1. **Linearization**

Code:

|  |
| --- |
| %% Linearization  syms x xdot y ydot z zdot phi theta psi p q r  syms u1 u2 u3 u4  syms m g I\_x I\_y I\_z k\_x k\_y k\_z k\_p k\_q k\_r  syms z\_d psi\_d  states = [x y z xdot ydot zdot phi theta psi p q r];  inputs = [u1 u2 u3 u4];  equilibrium = [0 0 z\_d 0 0 0 0 0 psi\_d 0 0 0 m\*g 0 0 0];  xddot = (1/m)\*((cos(phi)\*sin(theta)\*cos(psi) + sin(phi)\*sin(psi))\*u1 - k\_x\*xdot);  yddot = (1/m)\*((cos(phi)\*sin(theta)\*sin(psi) - sin(phi)\*cos(psi))\*u1 - k\_y\*ydot);  zddot = (1/m)\*(cos(phi)\*cos(theta)\*u1 - m\*g - k\_z\*zdot);  phi\_dot = p + sin(phi)\*tan(theta)\*q + cos(phi)\*tan(theta)\*r;  theta\_dot = cos(phi)\*q - sin(phi)\*r;  psi\_dot = (sin(phi) \* q)/cos(theta) + (cos(phi) \* r)/cos(theta);  pdot = (1/I\_x)\*((I\_y - I\_z)\*q\*r + u2 - k\_p\*p);  qdot = (1/I\_y)\*((I\_z - I\_x)\*p\*r + u3 - k\_q\*q);  rdot = (1/I\_z)\*((I\_x - I\_y)\*p\*q + u4 - k\_r\*r);  sys = [xdot, ydot, zdot, xddot, yddot, zddot, phi\_dot, theta\_dot, psi\_dot, pdot, qdot, rdot];  y = [x y z psi];  Asym = subs(jacobian(sys, states), [states inputs], equilibrium);  Bsym = subs(jacobian(sys, inputs), [states inputs], equilibrium);  Csym = subs(jacobian(y, states), [states inputs], equilibrium);  Dsym = subs(jacobian(y, inputs), [states inputs], equilibrium);  parameters = [m g I\_x I\_y I\_z k\_x k\_y k\_z k\_p k\_q k\_r z\_d psi\_d];  numerics = [0.03 9.81 1.5e-5 1.5e-5 3e-5 4.5e-3 4.5e-3 4.5e-3 4.5e-4 4.5e-4 4.5e-4 2 pi/4];  A = double(vpa(subs(Asym, parameters, numerics)));  B = double(vpa(subs(Bsym, parameters, numerics)));  C = double(vpa(subs(Csym, parameters, numerics)));  D = double(vpa(subs(Dsym, parameters, numerics)));  sys = ss(A, B, C, D);  TF = tf(sys); |

Transfer Functions:

1. **Stability**

Code:

|  |
| --- |
| %% Stability  [V,D,W] = eig(A); % V consists of the eigenvectors in its columns, D consists of the eigenvalues on its diagonals  [V\_Jordan, J] = jordan(A); % J is the Jordan form of A |

Jordan form:

]

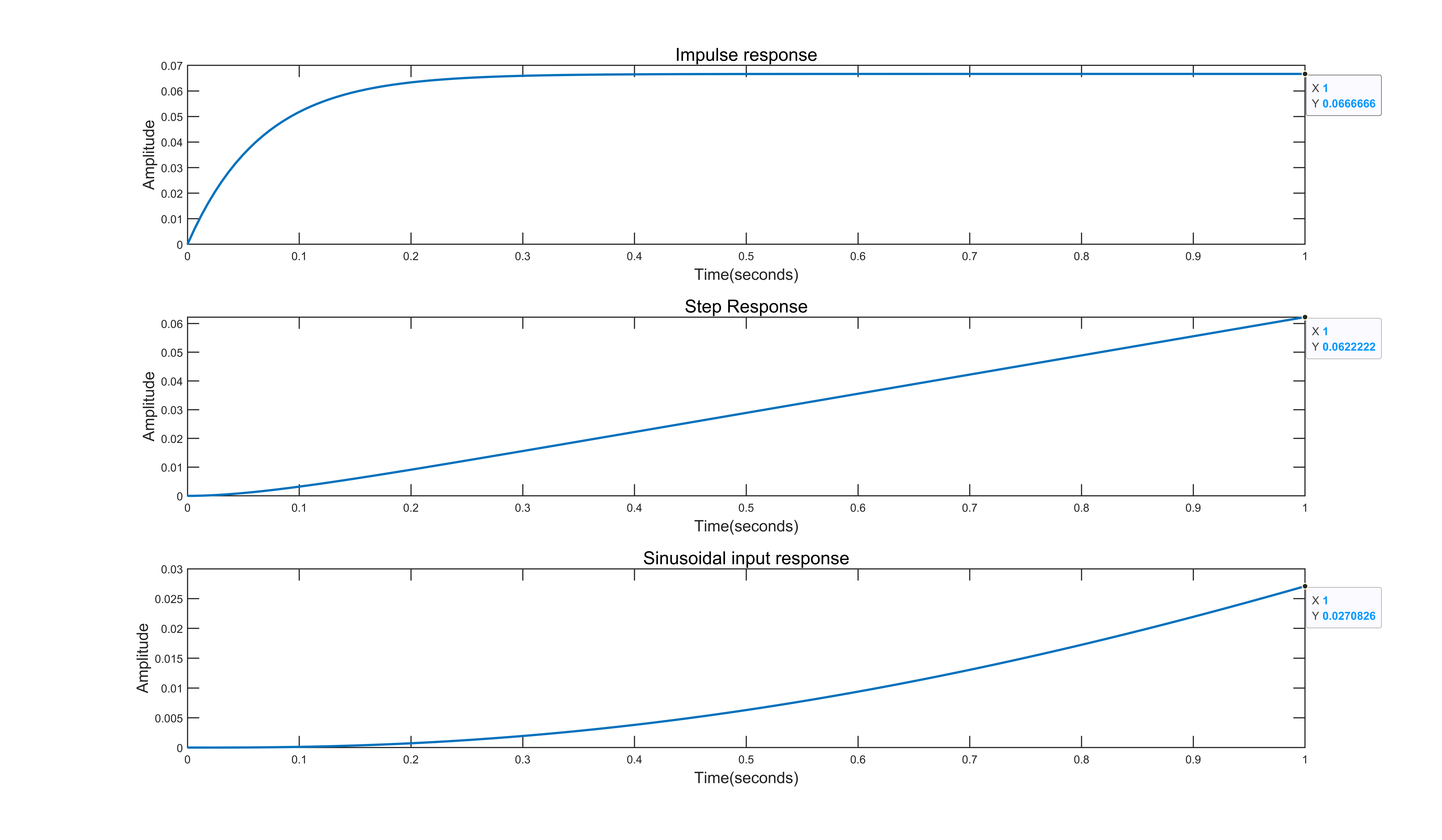
The linearized model at the equilibrium is unstable. Because two of the eigenvalues, namely 0, has a zero real part with corresponding Jordan block larger than 1\*1. They are J([1 2],[1 2]) = [0 1; 0 0] and J([5 6],[5 6]) = [0 1; 0 0], which show that the model is unstable.

1. **Forced Response**

Code:

|  |
| --- |
| %% Forced Response  A\_heading = [A(9,9) A(9,12); A(12,9) A(12,12)];  B\_heading = [B(9,4); B(12,4)];  C\_heading = [C(4,9) C(4,12)];  D\_heading = 0;  sys\_heading = ss(A\_heading, B\_heading, C\_heading, D\_heading);  subplot(3,1,1);  [y, t] = impulse(3e-5\*sys\_heading, 1);  plot(t, y, 'LineWidth',2)  ylim([0, 0.07])  title("Impulse response", 'FontSize',16);  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1);  subplot(3,1,2);  [y,t] = step(3e-5\*sys\_heading, 1);  plot(t, y, 'LineWidth',2)  title("Step Response", 'FontSize',16);  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1);  data\_points = 10000;  stoptime = 1;  dt = 1/data\_points;  t = (0:dt:stoptime);  u = 3e-5\*sin(t);  y = lsim(sys\_heading, u, t);  subplot(3,1,3);  plot(t,y, 'Linewidth', 2);  title("Sinusoidal input response", 'FontSize',16);  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1); |

Plots:



Verification using final value theorem:

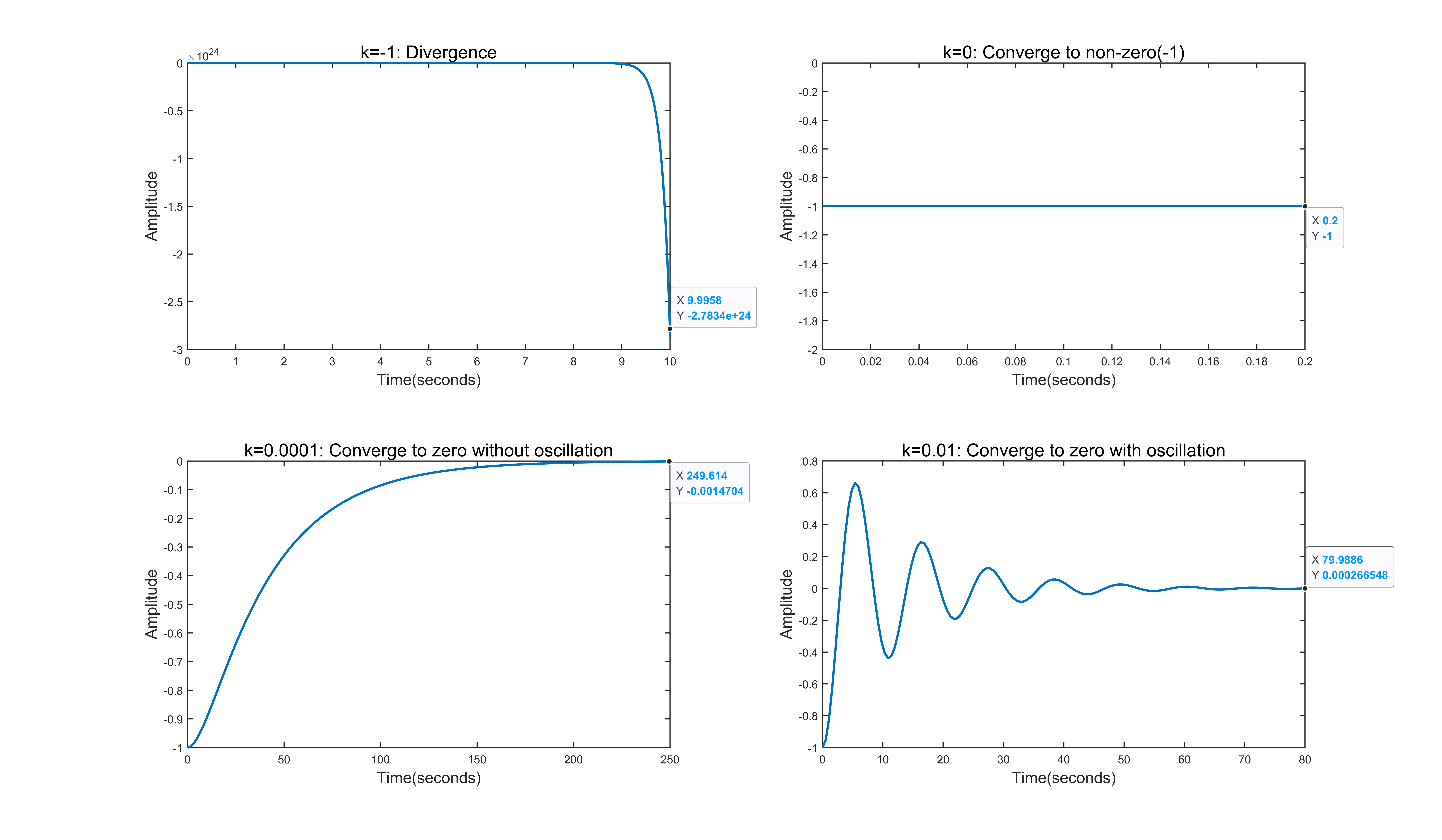
The result of final value theorem coincides with the result shown by MATLAB.

1. **Proportional Control**

Code:

|  |
| --- |
| %% Proportional Control  A\_altitude = [A(3,3) A(3,6); A(6,3) A(6,6)];  B\_altitude = [B(3,1); B(6,1)];  C\_altitude = [1 0];  D\_altitude = 0;  x0 = [-1;0];  k = -1;  A\_controlled\_altitude = A\_altitude - k\*B\_altitude\*C\_altitude;  sys\_altitude = ss(A\_controlled\_altitude, [], C\_altitude, []);  subplot(2,2,1)  [y, t] = initial(sys\_altitude, x0);  plot(t,y, 'linewidth',2)  xlim([0,10])  title("k=-1: Divergence", 'Fontsize',16)  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1);  k = 0;  A\_controlled\_altitude = A\_altitude - k\*B\_altitude\*C\_altitude;  sys\_altitude = ss(A\_controlled\_altitude, [], C\_altitude, []);  subplot(2,2,2)  [y,t]=initial(sys\_altitude, x0);  plot(t,y, 'linewidth', 2)  title("k=0: Converge to non-zero(-1)", 'Fontsize',16)  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1);  k = 0.0001;  A\_controlled\_altitude = A\_altitude - k\*B\_altitude\*C\_altitude;  sys\_altitude = ss(A\_controlled\_altitude, [], C\_altitude, []);  subplot(2,2,3)  [y,t]=initial(sys\_altitude, x0);  plot(t,y, 'linewidth', 2)  xlim([0,250])  title("k=0.0001: Converge to zero without oscillation", 'Fontsize',16)  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1);  k = 0.01;  A\_controlled\_altitude = A\_altitude - k\*B\_altitude\*C\_altitude;  sys\_altitude = ss(A\_controlled\_altitude, [], C\_altitude, []);  subplot(2,2,4)  [y,t]=initial(sys\_altitude, x0);  plot(t,y,'linewidth',2)  xlim([0,80])  title("k=0.01: Converge to zero with oscillation", 'Fontsize',16)  xlabel("Time(seconds)", 'FontSize',14)  ylabel("Amplitude", 'FontSize',14)  set(gca,'linewidth',1); |

Plots:



Case Divergence: choose k = -1

Case Converge to non-zero: choose k = 0

Case Converge to zero without oscillation: choose k = 0.0001

Case Converge to zero with oscillation: choose k = 0.01