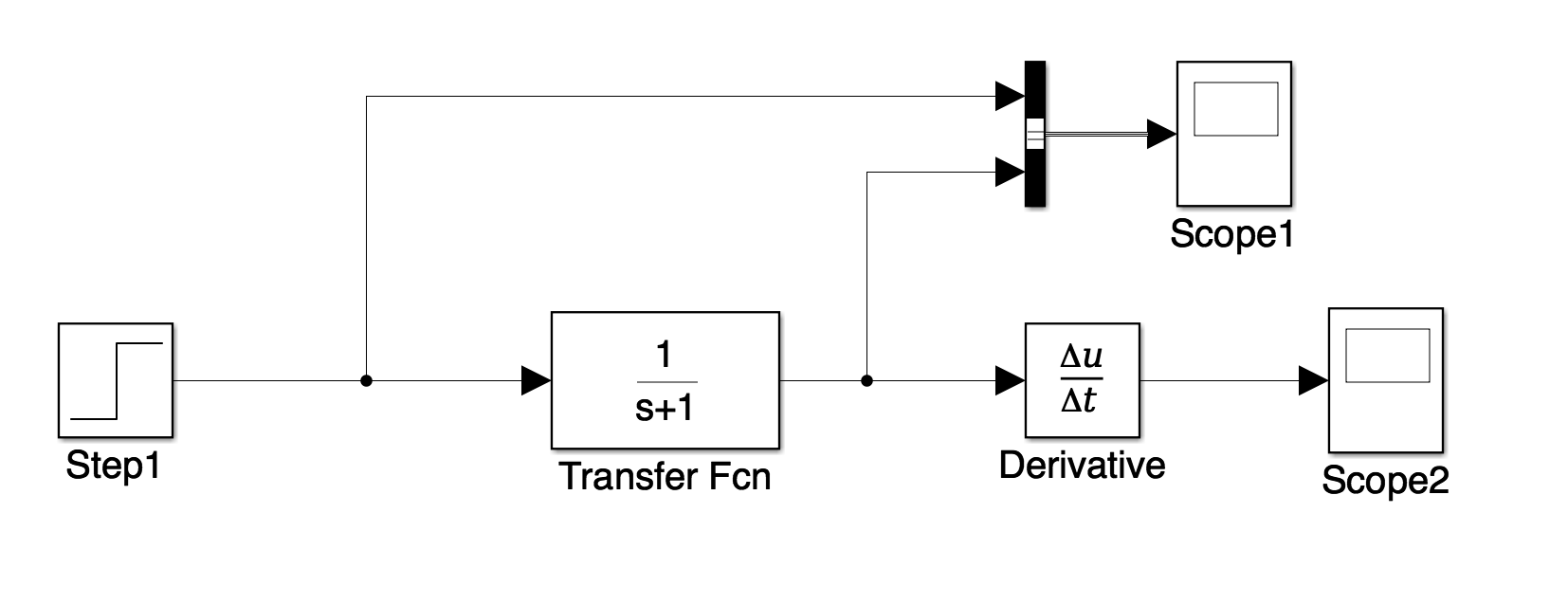
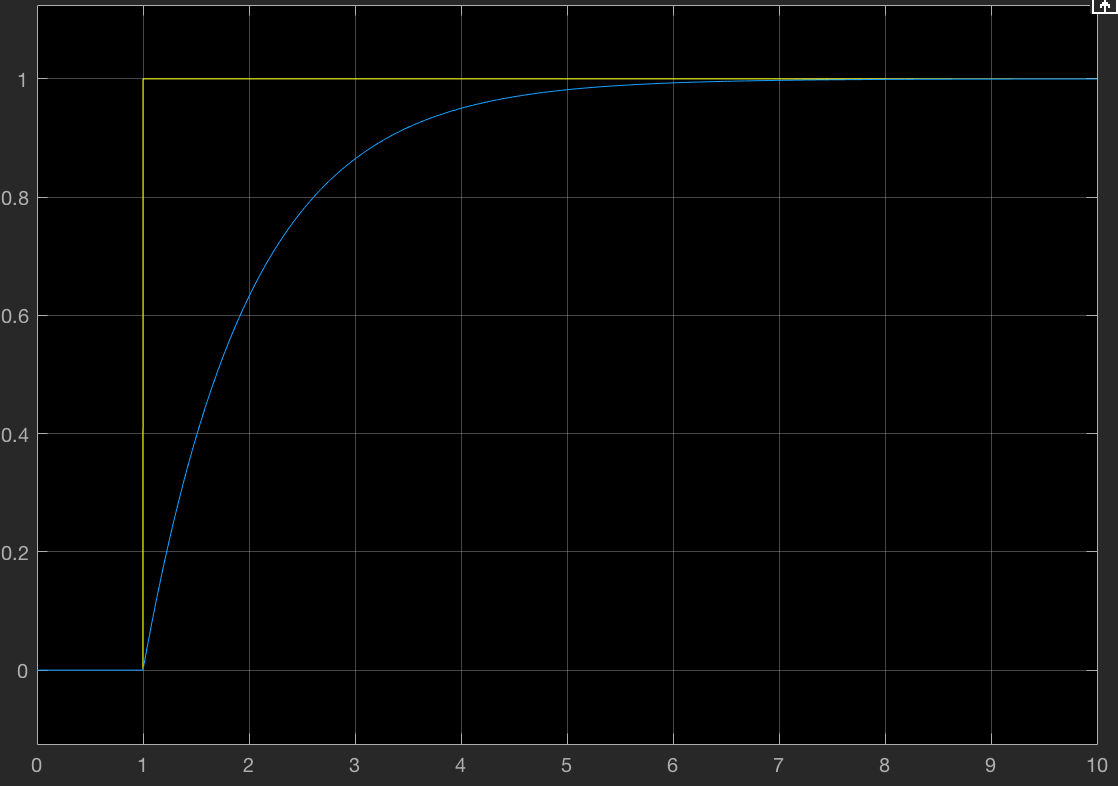
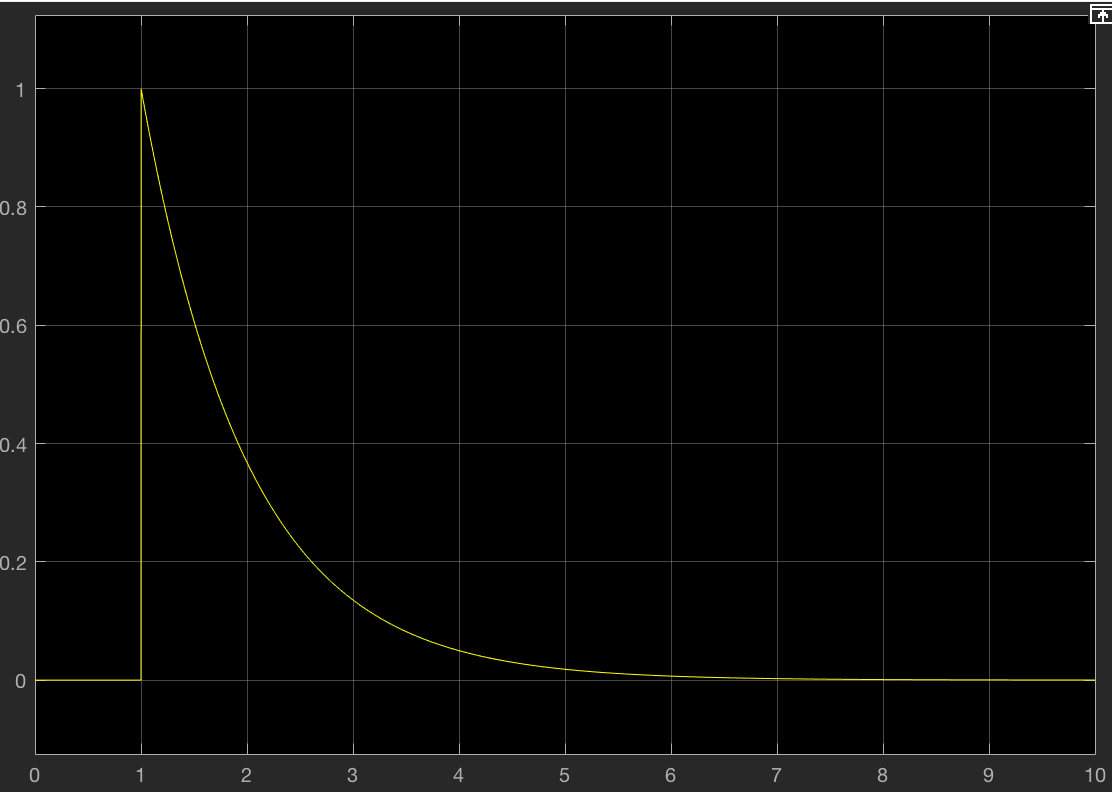
(a)





The yellow line is input,

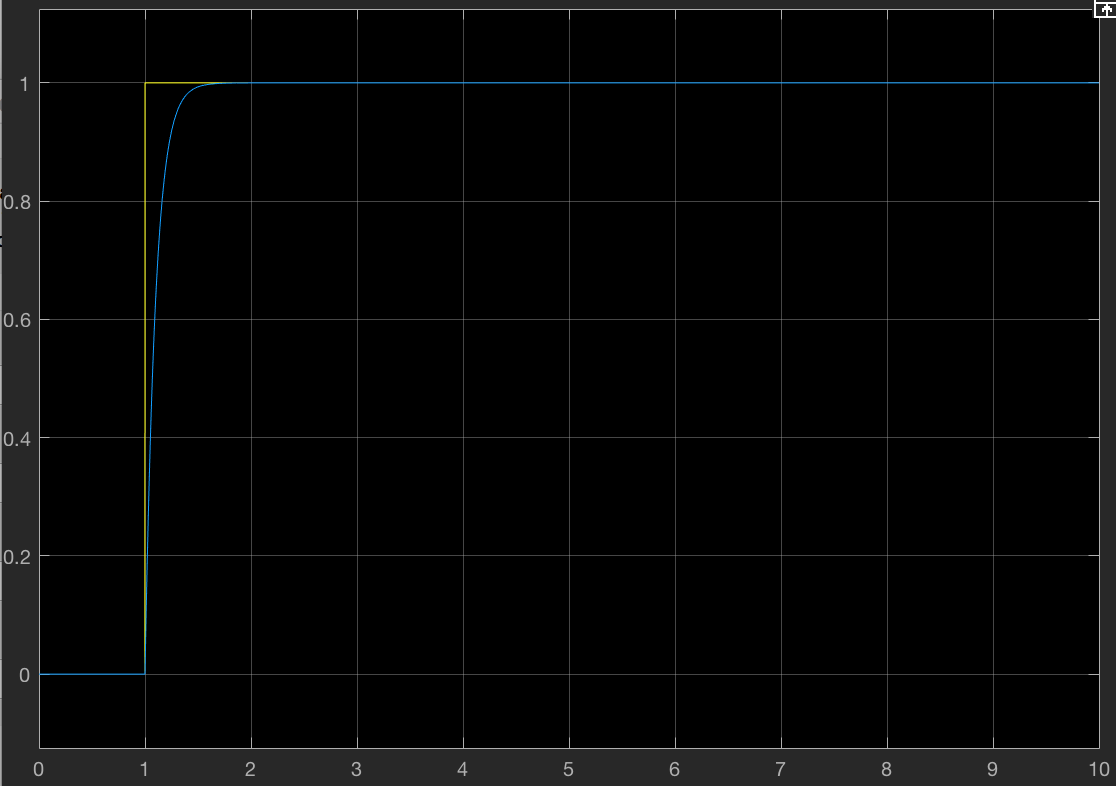
The blue line is position



Line of velocity

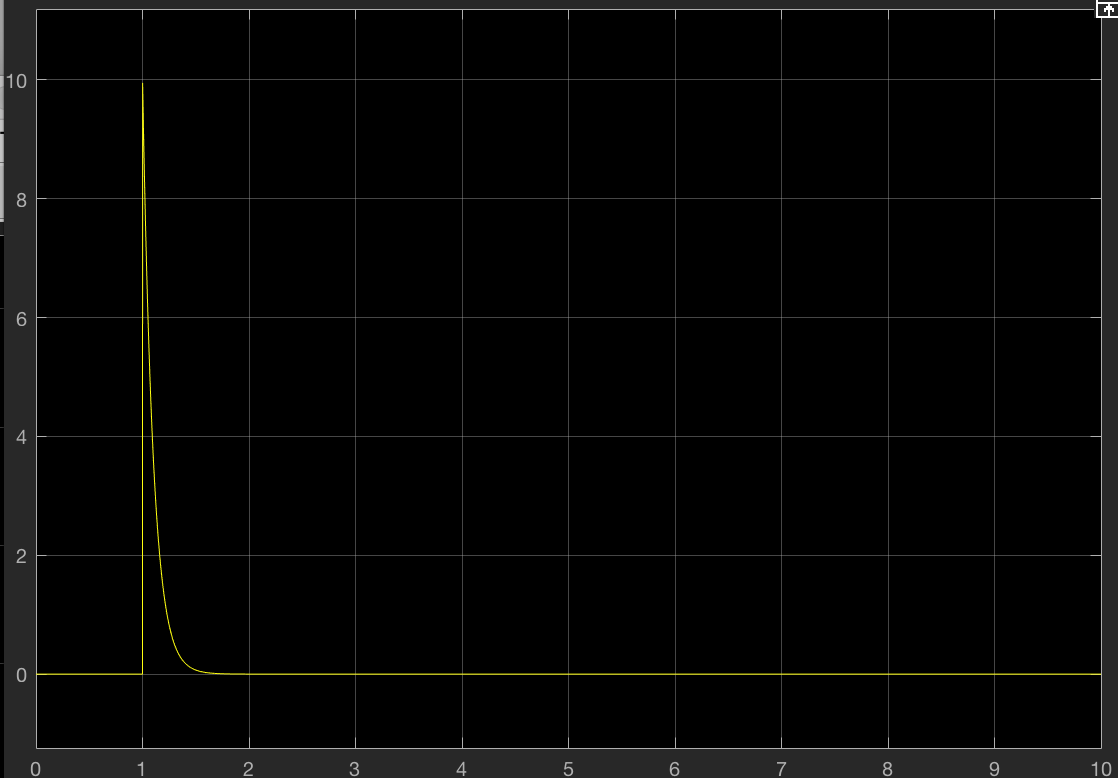
(b)

let



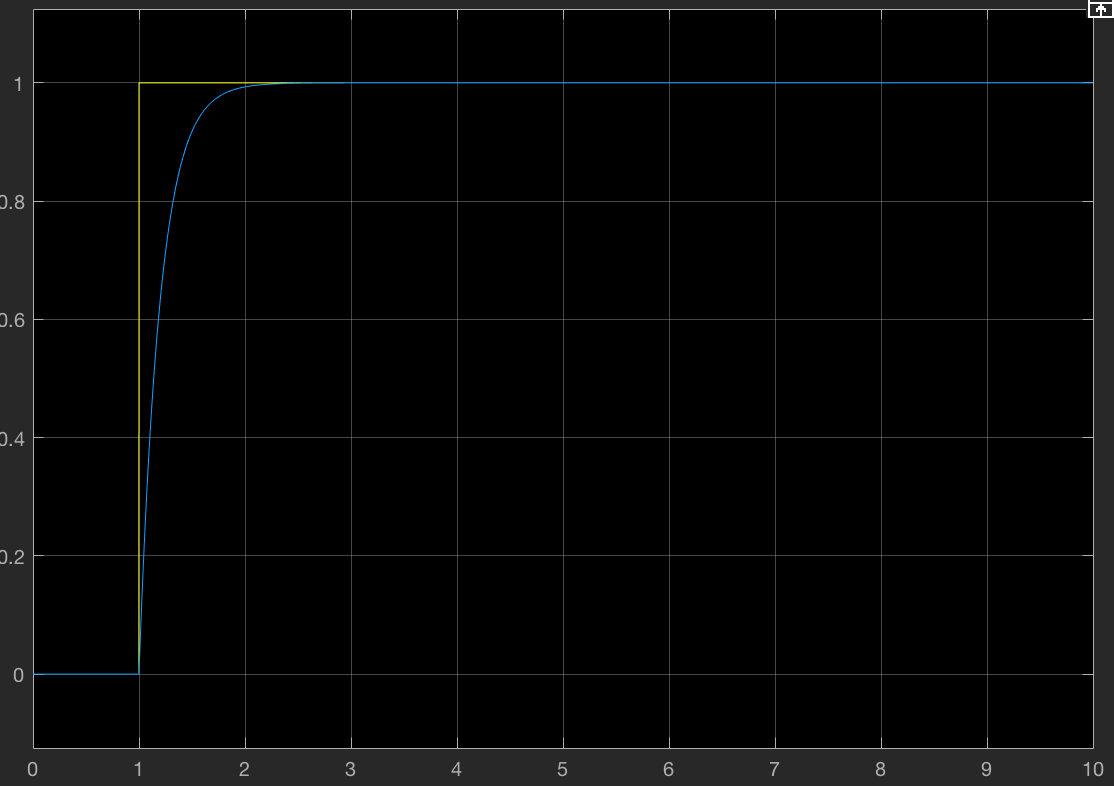
The yellow line is input,

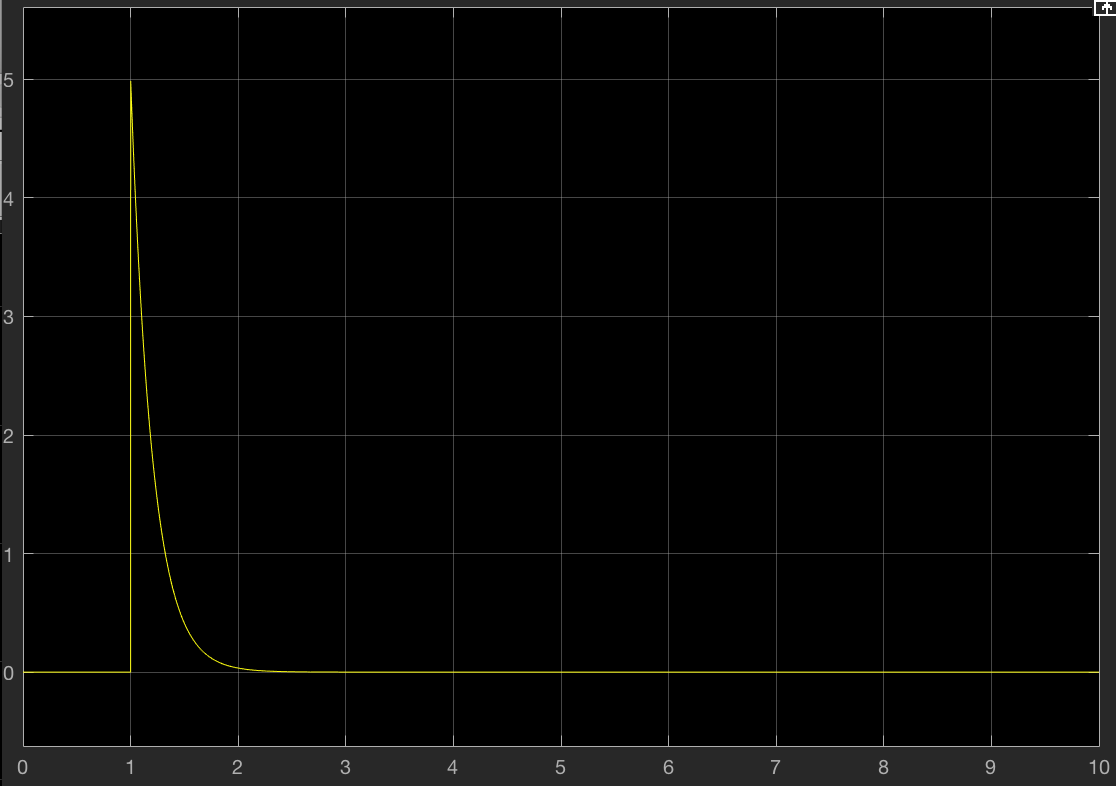
The blue line is position



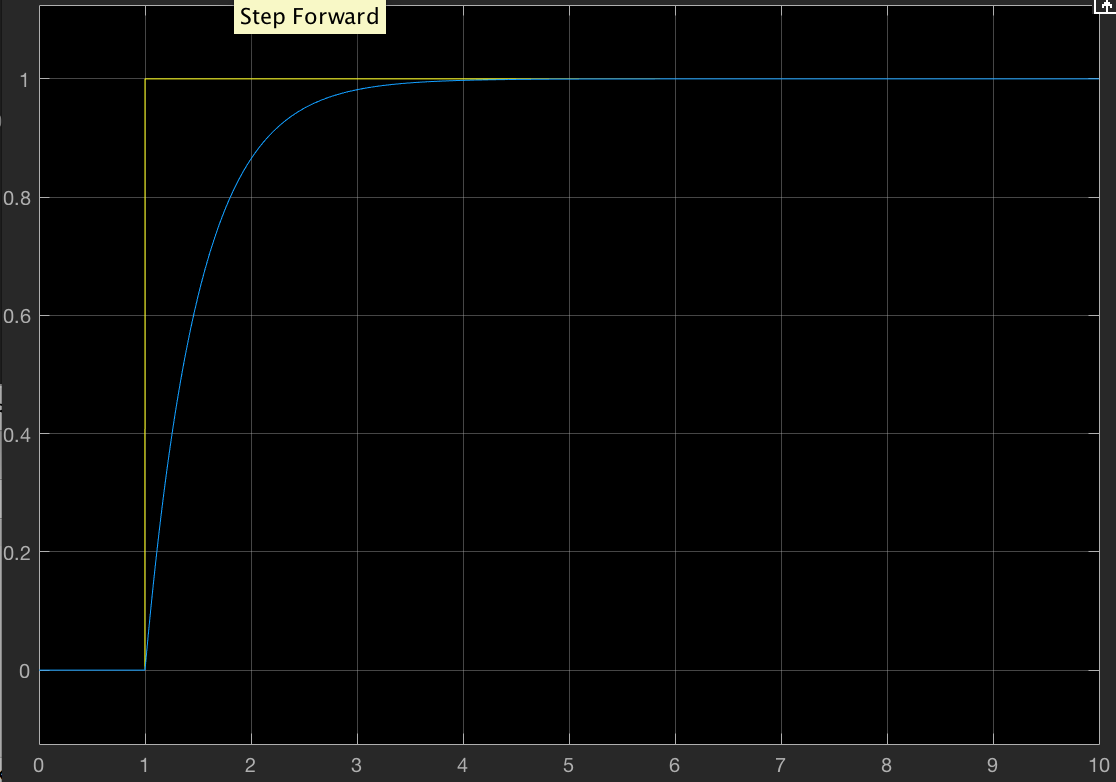
Line of velocity

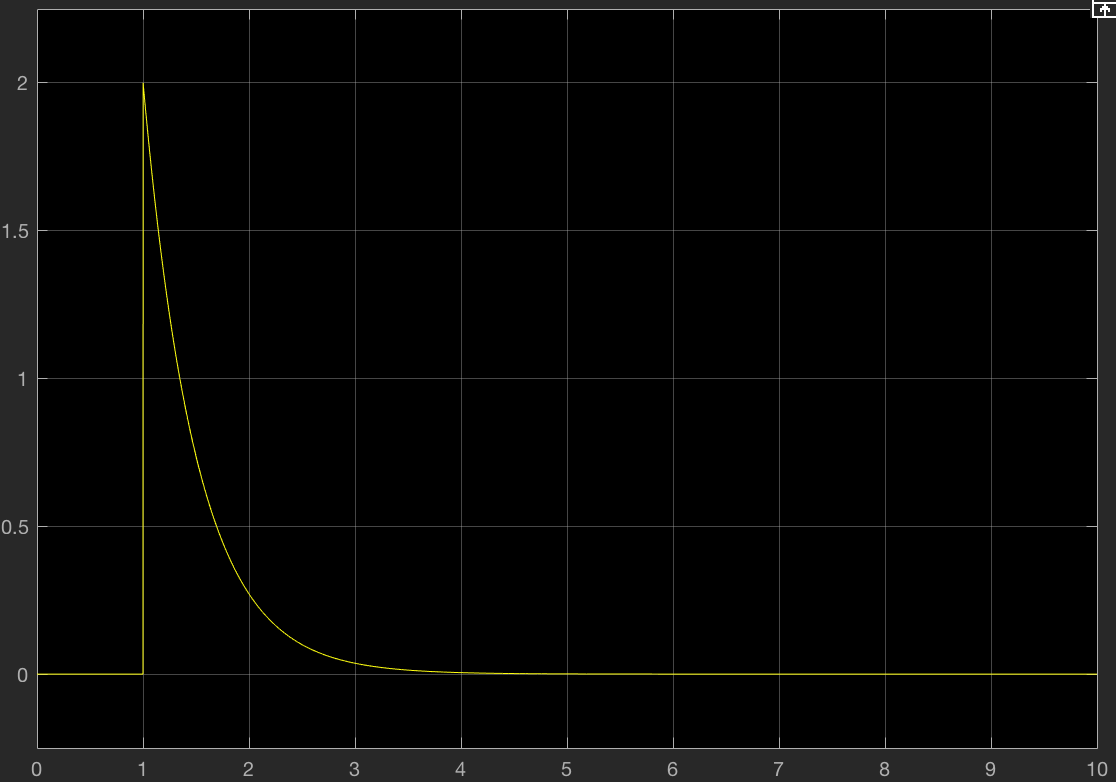
when :



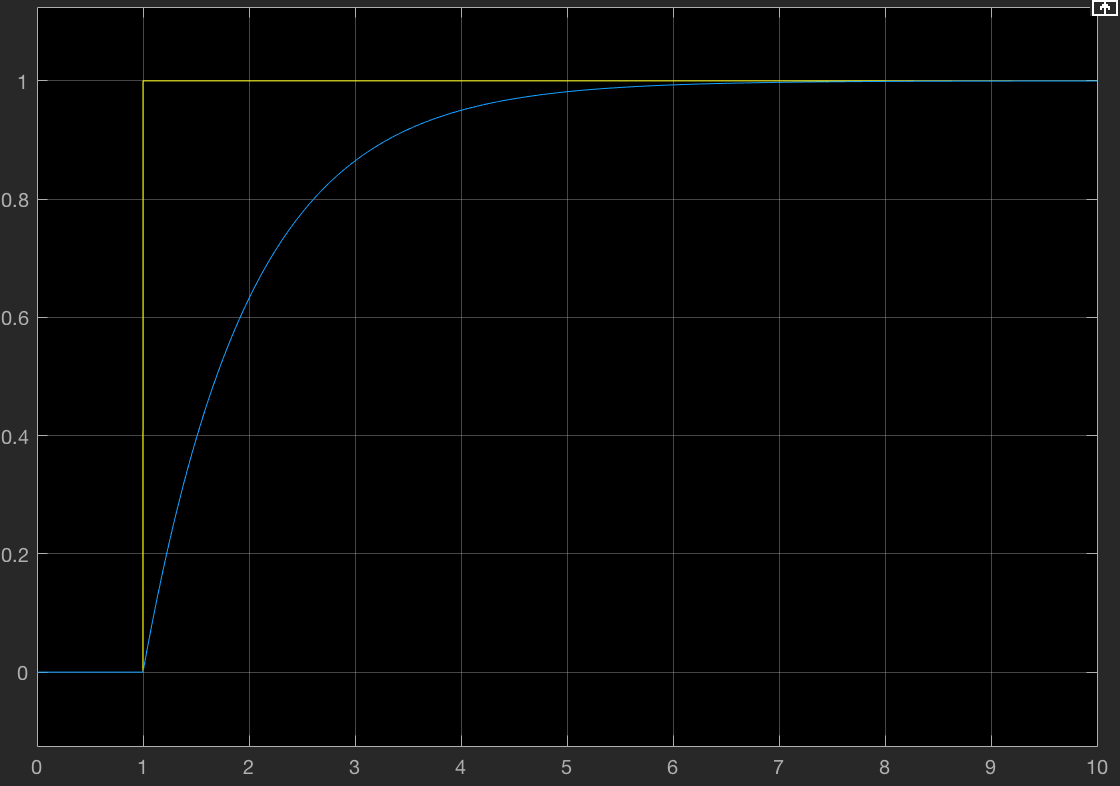


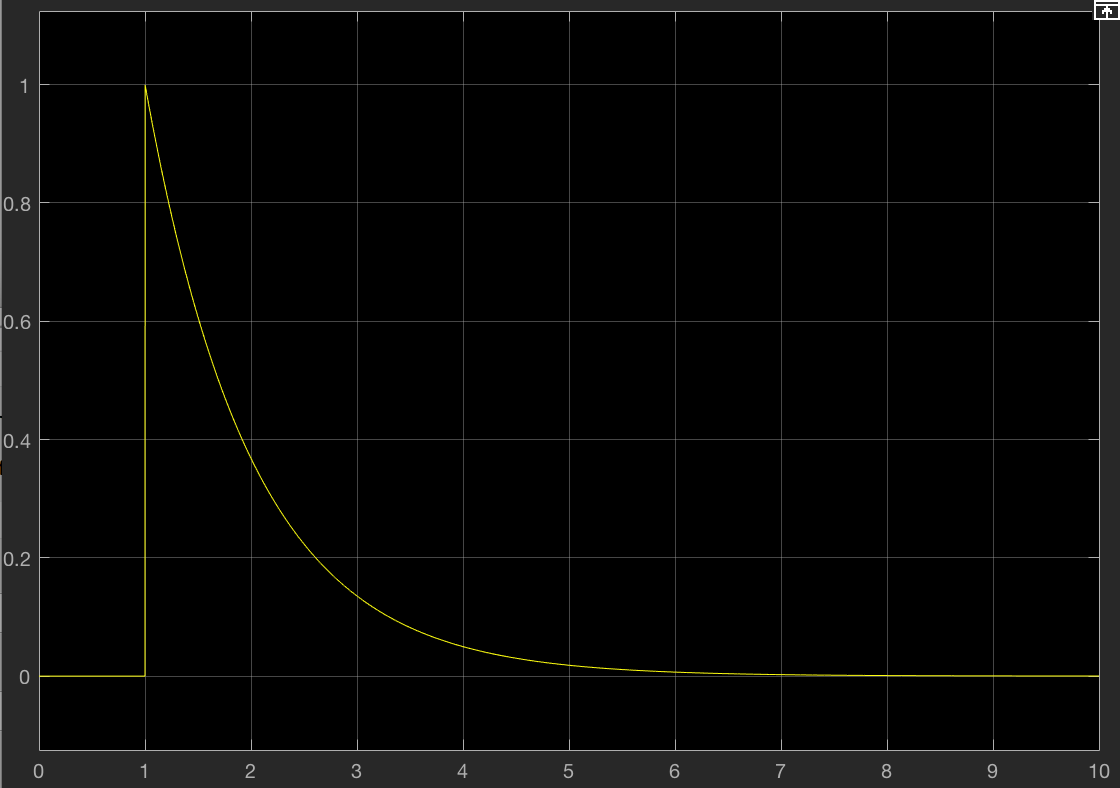
when :





when :





(c)

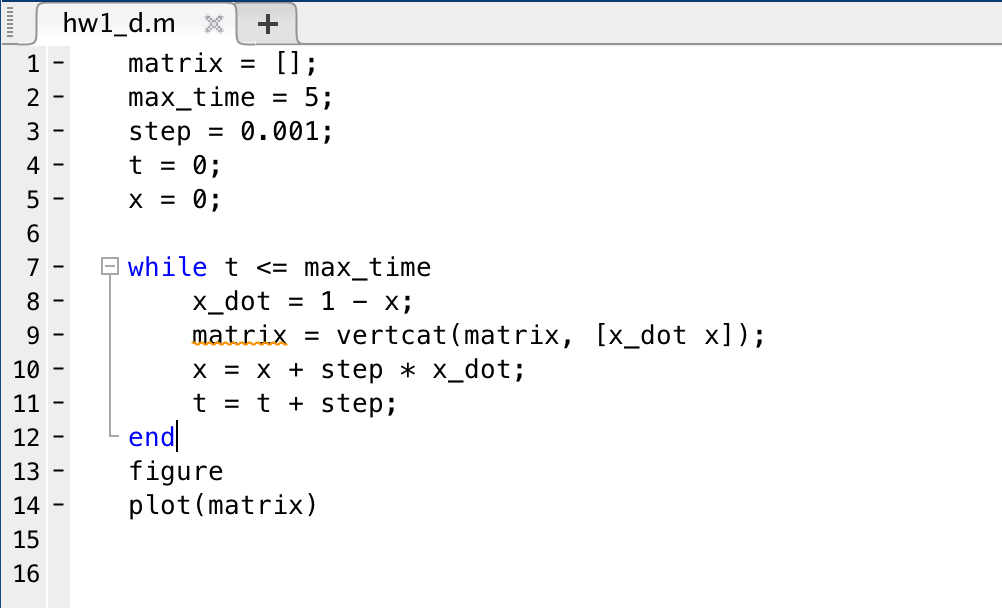
pros:

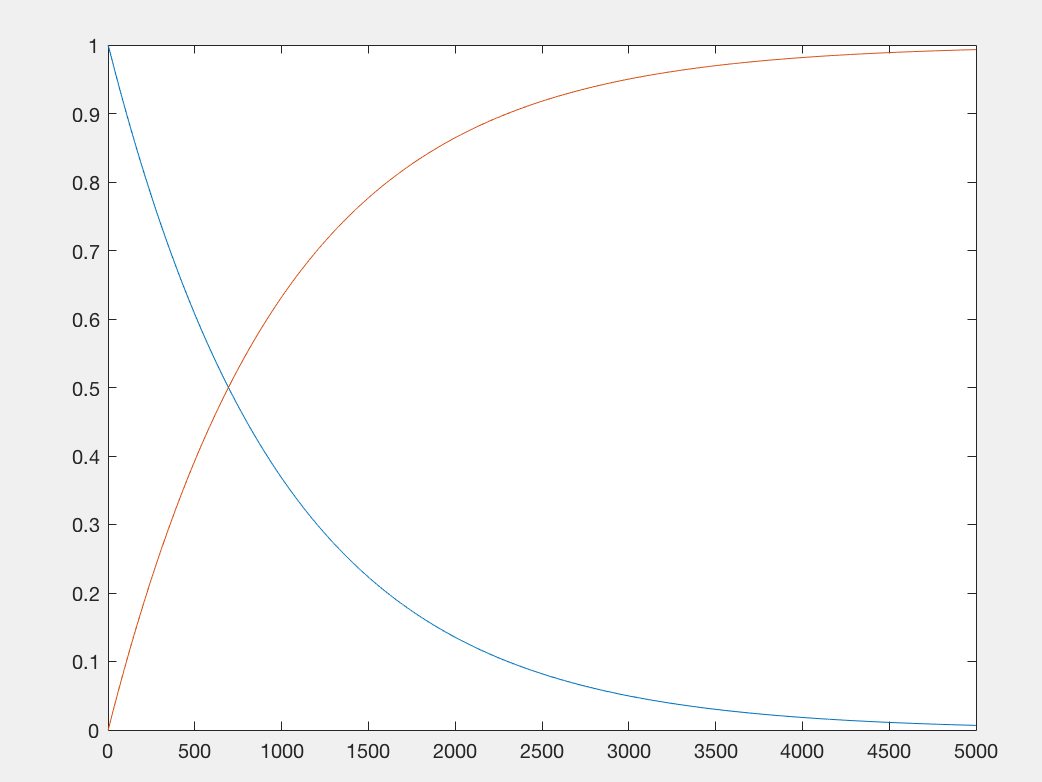
1. Easy to understand and implement
2. Easy to adjust the system through changing the parameters

cons:

1. System need to get a high speed immediately at beginning, which means the acceleration is very large at the beginning.
2. Only approximately approach to the goal, cannot 100% approach to the goal, even though really close to the goal at last.

(d)



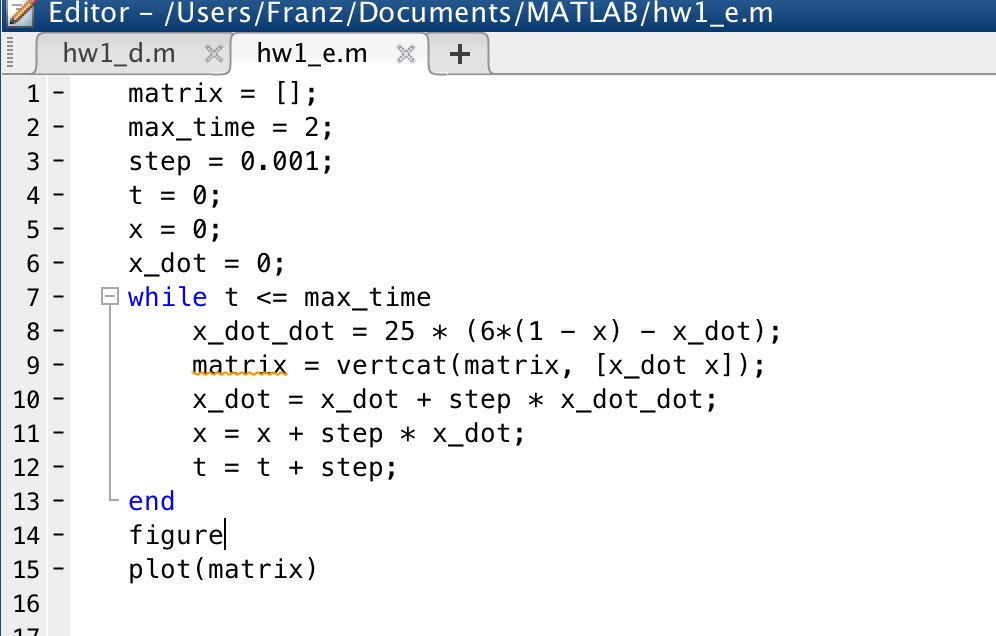


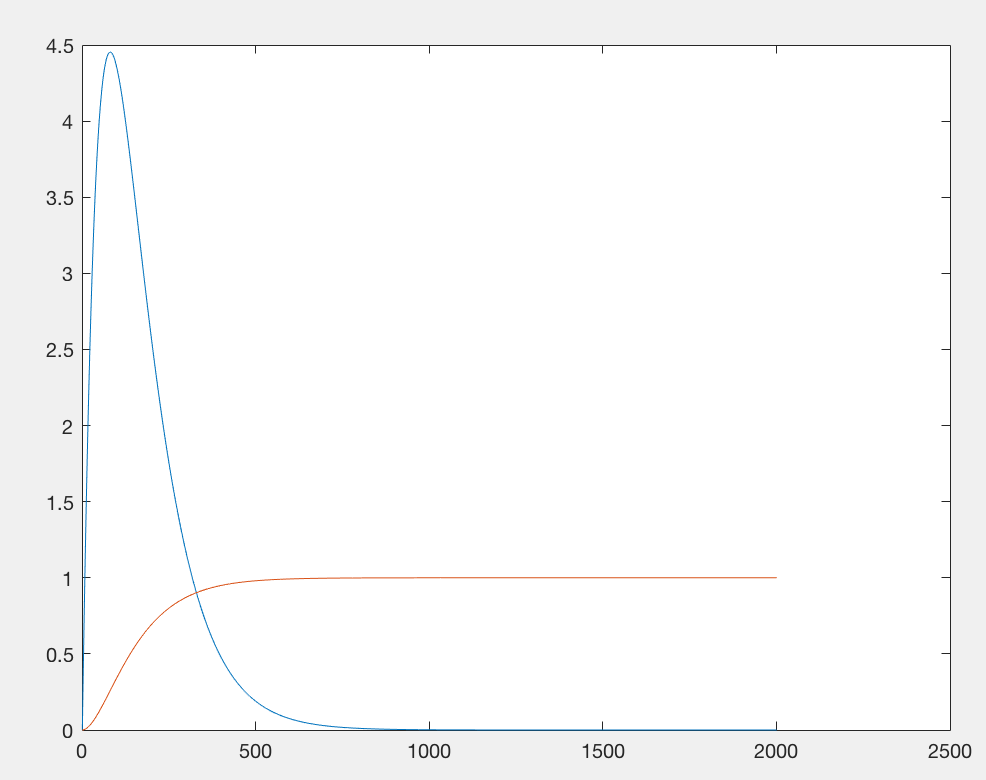
Difference:

For the position, it will move faster when using Euler method than 1a)

For the velocity, it will slow down faster when using Euler method than 1a)

(e)





difference:

1. The speed is not from 1 and then decrease, but start from 0 and then increase and then decrease
2. The speed approach close to 0 faster than before
3. The position approach to destination faster than before

Pros:

1. The speed is not from 1, so it is more practical in real life
2. It is faster to approach equilibrium state

Cons:

1. It will have a larger maximum value of speed

(f)

(g)

global xf xf\_dot xf\_dot\_dot

xf = 1;

xf\_dot = 0;

xf\_dot\_dot = 0;

matrix = [];

max\_time = 1;

step = 0.001;

t\_togo = max\_time;

x = 0;

x\_dot = 0;

x\_dot\_dot = 0;

matrix = vertcat(matrix, [x\_dot x]);

while t\_togo >= 0

    x = x + step \* x\_dot;

    x\_dot = x\_dot + step \* x\_dot\_dot;

    x\_dot\_dot\_dot = getX\_dot\_dot\_dot(x, x\_dot, x\_dot\_dot, t\_togo)

    x\_dot\_dot = x\_dot\_dot + step \* x\_dot\_dot\_dot;

    matrix = vertcat(matrix, [x\_dot x]);

    t\_togo = t\_togo - step;

end

figure

plot(matrix)

function res = getC0(x)

res = x;

end

function res = getC1(x\_dot)

res = x\_dot;

end

function res = getC2(x\_dot\_dot)

res = x\_dot\_dot / 2.0;

end

function res = getC3(x, x\_dot, x\_dot\_dot, t)

res1 = 20\*getX(x, x\_dot, x\_dot\_dot, t) - 8\*getY(x, x\_dot, x\_dot\_dot, t) + getZ(x, x\_dot, x\_dot\_dot, t);

res = res1 / 2.0;

end

function res = getC4(x, x\_dot, x\_dot\_dot, t)

res1 = -15\*getX(x, x\_dot, x\_dot\_dot, t) + 7\*getY(x, x\_dot, x\_dot\_dot, t) - getZ(x, x\_dot, x\_dot\_dot, t);

res = res1 / t;

end

function res = getC5(x, x\_dot, x\_dot\_dot, t)

res1 = 12\*getX(x, x\_dot, x\_dot\_dot, t) - 6\*getY(x, x\_dot, x\_dot\_dot, t) + getZ(x, x\_dot, x\_dot\_dot, t);

res2 = 2\*t\*t;

res = res1 / res2;

end

function res = getX(x, x\_dot, x\_dot\_dot, t)

global xf

res1 = 2\*xf - 2\*x - 2\*t\*x\_dot - x\_dot\_dot\*t\*t;

res2 = 2.0 \* t\*t\*t;

res = res1 / res2;

end

function res = getY(x, x\_dot, x\_dot\_dot, t)

global xf\_dot

res1 = xf\_dot - x\_dot - t\*x\_dot\_dot;

res2 = t\*t;

res = res1 / res2;

end

function res = getZ(x, x\_dot, x\_dot\_dot, t)

global xf\_dot\_dot

res1 = xf\_dot\_dot - x\_dot\_dot;

res2 = t;

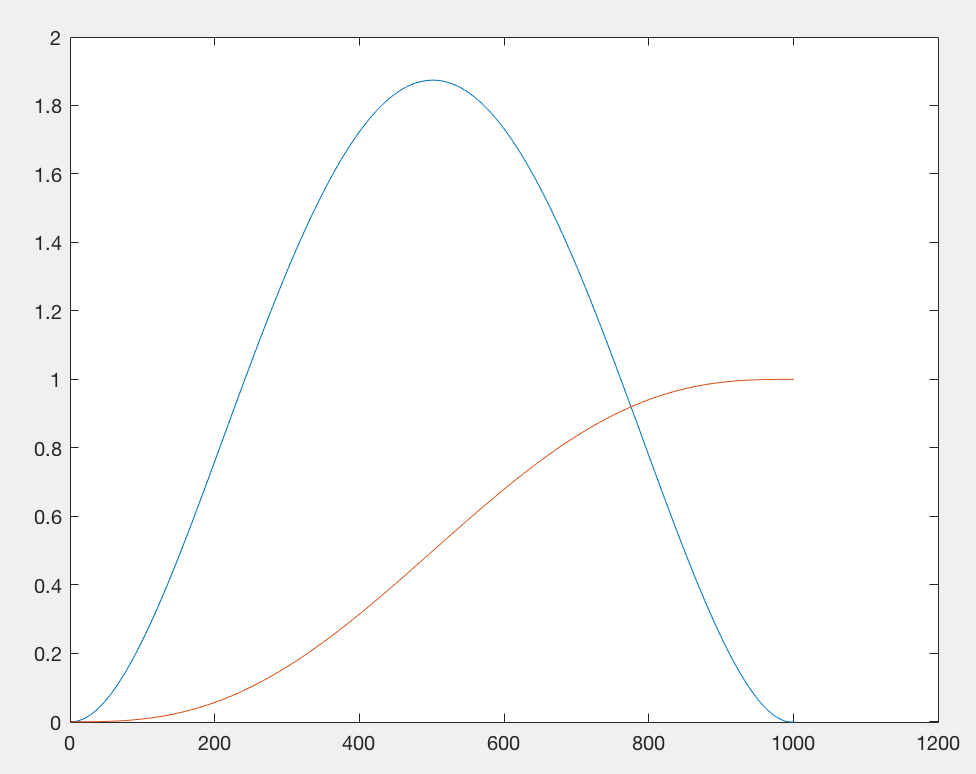
res = res1 / res2;

end

function res = getX\_dot\_dot\_dot(x, x\_dot, x\_dot\_dot, t)

res = 6\*getC3(x, x\_dot, x\_dot\_dot, t) ;

end



Similarity:

1. The lines of speed in both graphs are increase from 0 and then decrease to 0 smoothly
2. The lines of position in both graphs are increase smoothly and reach 1 at 1s.

Differences:

1. In this question, speed increase slower
2. In this question, the maximum number of speed is smaller
3. In this question, the line of speed is basically symmetric
4. In this question, the position increase slower

(h)