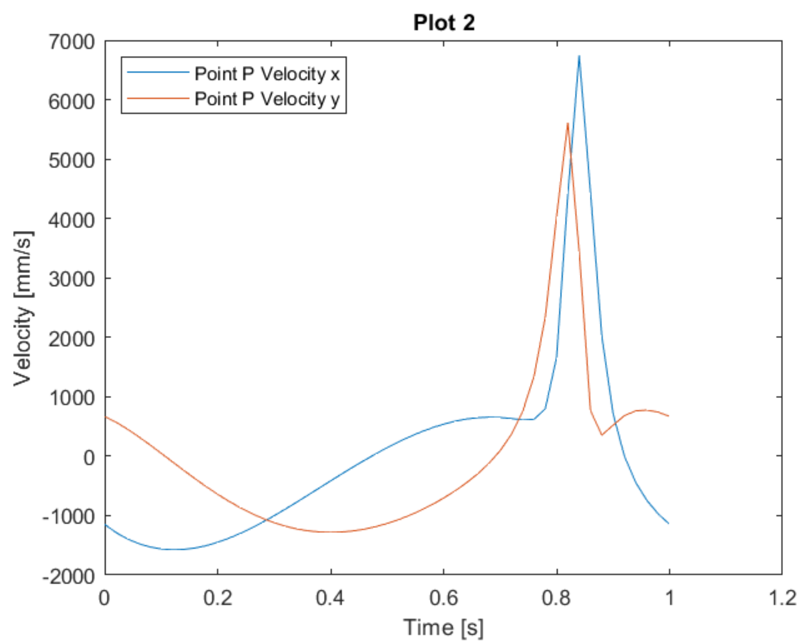

Figure 2: Velocity of point P as a function of time.

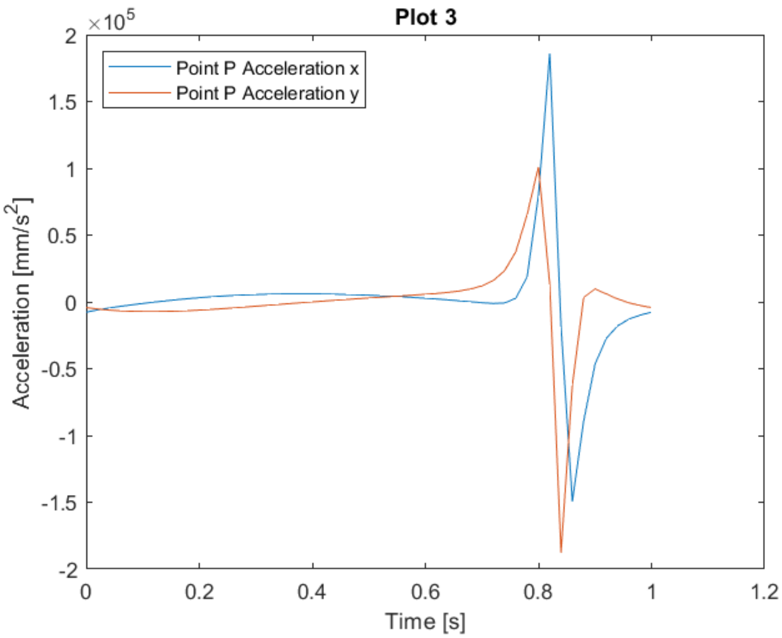

Figure 3: Accerelation of point P as a function of time.

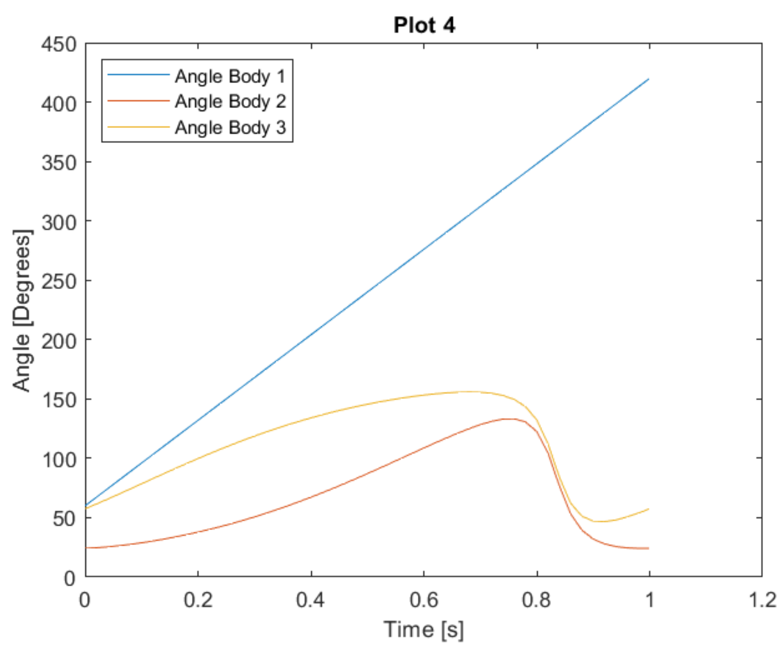

Figure 4: Angle of body 1, 2, and 3 as functions of time.

Task 2

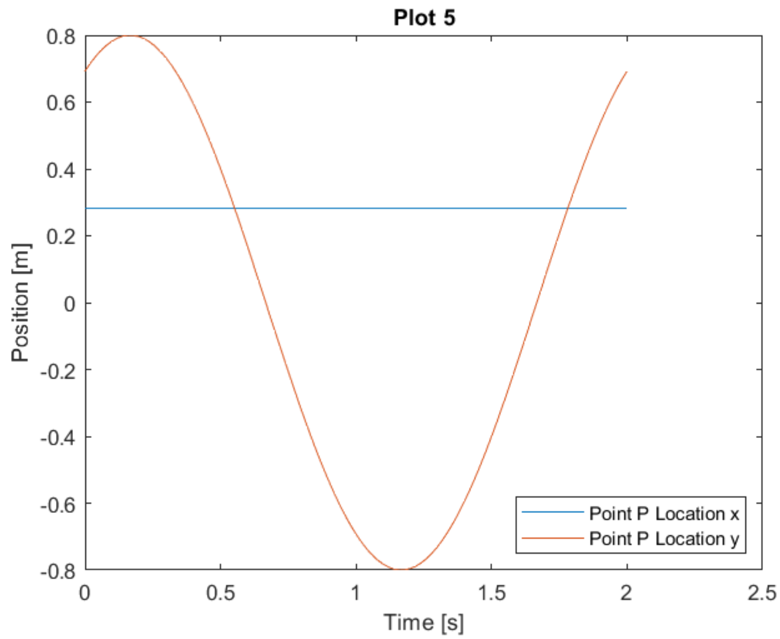

Figure 5: Location of point P as a function of time.

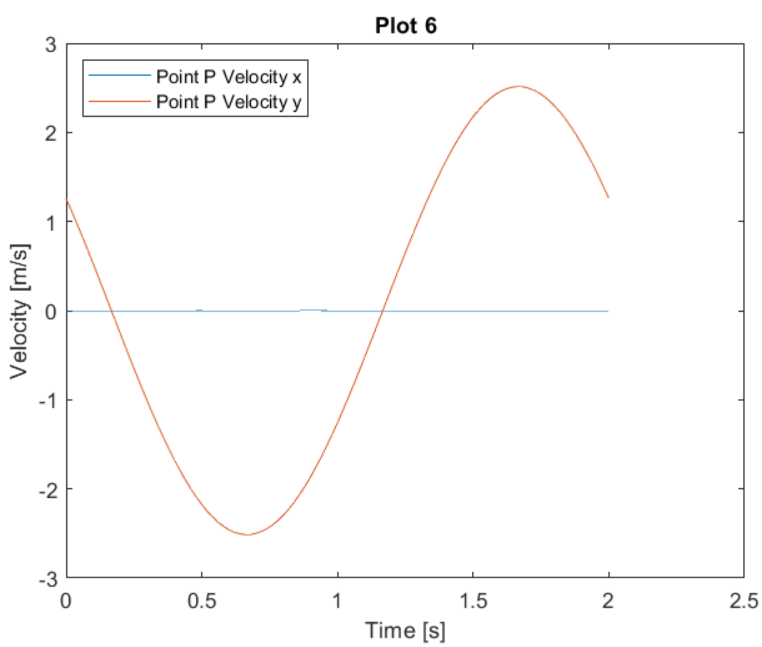

Figure 6: Velocity of point P as a function of time.

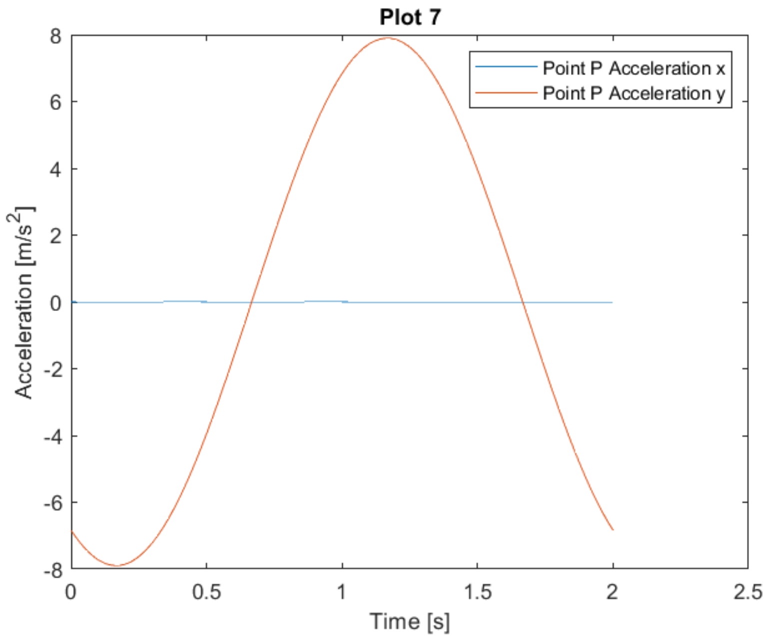

Figure 7: Acceleration of point P as a function of time.

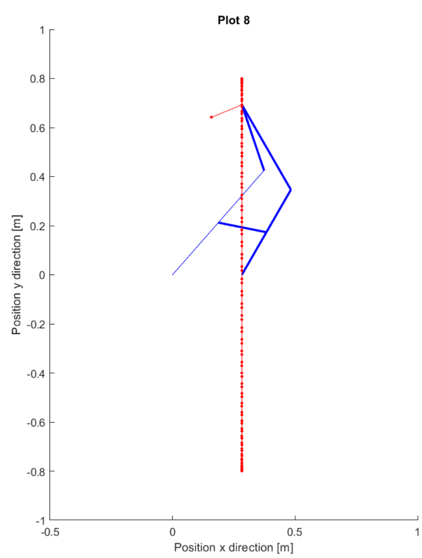

Figure 8: Path of point P as a function of time.

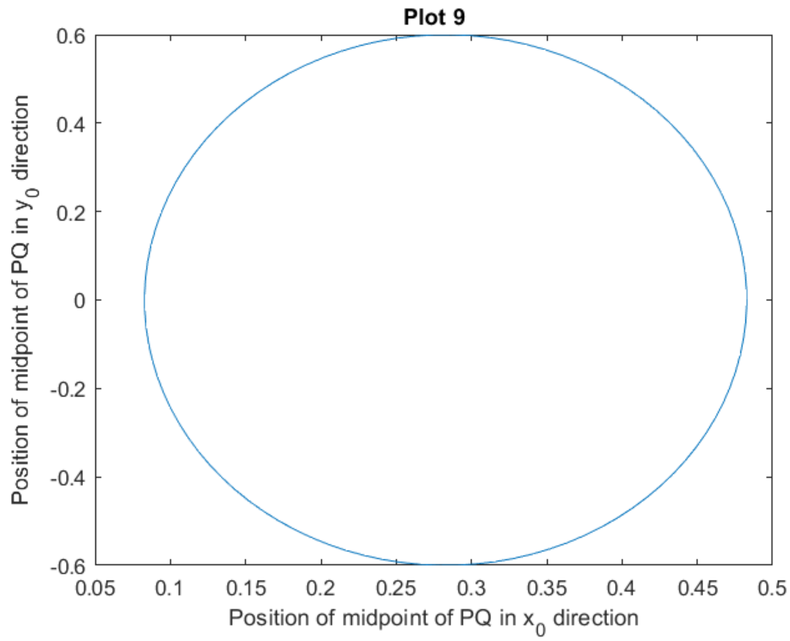

Figure 9: Location of the midpoint of body PQ in the y_0 direction as a functions of the same point's location in the x_0 direction.

Task 3

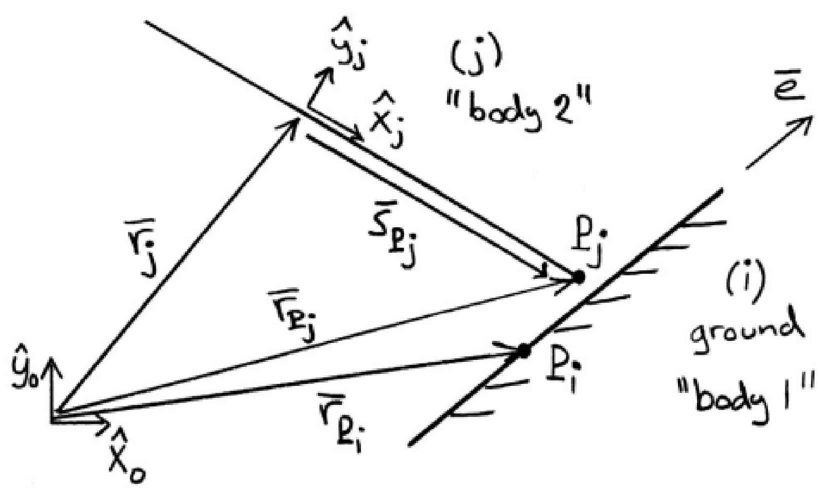

Figure 10: Figure of an RPg joint.

(a) The constraint equation that a RPg joint gives rise to can be described as

$$\overline{H}^{rpg} = \tilde{e}^T(\overline{r}_{P_j} - \overline{r}_{P_i}) = 0. \tag{1}$$

This can be expanded to

$$\overline{H}^{rpg} = \tilde{e}^T(\overline{r}_j + \overline{s}_{P_j} - \overline{r}_{P_i}) = 0. \tag{2}$$

When we differentiate this, we get

$$\dot{\overline{H}}^{rpg} = \tilde{e}^T(\dot{\overline{r}}_j + \dot{\theta}_j \tilde{s}_{P_j}), \tag{3}$$

which we can use to get the Jacobian

$$\overline{H}^{rpg}_{q_j} = [\tilde{e}^T \quad \overline{e}^T \overline{s}_{P_j}] \tag{4}$$

as $\tilde{e}^T \tilde{s}_{P_j} = \overline{e}^T \overline{s}_{P_j}$.

If we differentiate again, we get

$$\ddot{\overline{H}}^{rpg} = \tilde{e}^T(\ddot{\overline{r}}_j + \ddot{\theta}_j \tilde{s}_{P_j} - \dot{\theta}_j^2 \overline{s}_{P_j}). \tag{5}$$

With this, we can determine the $\overline{c}^{rpg}$.

$$\overline{c}^{rpg} = \tilde{e}^T \dot{\theta}_j^2 \overline{s}_{P_j} \tag{6}$$

Implementing these results in matlab yields the following code lines in the files get_H_kin.m, get_jac_kin.m and get_c_kin.m respectively.

get_H_kin.m	s_j=rot(theta_j,s_j_prime);
	e_tilde=[-e(2);e(1)];
	h= e_tilde'*(r_j+s_j-r_pi);

get_jac_kin.m	s_j = rot(theta_j,s_j_prime);
	e_tilde=[-e(2);e(1)];
	jac=[e_tilde' e'*s_j];

get_c_kin.m	s_j = rot(theta_j,s_j_prime);
	e_tilde=[-e(2);e(1)];
	c = e_tilde'*theta_dot_j$^2 *$ s_j;

Table 1: Table of Matlab implementations.

(b) The requested graphs look as follows:

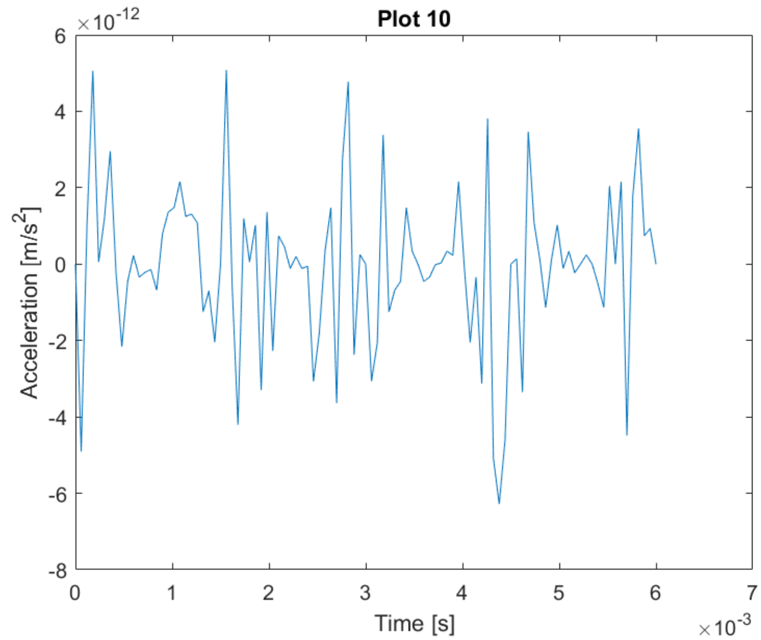

Figure 11: Acceleration of point P in x_0 direction.

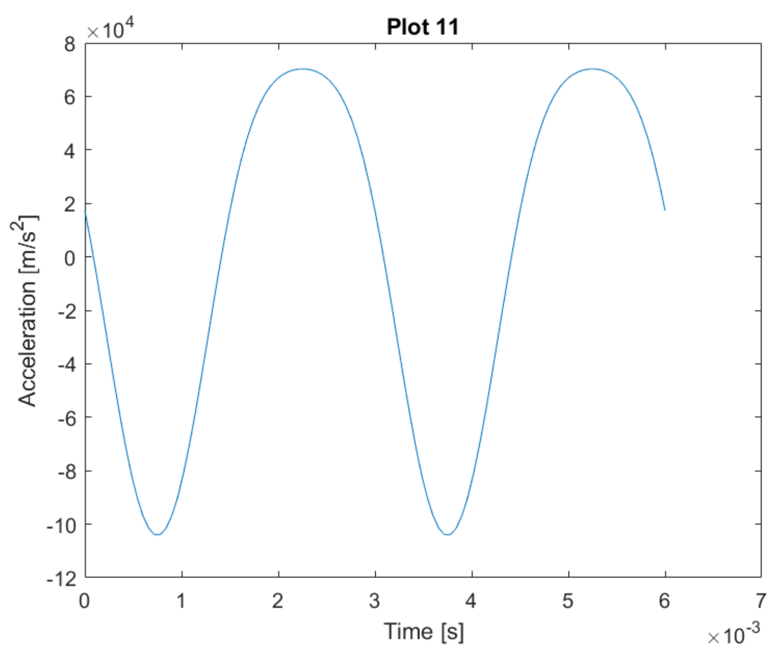

Figure 12: Acceleration of point P in y_0 direction.

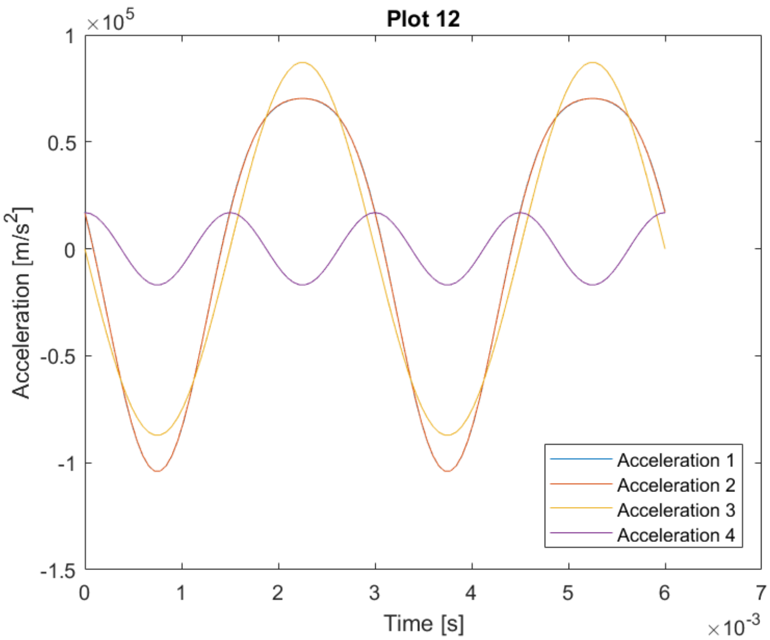

Figure 13: Acceleration of point P in y_0 direction using different formulas. Acceleration 1 and 2 are so similar it is hard see their differences in this plot, hence it only looks like three graphs.

(c) The requested graphs look as follows:

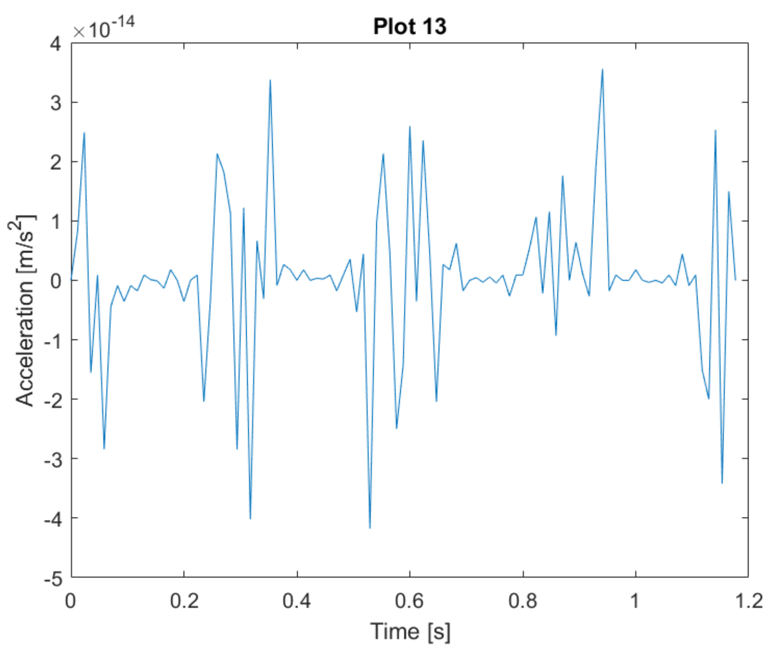

Figure 14: Acceleration of point P in x_0 direction.

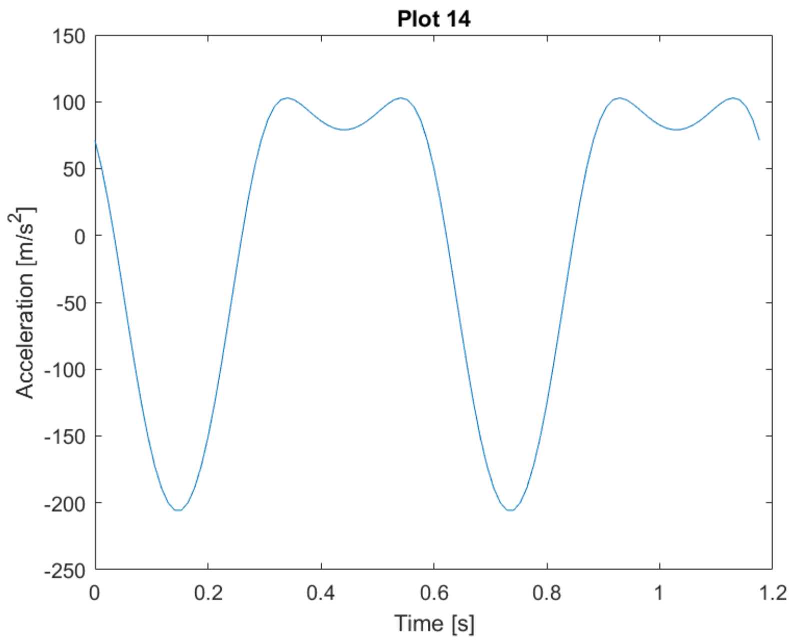

Figure 15: Acceleration of point P in y_0 direction.

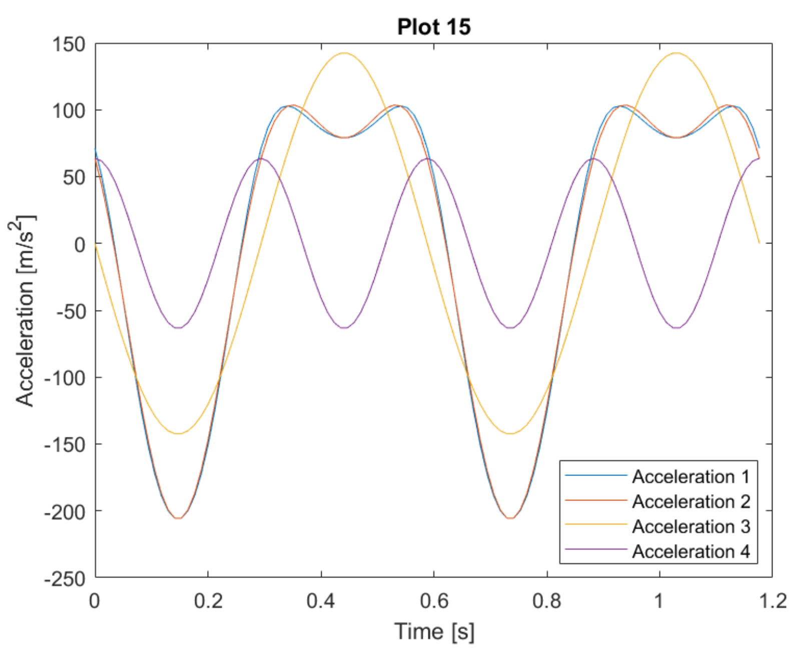

Figure 16: Acceleration of point P in y_0 direction using different formulas.

Task 4

The code implemented for the improvements of order 1 and order 2 respectively were:

Order 1 Improvement	q = q + t_inc*q_dot;
Order 2 Improvement	q = q + t_inc*q_dot+1/2*t_inc$^2 *$ q_ddot;

Table 2: Table of Matlab implementations.

Number of Newton iterations per timestep without any improvements:

$$its\_hist = [3,3,3,3,3,4,5,6,5,4,4,3,3,3,2,2,2,2,2,2,2]. \tag{7}$$

The average was 3.1429.

Number of Newton iterations per timestep with improvement of order 1:

$$its\_hist = [3,2,2,2,2,3,4,5,4,3,3,2,2,2,1,1,1,1,1,1,1]. \tag{8}$$

The average was 2.1905.

Number of Newton iterations per timestep with improvement of order 2:

$$its\_hist = [3,1,1,1,2,2,3,4,2,3,2,2,1,1,1,1,1,1,1,1,1]. \tag{9}$$

The average was 1.6667.