Homework 3

Kai Li from MRSD

Question 4 (PS2)

Please refer to rlocus.m for details.

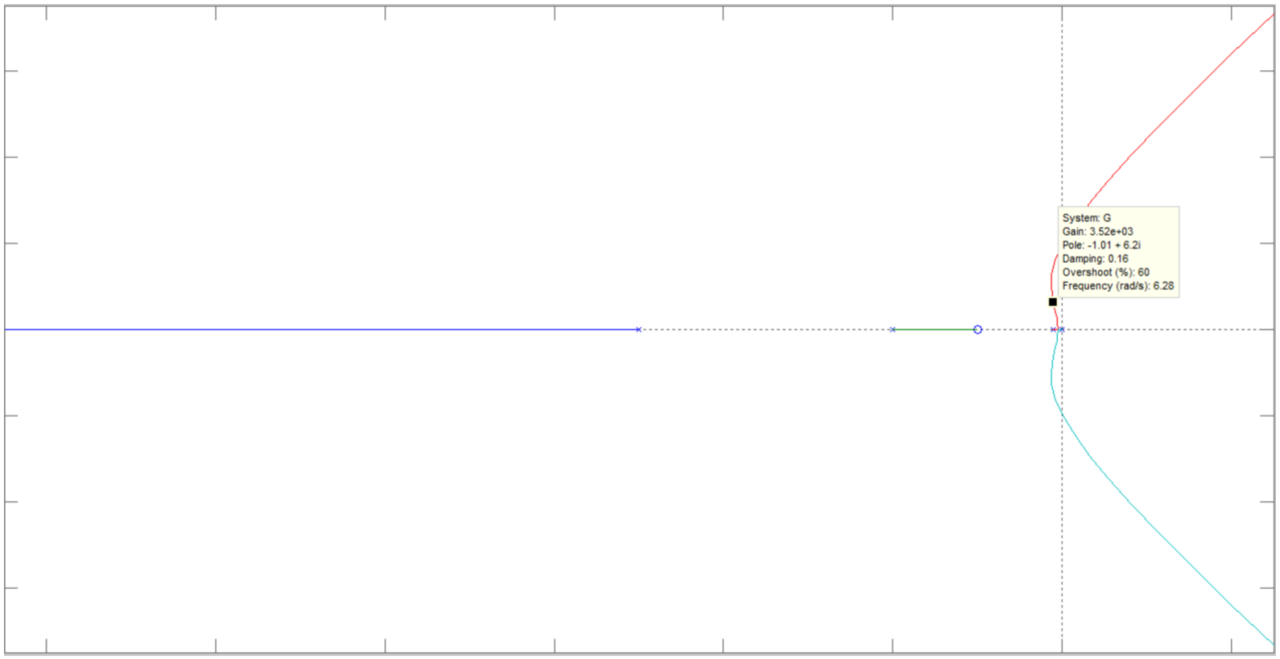
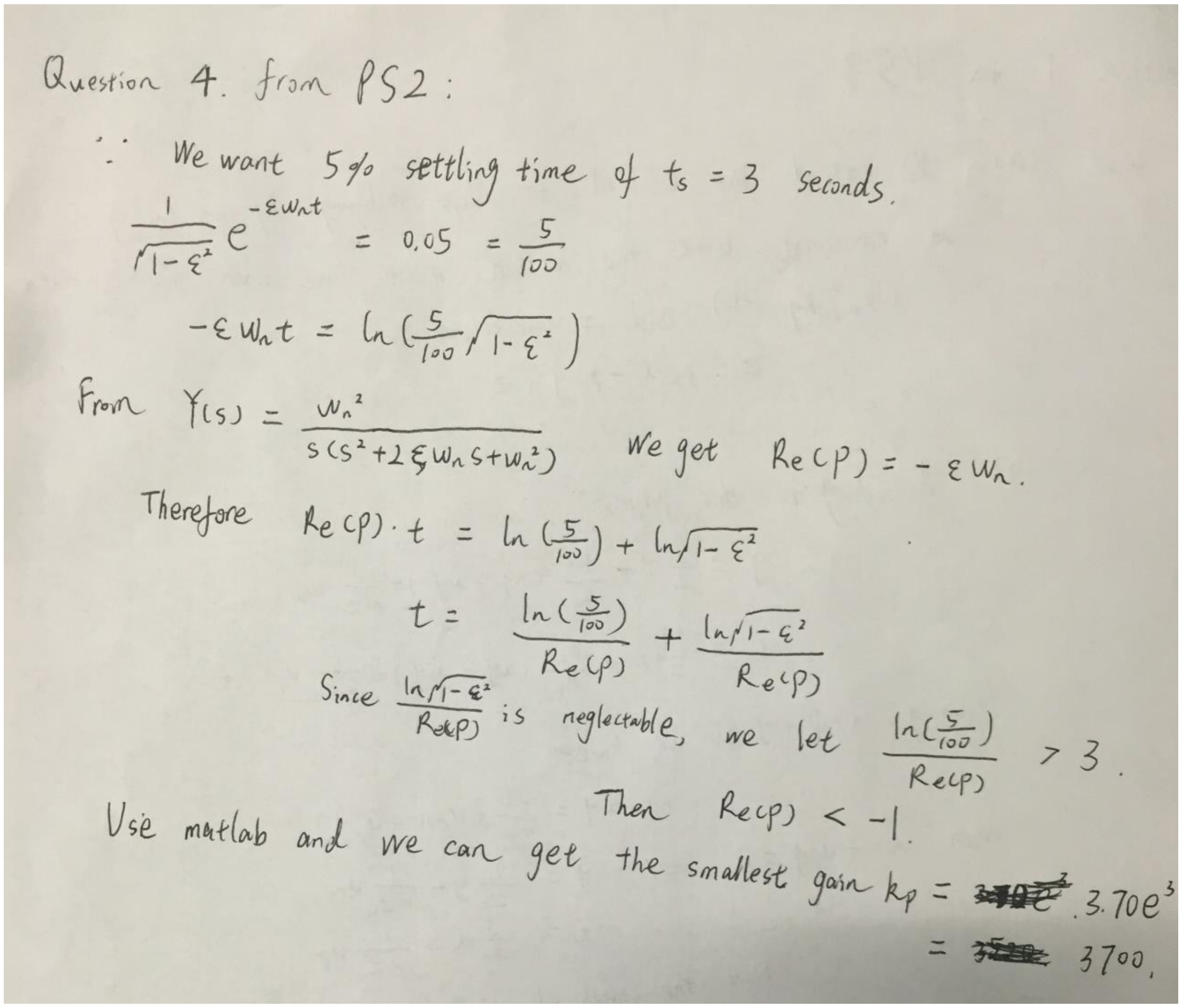


Figure 1. Root locus plot



Question 4. from PS2:

$\because$ We want 5% settling time of $t_s = 3$ seconds.

$$\frac{1}{\sqrt{1-\varepsilon^2}} e^{-\varepsilon w_n t} = 0.05 = \frac{5}{100}$$

$$-\varepsilon w_n t = \ln\left(\frac{5}{100}\sqrt{1-\varepsilon^2}\right)$$

From $Y(s) = \dfrac{w_n^2}{s(s^2+2\xi w_n s + w_n^2)}$ We get $Re(p) = -\varepsilon w_n$.

Therefore $Re(p) \cdot t = \ln\left(\frac{5}{100}\right) + \ln\sqrt{1-\varepsilon^2}$

$$t = \frac{\ln\left(\frac{5}{100}\right)}{Re(p)} + \frac{\ln\sqrt{1-\varepsilon^2}}{Re(p)}$$

Since $\dfrac{\ln\sqrt{1-\varepsilon^2}}{Re(p)}$ is neglectable, we let $\dfrac{\ln\left(\frac{5}{100}\right)}{Re(p)} > 3$.

Then $Re(p) < -1$.

Use matlab and we can get the smallest gain $k_p =$ [illegible] $3.70e^3$

$=$ [illegible] 3700.

Figure 2. Q4_Write Up

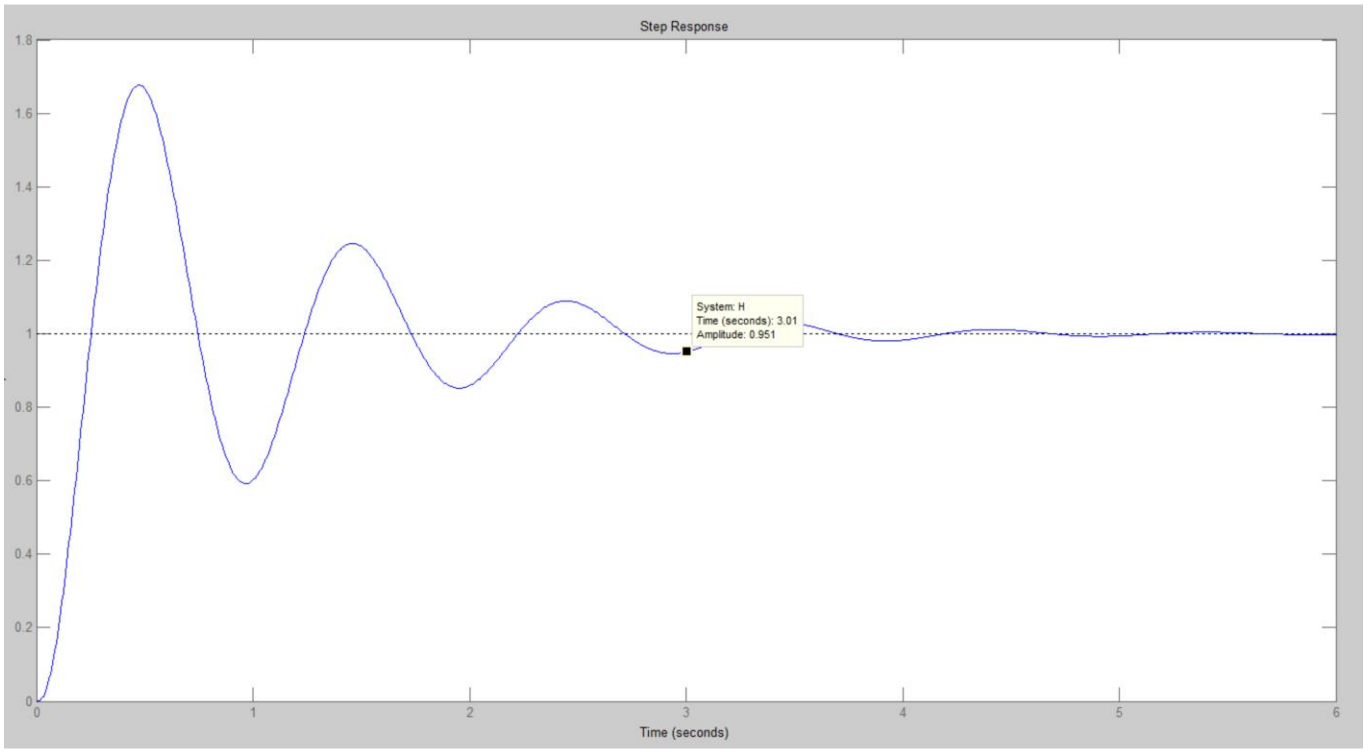


Figure 3. Step response used for testing

We can see the Kp=3700 satisfy the required conditions.

Question 5 (PS2)

After manually tuning it, my final gains are:

```
kp =360;
ki = 102.5;
kd =580;
```

The step information:

```
RiseTime: 0.5000

SettlingTime: 7.8574

SettlingMin: 0.9005

SettlingMax: 1.0389

Overshoot: 3.8893

Undershoot: 0

Peak: 1.0389

PeakTime: 4.4446
```

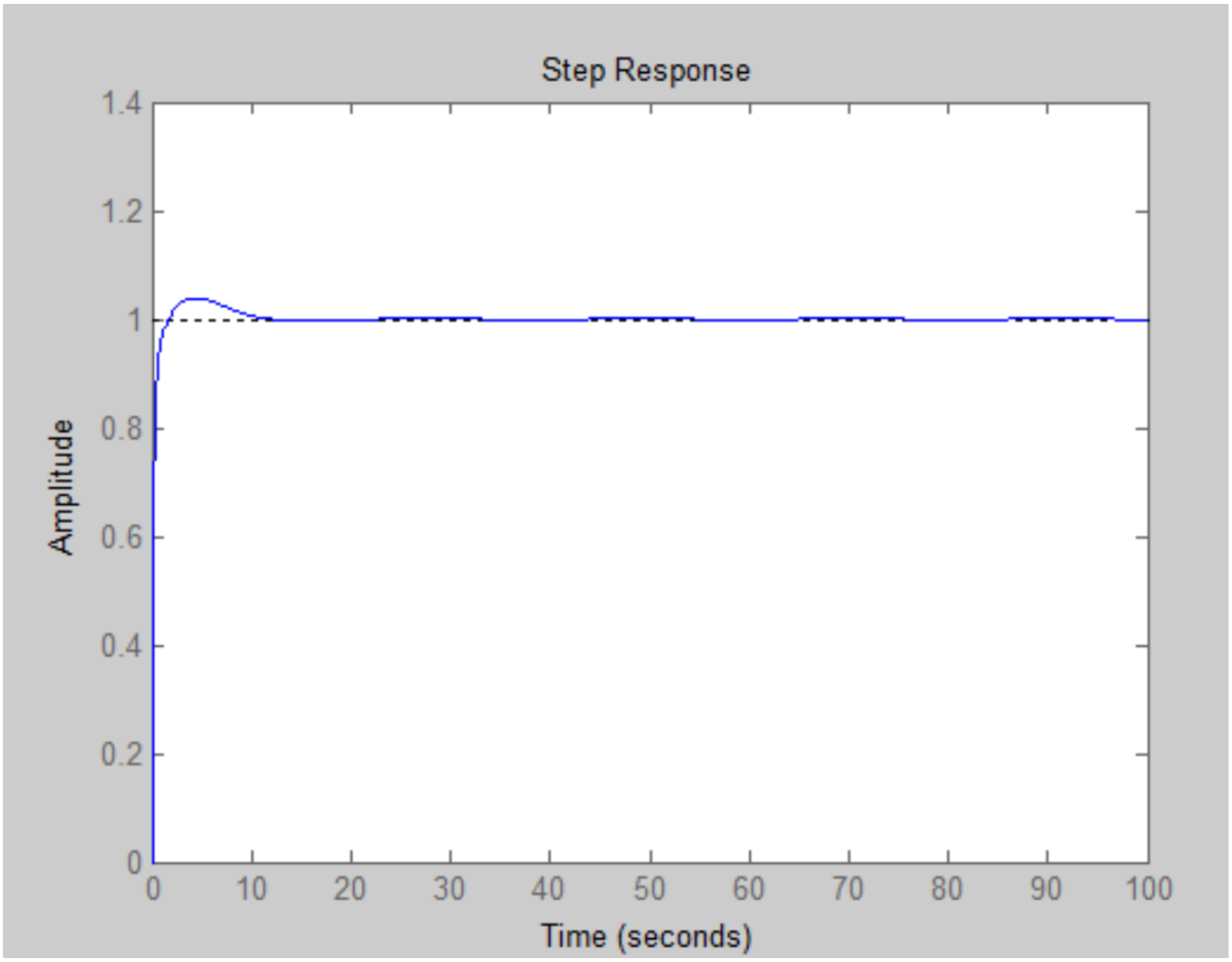
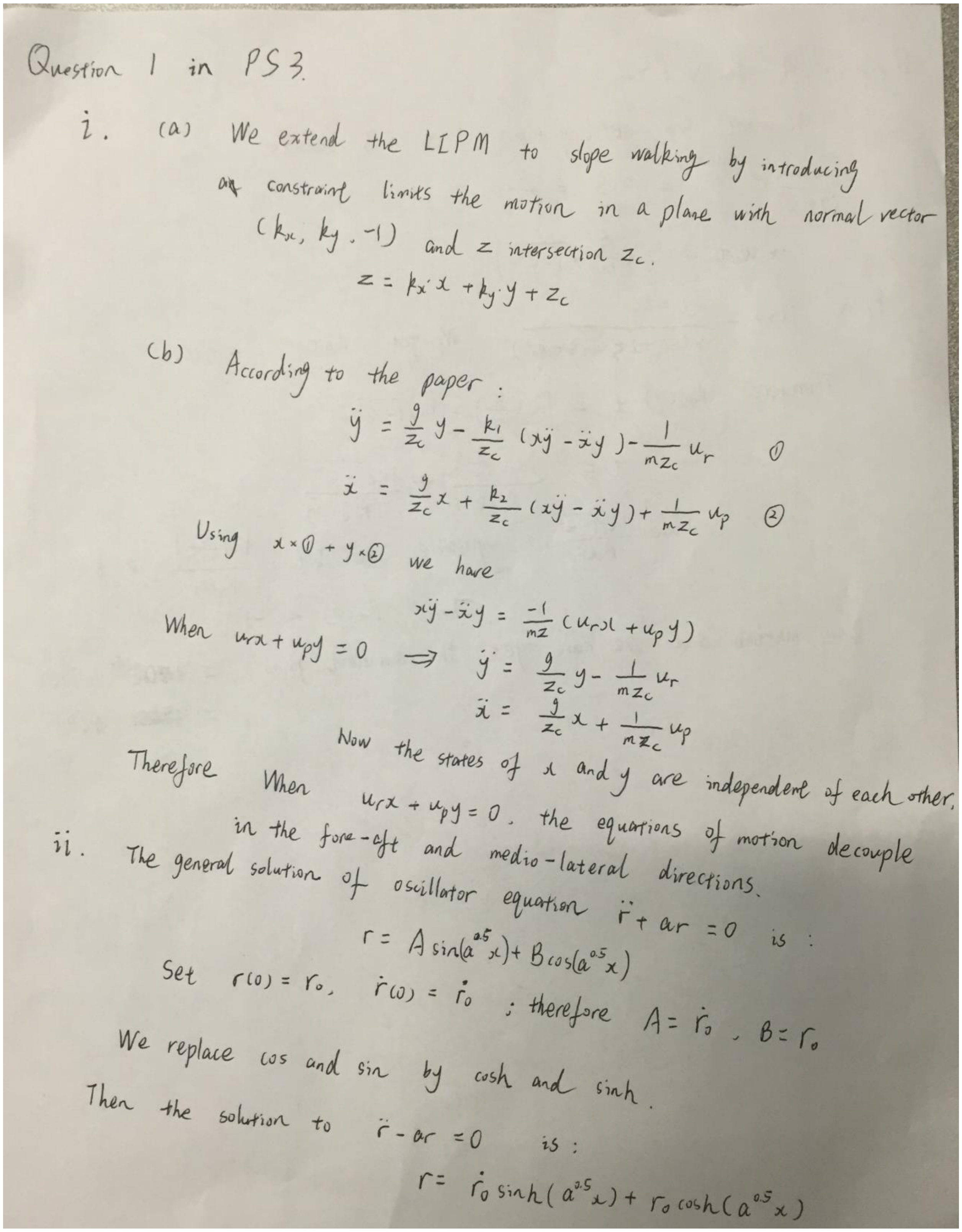


Figure 4. A plot of the resulting closed loop step response

Please refer to tuning.m for details.

Question 1 (PS3)



Question 1 in PS3.

i. (a) We extend the LIPM to slope walking by introducing ~~a~~ constraint limits the motion in a plane with normal vector $(k_x, k_y, -1)$ and z intersection z_c.

$$z = k_x \cdot x + k_y \cdot y + z_c$$

(b) According to the paper :

$$\ddot{y} = \frac{g}{z_c} y - \frac{k_1}{z_c}(x\ddot{y} - \ddot{x}y) - \frac{1}{mz_c} u_r \quad ①$$

$$\ddot{x} = \frac{g}{z_c} x + \frac{k_2}{z_c}(x\ddot{y} - \ddot{x}y) + \frac{1}{mz_c} u_p \quad ②$$

Using $x \times ① + y \times ②$ we have

$$x\ddot{y} - \ddot{x}y = \frac{-1}{mz}(u_r x + u_p y)$$

When $u_r x + u_p y = 0$ $\Rightarrow$

$$\ddot{y} = \frac{g}{z_c} y - \frac{1}{mz_c} u_r$$

$$\ddot{x} = \frac{g}{z_c} x + \frac{1}{mz_c} u_p$$

Now the states of x and y are independent of each other.

Therefore When $u_r x + u_p y = 0$. the equations of motion decouple in the fore-aft and medio-lateral directions.

ii. The general solution of oscillator equation $\ddot{r} + ar = 0$ is :

$$r = A\sin(a^{0.5}x) + B\cos(a^{0.5}x)$$

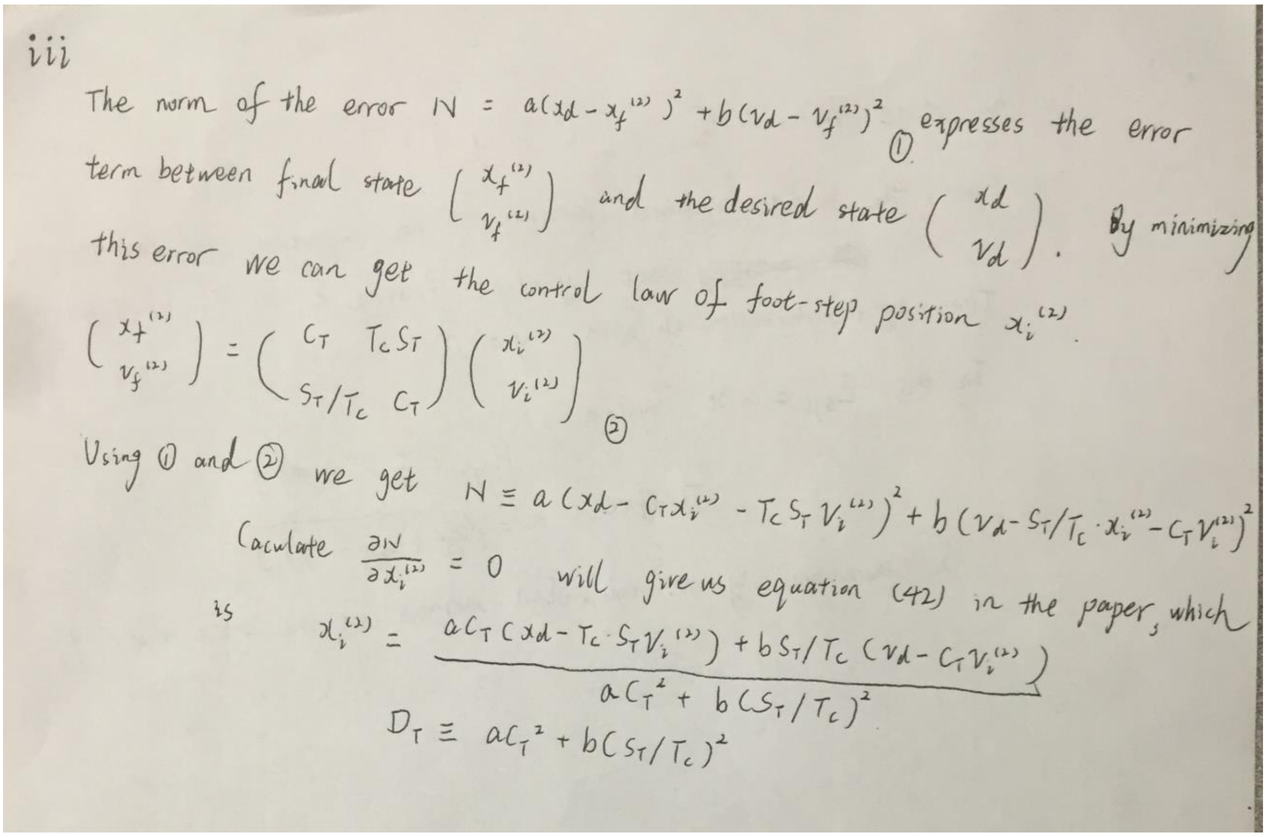
Set $r(0) = r_0$, $\dot{r}(0) = \dot{r}_0$; therefore $A = \dot{r}_0$, $B = r_0$

We replace cos and sin by cosh and sinh.

Then the solution to $\ddot{r} - ar = 0$ is :

$$r = \dot{r}_0 \sinh(a^{0.5}x) + r_0\cosh(a^{0.5}x)$$

Figure 5. Q_1.1 & 1.2 Write Up



iii

The norm of the error $N = a(x_d - x_f^{(2)})^2 + b(v_d - v_f^{(2)})^2$ ① expresses the error term between final state $\begin{pmatrix} x_f^{(2)} \\ v_f^{(2)} \end{pmatrix}$ and the desired state $\begin{pmatrix} x_d \\ v_d \end{pmatrix}$. By minimizing this error we can get the control law of foot-step position $x_i^{(2)}$.

$$\begin{pmatrix} x_f^{(2)} \\ v_f^{(2)} \end{pmatrix} = \begin{pmatrix} C_T & T_c S_T \\ S_T/T_c & C_T \end{pmatrix} \begin{pmatrix} x_i^{(2)} \\ v_i^{(2)} \end{pmatrix} \quad ②$$

Using ① and ② we get $N \equiv a(x_d - C_T x_i^{(2)} - T_c S_T v_i^{(2)})^2 + b(v_d - S_T/T_c \cdot x_i^{(2)} - C_T v_i^{(2)})^2$

Caculate $\frac{\partial N}{\partial x_i^{(2)}} = 0$ will give us equation (42) in the paper, which is

$$x_i^{(2)} = \frac{a C_T (x_d - T_c \cdot S_T v_i^{(2)}) + b S_T/T_c (v_d - C_T v_i^{(2)})}{a C_T^2 + b(S_T/T_c)^2}$$

$$D_T \equiv a C_T^2 + b(S_T/T_c)^2$$

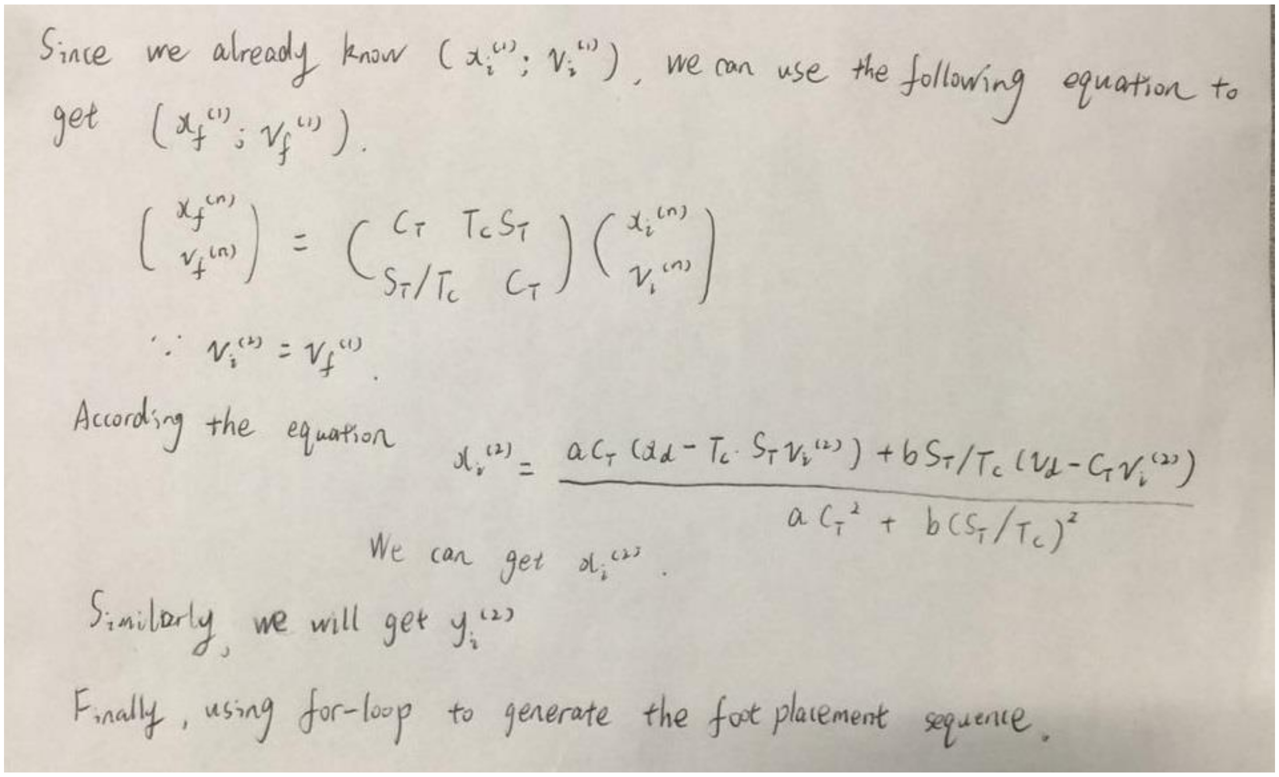
Figure 6. Q_1.3 Write Up

Q1.4

Here we assume the initial speed is (0.1,0.1), and the initial and final position are (0,0) and (0,0). And we set the parameters as follows:

```
% Weight
a = 1;
b = 1;

% Constant
Ts = 0.2;
g = 9.81;
Zc = 1;
```



Since we already know $(x_i^{(1)}; v_i^{(1)})$, we can use the following equation to get $(x_f^{(1)}; v_f^{(1)})$.

$$\begin{pmatrix} x_f^{(n)} \\ v_f^{(n)} \end{pmatrix} = \begin{pmatrix} C_T & T_c S_T \\ S_T/T_c & C_T \end{pmatrix} \begin{pmatrix} x_i^{(n)} \\ v_i^{(n)} \end{pmatrix}$$

$\because v_i^{(2)} = v_f^{(1)}$.

According the equation $$x_i^{(2)} = \frac{aC_T(x_d - T_c \cdot S_T v_i^{(2)}) + bS_T/T_c(v_d - C_T v_i^{(2)})}{aC_T^2 + b(S_T/T_c)^2}$$

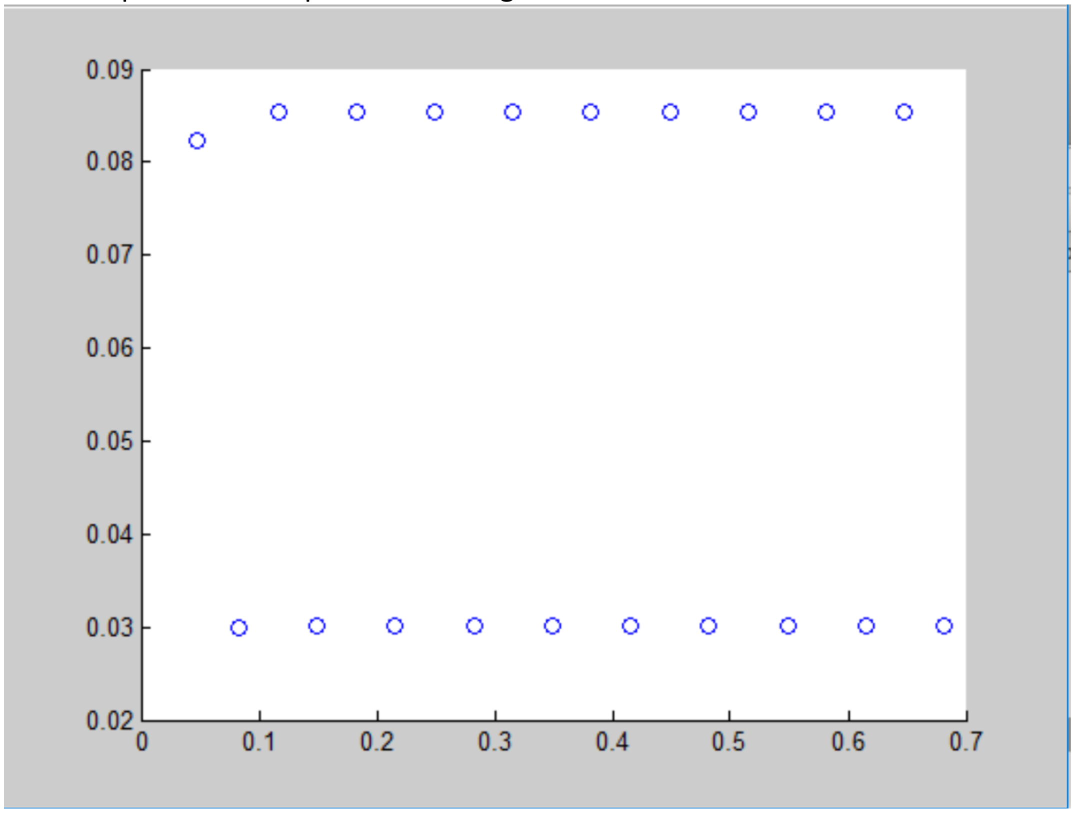
We can get $x_i^{(2)}$.

Similarly, we will get $y_i^{(2)}$

Finally, using for-loop to generate the foot placement sequence.

Figure 7. Q1_4 Write Up

The foot placement sequence for straight line locomotion is below:



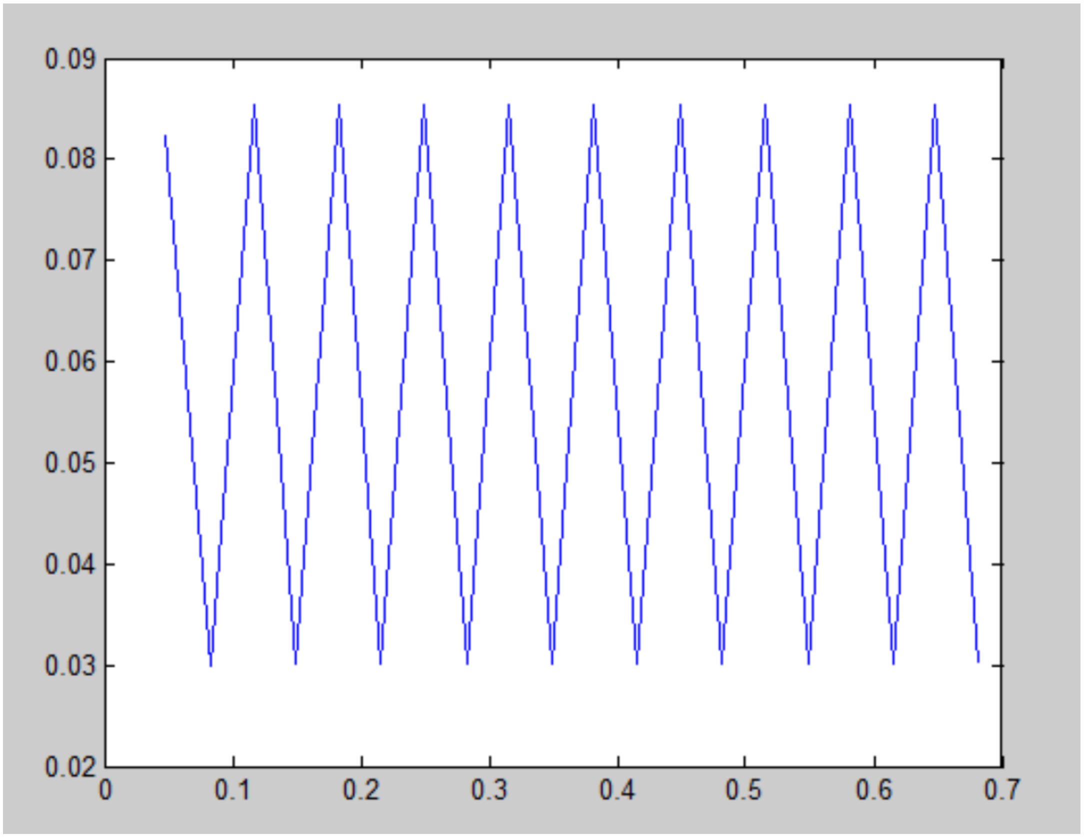


Figure 8. foot placement sequence

Please refer to Q1_4.m, xfootplace.m and yfootplace.m for detailed implementation.

Q1.5 (Optional)

Here I applied a rotation angle of pi/10 every step, please refer to Q1_5 for implementation. And the foot placement sequence is as follows:

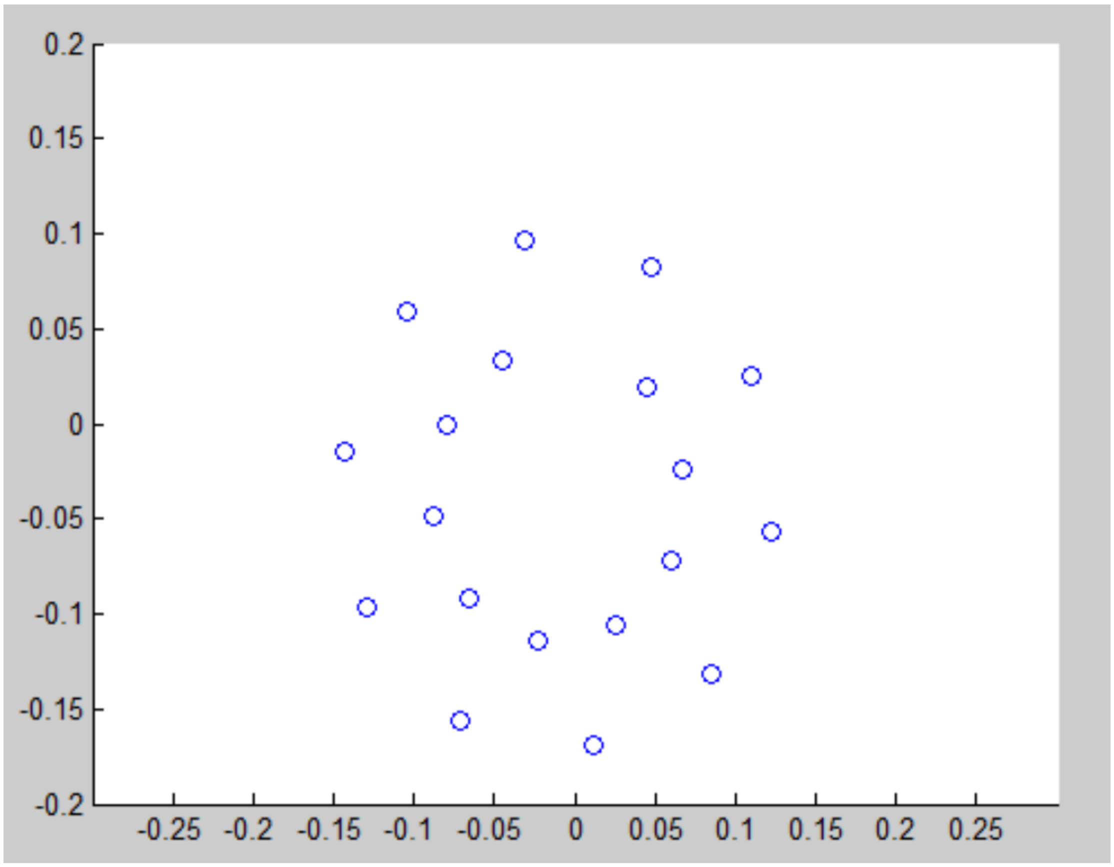


Figure 9. The foot placement plot

Question 2 (PS3)

Q2.1

Here I used ode45 to implement a numerical integration of the model dynamics. The stance_situation.m and flying_situation.m are used for depict two states of the system. And event_function.m is for stop the ode45 when the spring touch the ground; while event_function2.m is for stop the ode45 when the spring is about to leave the ground. (Here I set a small tolerance value 0.01 between transitions between states to avoid stucking)

The screenshot of my animation is as follows and the curve represents the trace of mass center. And the red part is stance state while the blue part is flying state.

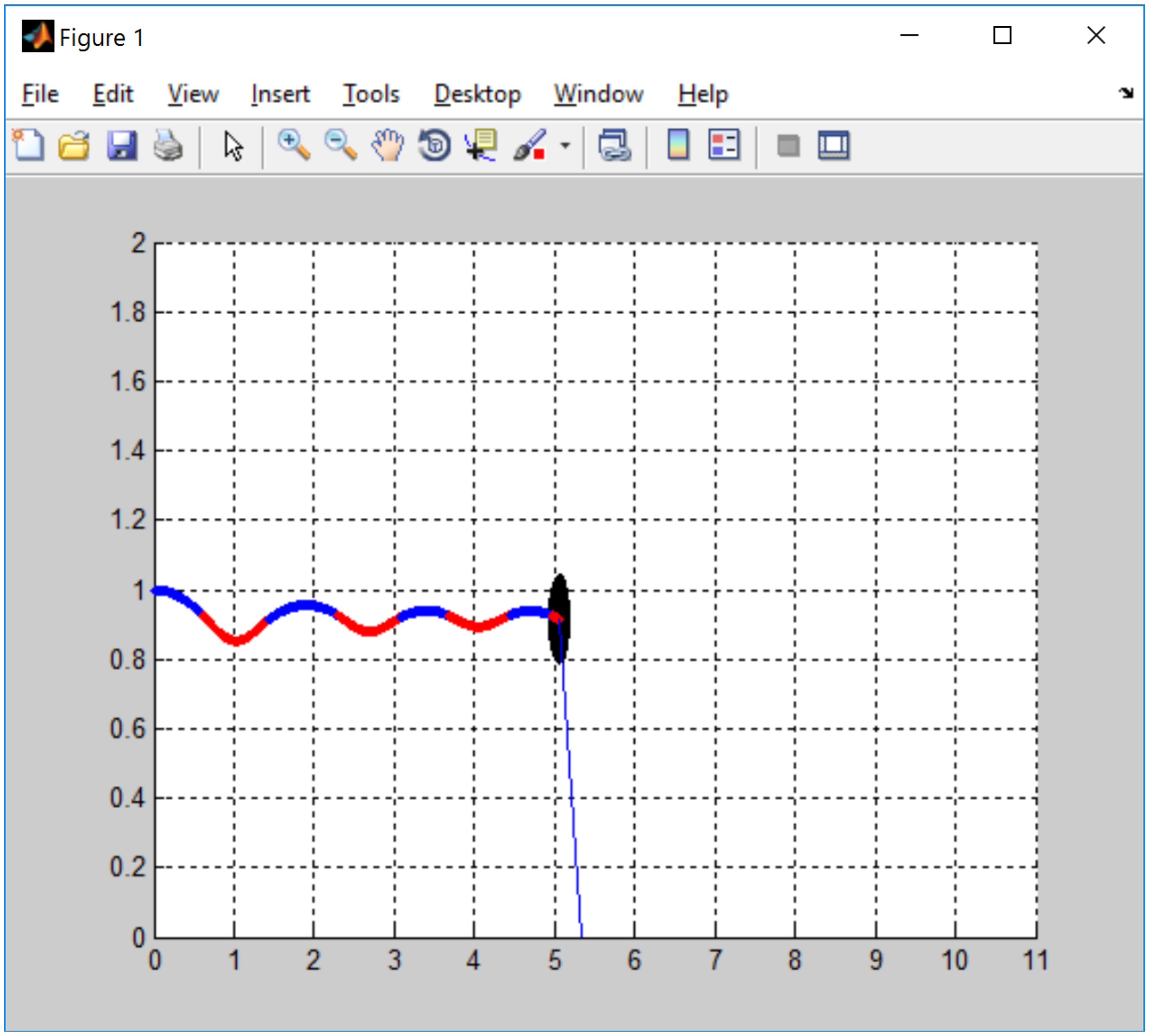


Figure 10. Animation of my model

Q2.2

Based on 2.1, we use the required initial conditions to generate next peak.

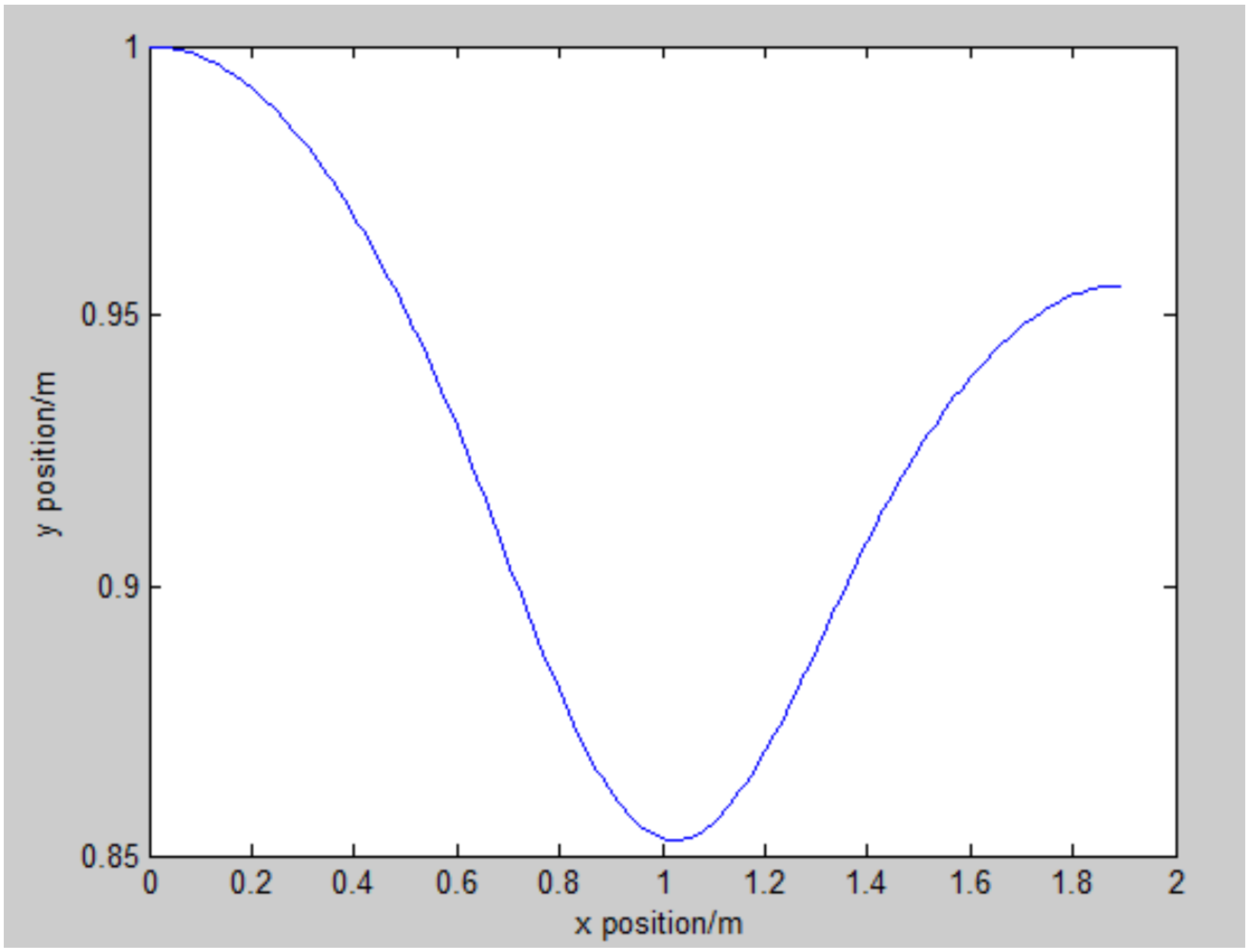


Figure 11. x-y position

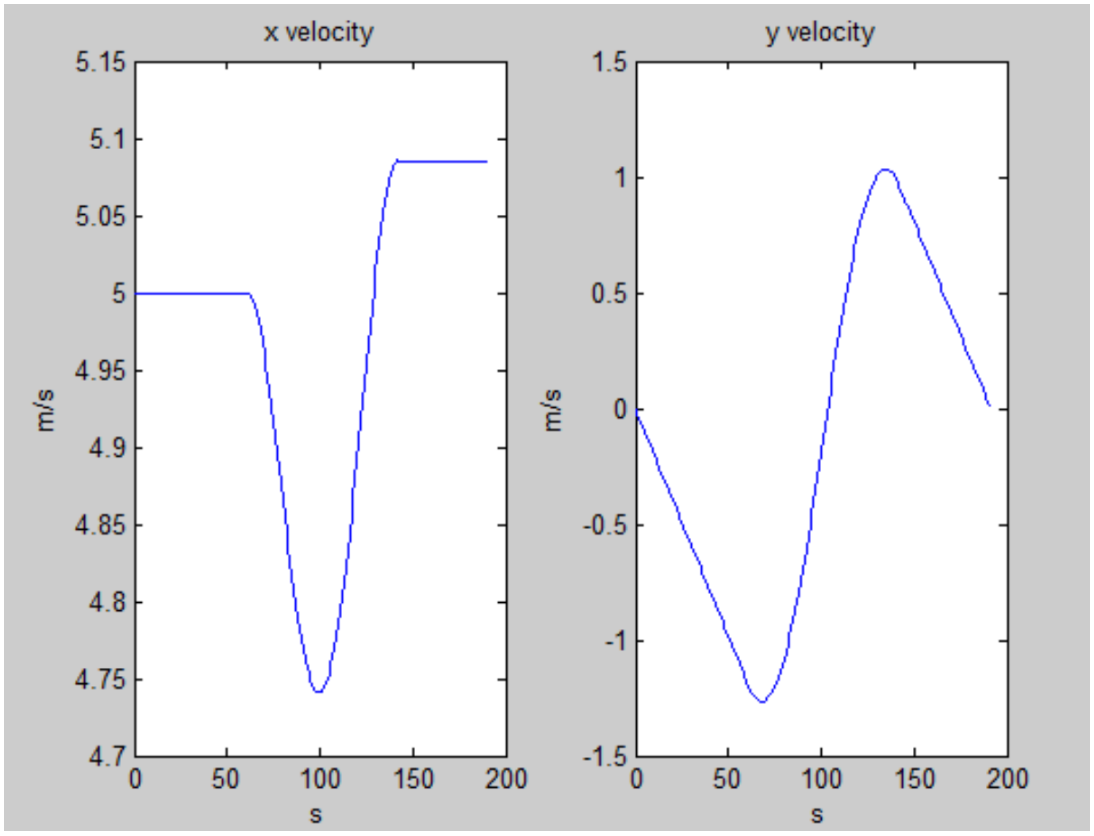
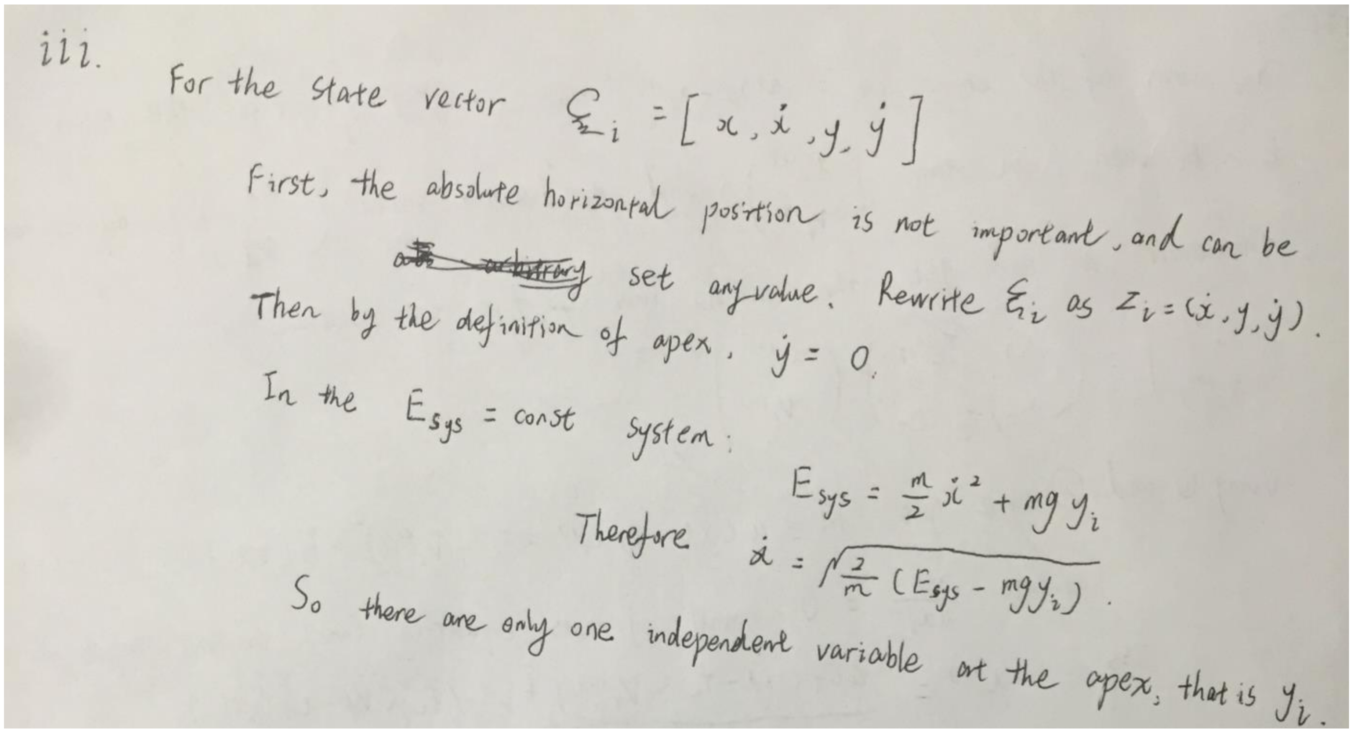


Figure 12. x-y velocity

From the two figures above, we know that the vertical position of apex i+1 is around 0.956, and the horizontal velocity of apex i+1 is 5.0845. The vertical velocity of apex i+1 is zero by definition.

Q2.3



iii.

For the state vector $\xi_i = [x, \dot{x}, y, \dot{y}]$

First, the absolute horizontal position is not important, and can be ~~arbitrary~~ set any value. Rewrite ξ_i as $z_i = (\dot{x}, y, \dot{y})$.

Then by the definition of apex, $\dot{y} = 0$.

In the $E_{sys} = \text{const}$ system:

$$E_{sys} = \frac{m}{2}\dot{x}^2 + mgy_i$$

Therefore $\dot{x} = \sqrt{\frac{2}{m}(E_{sys} - mgy_i)}$.

So there are only one independent variable at the apex, that is y_i.

Figure 13. Q_2.3 Write Up

Q2.4

In this part, we used two for-loop, the inner one is similar as part 2.1's main functionality, which can generate the trace of center of mass. And we only store the initial apex with the next apex in the inner loop. The outside for-lop is to change the initial state condition, the initial height of COM will have a feasible interval, and Vx can be inferred according to conservation of energy.

Please refer to Q2_4.m and Q2_4_plot.m for detailed implementation. And the Poincare map under three different angles (60 degree, 68 degree, 75 degree) is as below:

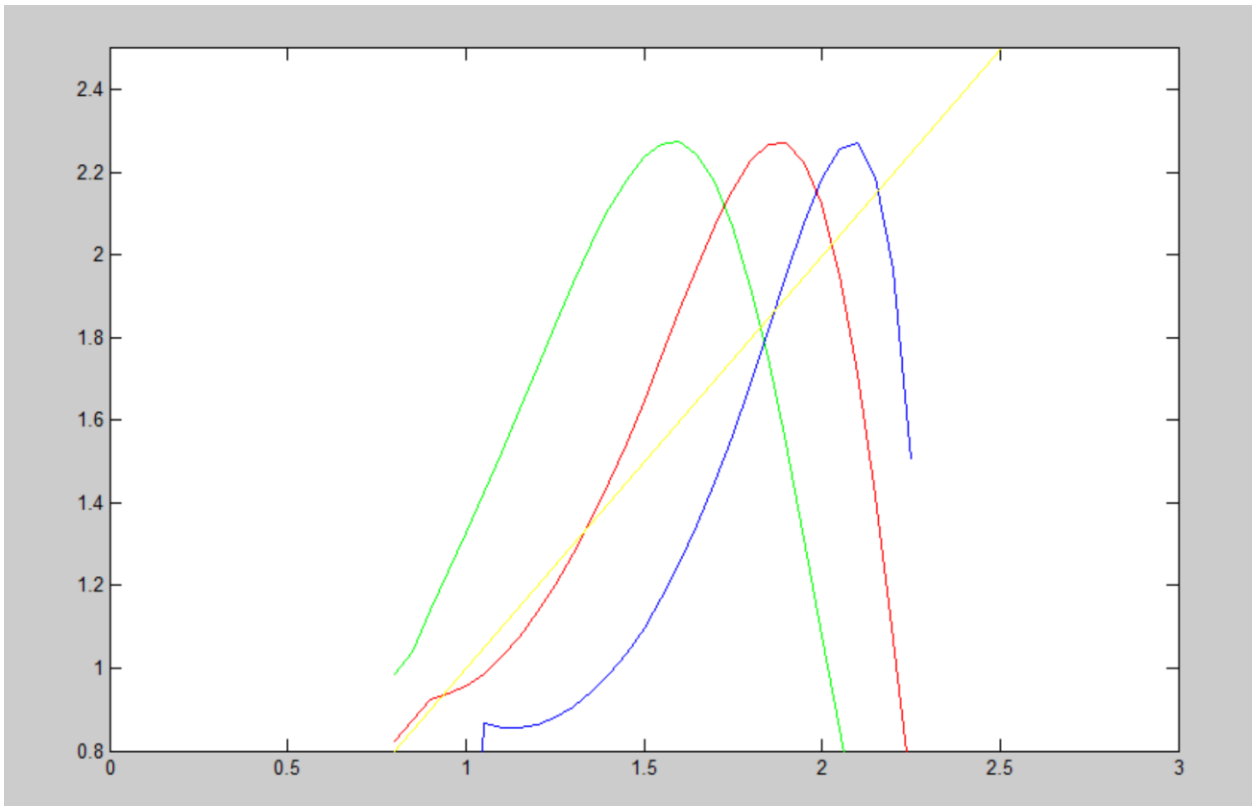


Figure 14. Poincare map

The green curve represents 60 degree, the red one represents 68 degree and the blue one represent 75 degree. The yellow straight line is the y(i)=y(i+1)

We can see 60 degree: there are only one intersection, and the slope at that intersection point is beyong -1 to 1, therefore there is one periodic solution but not stable. This might be robot jumping back and forth.

We can see 68 degree: there are three intersections, so there are three periodic solutions but only the lowest one is stable. The lowest one is moving forward, the middle one is also moving while the highest one is jumping back and forth.

We can see 75 degree : there are only two intersections in my simulation because the initial height is too low and the Vx is big, therefore there will be no apex under that condition. The two periodic solutions are both unstable. The higher one is jumping back and forth and the lower one is moving forward.