# **State Observers**

## Notes & Screenshots

**Example 1** – **Heat System**

Observers\_Ex1\_HeatSys.m

**% 4th Order Heat System, Observer**

**A = [-2 1 0 0; 1 -2 1 0; 0 1 -2 1; 0 0 1 -1];**

**B = [1;0;0;0]; C = [0 0 0 1]; D = 0;**

**% Let's see if we can observe using just the last state**

**rank([C; C\*A; C\*A^2; C\*A^3])**

**% pause**

**% Cool. Let's find H then.**

**des\_poles = [-1 -2 -3 -4];**

**H = placePoles(A',C',C, des\_poles)**

**% pause**

**% Simulate and check that plant states and observer states converge**

**Aaug = [A, zeros(4,4); H'\*C, A-H'\*C]; Baug = [B;B];**

**Caug = eye(8); Daug = zeros(8,1);**

**X0 = [0;0;0;0; 5\*rand(4,1)];**

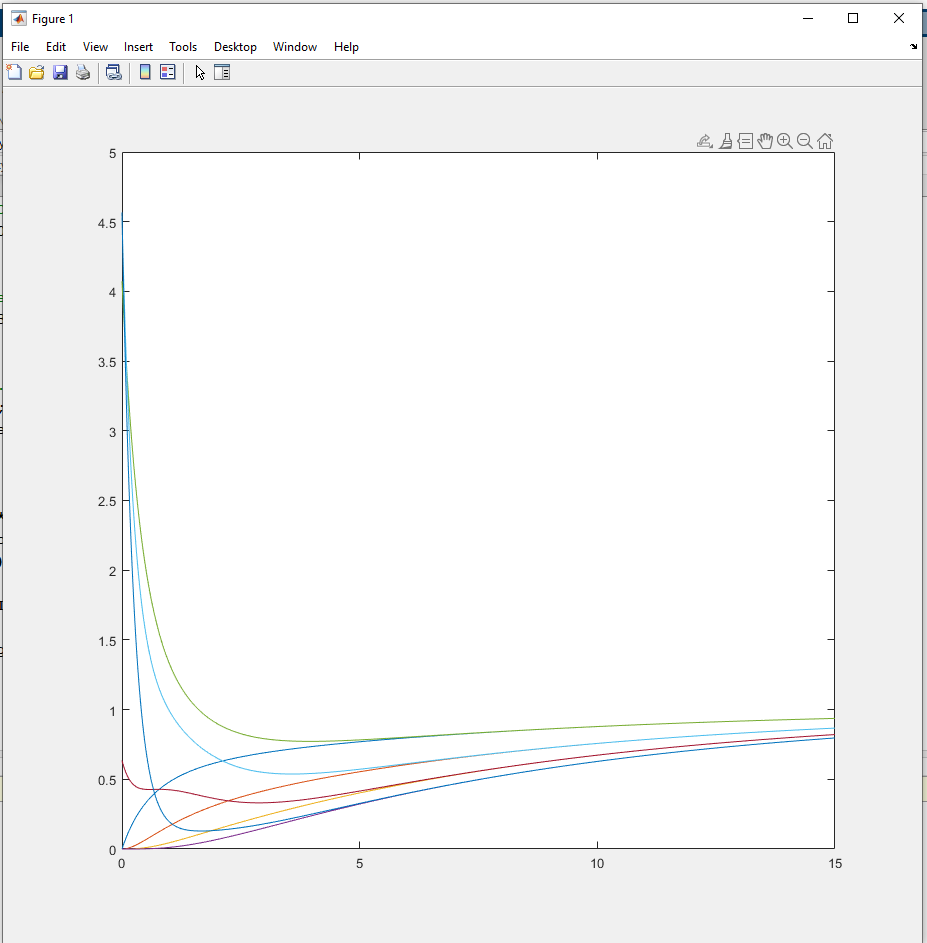
**T\_end = 15;**

**t = transpose(linspace(0,T\_end,1001));**

**R = 0\*t + 1;**

**Y = step3(Aaug, Baug, Caug, Daug, t, X0, R);**

**plot(t, Y)**



HeatObs.m

**%% Observer**

**% H = [1 2 2 3]';**

**H = transpose(placePoles(Aapprox',Capprox',Capprox,[-1,-2,-3,-4]));**

**Ae = Aapprox; Be = Bapprox; Ce = Capprox; De = D;**

**%% Simulate**

**V = zeros(20,1); Z = 0; Xe = zeros(4,1);**

**dt = 100e-6; T\_end = 20; t = 0;**

**Ref = 1;**

**N = (T\_end / dt) + 1;**

**DATA = zeros(N,9);**

**i=1;**

**tic**

**while(t < T\_end)**

**V0 = 1 + sin(2\*t);**

**dV = A\*V + B\*V0;**

**dXe = Ae\*Xe + Be\*V0 + H\*(V(20)-Ce\*Xe);**

**V = V + dV \* dt;**

**Xe = Xe + dXe \* dt;**

**t = t + dt;**

**DATA(i,:) = [V([5,10,15,20])', Xe', V0];**

**i = i+1;**

**end**

**toc**

**t = [1:length(DATA)]' \* dt;**

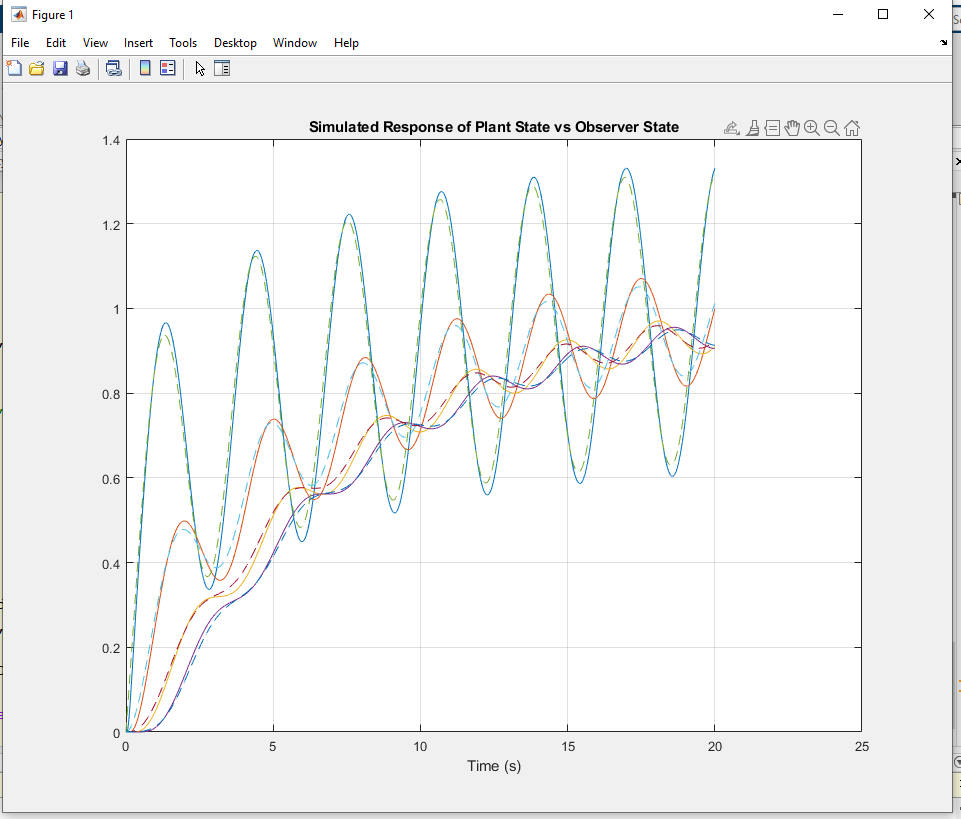
**DATAds = downsample(DATA,10);**

**tds = downsample(t,10);**

**plot(t,DATA(:,[1:4]), t,DATA(:,[5:8]), '--');**

**grid on;**

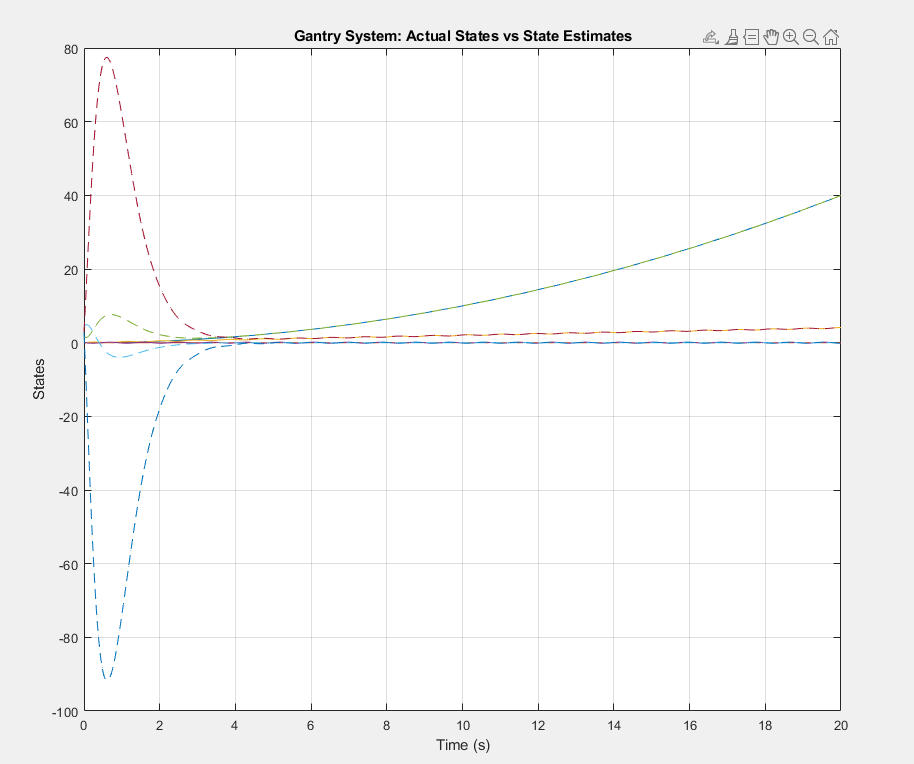
**title('Simulated Response of Plant State vs Observer State'); xlabel('Time (s)');**



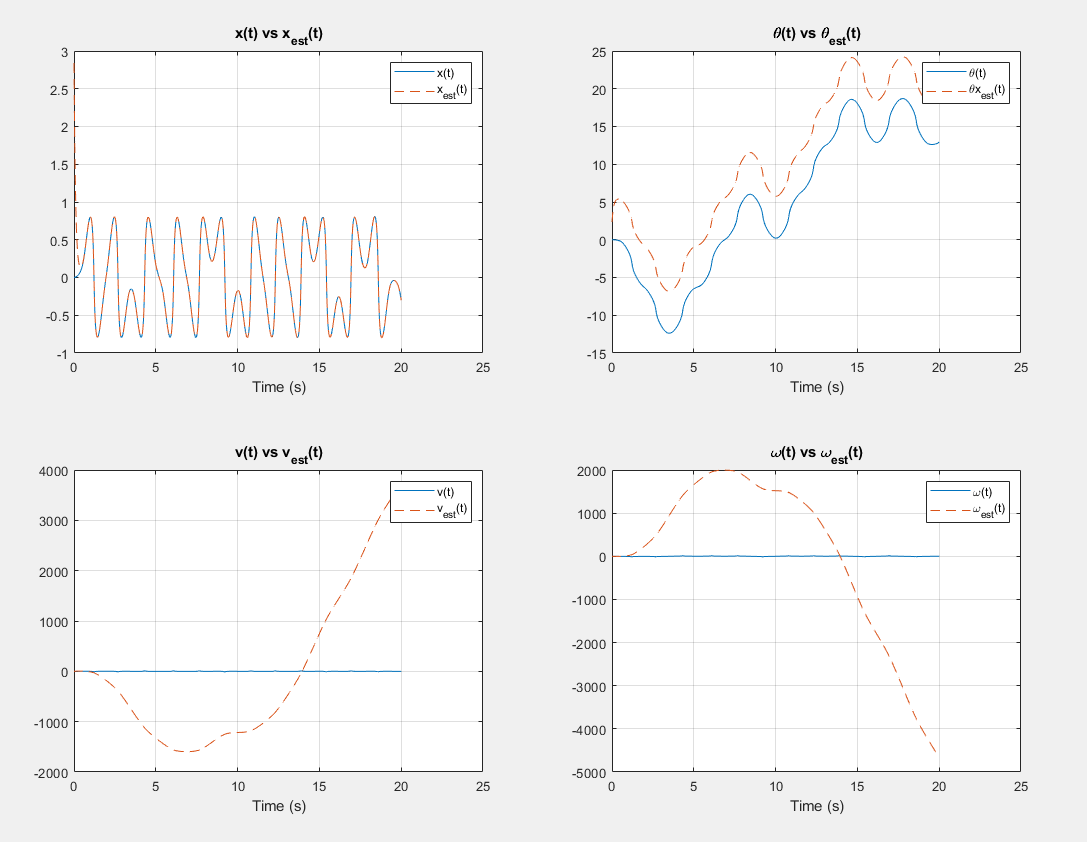
Now let’s try a gantry system /w mGC = 1kg, mL = 4kg, and L = 1m.

*I had a weird issue here…*

So with the step3 linearized simulation, I get observer convergence. Great. But…

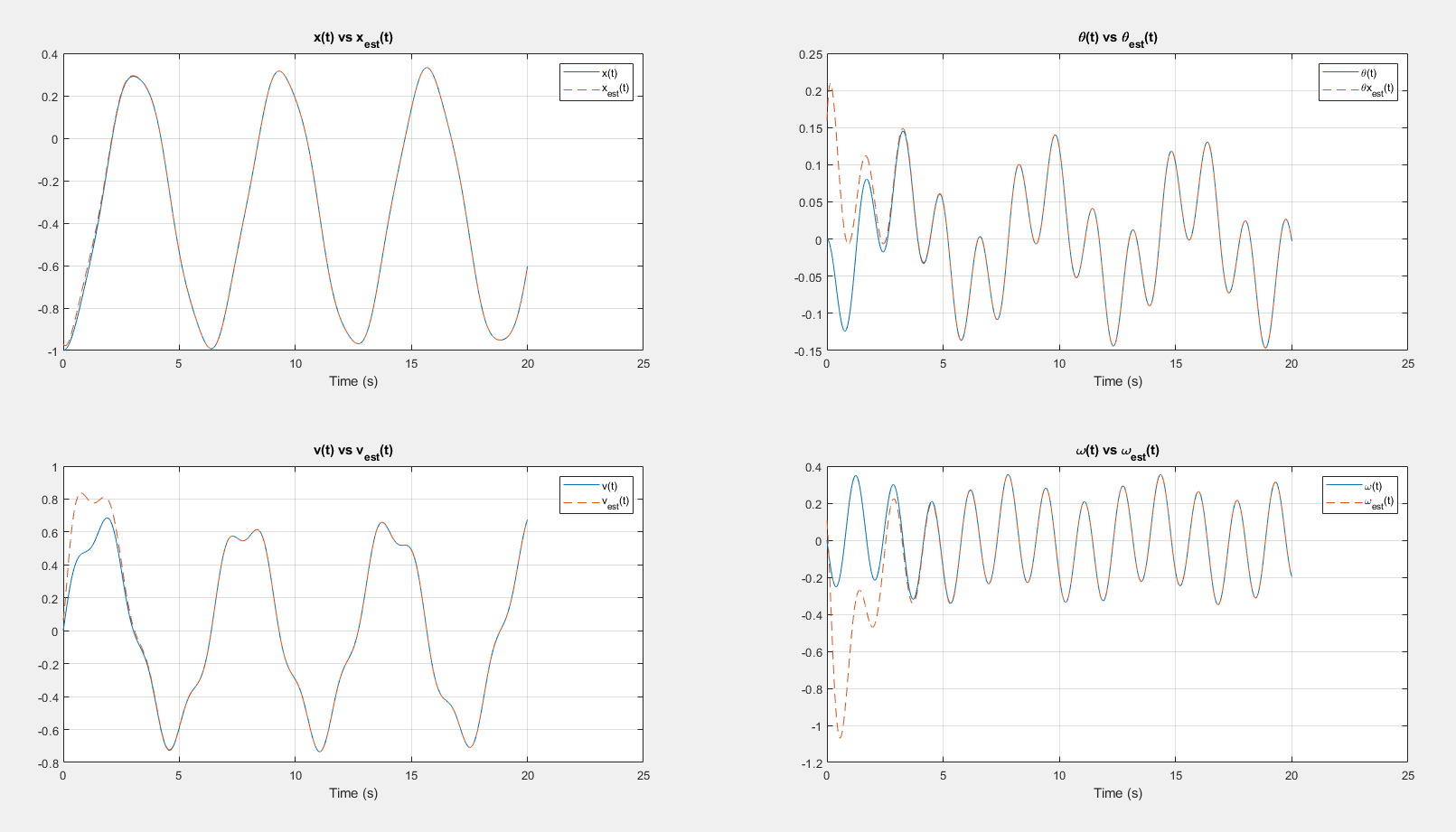


… with the nonlinear system simulation…



There is convergence with x(t), θ(t) is there but /w an offset of around 5 for some reason…, and v and ω are just totally off.

And somehow Glower’s works just fine:



Omg… Nvm. I got it now…

I had a g=-9.8 instead of g=9.8 in my GantryDynamics function and also was using dXe = A\*X + B\*U + H\*C\*(X-Xe) instead of dXe = A\*Xe + B\*U + H\*C\*(X-Xe)…

GantryObs.m

**rng(1168874);**

**% Gantry System ( Sp21 version)**

**mgc = 2.0;**

**ml = 1.0;**

**L = 1.0;**

**%% System model**

**[A,B] = linearizedGantry(mgc,ml,L);**

**C = [1,0,0,0];**

**%% I checked observability through x and system is observable**

**% NN = rank([C; C\*A; C\*A^2; C\*A^3])**

**% pause**

**%% Now compute H using Bass-Gura**

**des\_poles = [-1 -2 -3 -4];**

**H = transpose(placePoles(A', C', C, des\_poles));**

**%% Now check using step3**

**% Aaug = [A, zeros(4,4); H\*C, A-H\*C]; Baug = [B;B];**

**% Caug = eye(8); Daug = zeros(8,1);**

**% X0 = [zeros(4,1);5\*rand(4,1)];**

**% T\_end = 20; t = transpose(linspace(0,T\_end,1001));**

**% R = 0\*t + 1; % We'll just see step response**

**% Y = step3(Aaug, Baug, Caug, Daug, t, X0, R);**

**% plot(t,Y(:,[1:4]), t,Y(:,[5:8]),'--');**

**% grid on; xlabel('Time (s)'), ylabel('States'), title('Gantry System: Actual States vs State Estimates');**

**% pause;**

**% % return**

**%% Nonlinear Simulation**

**clf**

**X = zeros(4,1); Xe = 5\*rand(4,1);**

**dX = zeros(4,1); dXe = zeros(4,1);**

**dt = 100e-6; T\_end = 20; t = 0;**

**N = (T\_end / dt) + 1;**

**DATA = zeros(N,9); i=1;**

**tic**

**while(t < T\_end)**

**U = 2\*cos(t);**

**dX = GantryDynamics(X,U, mgc,ml,L);**

**dXe = A\*Xe + B\*U + H\*C\*(X - Xe);**

**X = X + dX \* dt;**

**Xe = Xe + dXe \* dt;**

**t = t + dt;**

**DATA(i,:) = [X', Xe', norm(dXe)];**

**i = i+1;**

**end**

**toc**

**t = [1:length(DATA)]' \* dt;**

**DATAds = downsample(DATA,10);**

**tds = downsample(t,10);**

**subplot(2,2,1);**

**plot(t,DATA(:,1), t,DATA(:,5),'--');**

**legend('x(t)', 'x\_{est}(t)');**

**grid on;**

**title('x(t) vs x\_{est}(t)'); xlabel('Time (s)');**

**subplot(2,2,2);**

**plot(t,DATA(:,2), t,DATA(:,6),'--');**

**legend('\theta(t)', '\thetax\_{est}(t)');**

**grid on;**

**title('\theta(t) vs \theta\_{est}(t)'); xlabel('Time (s)');**

**subplot(2,2,3);**

**plot(t,DATA(:,3), t,DATA(:,7),'--');**

**legend('v(t)', 'v\_{est}(t)');**

**grid on;**

**title('v(t) vs v\_{est}(t)'); xlabel('Time (s)');**

**subplot(2,2,4);**

**plot(t,DATA(:,4), t,DATA(:,8),'--');**

**legend('\omega(t)', '\omega\_{est}(t)');**

**grid on;**

**title('\omega(t) vs \omega\_{est}(t)'); xlabel('Time (s)');**

GantryDynamics.m

**function [ dX ] = GantryDynamics( X, F, mgc, ml, L )**

**% X = [x, q, dx, dq]**

**x = X(1);**

**q = X(2);**

**dx = X(3);**

**dq = X(4);**

**g = 9.8;**

**M = [(mgc+ml), ml\*L\*cos(q); ml\*L\*cos(q), ml\*L^2];**

**A = [ml\*L\*dq\*dq\*sin(q); -ml\*L\*g\*sin(q)];**

**B = [1;0];**

**d2X = inv(M) \* (A + B\*F);**

**dX = [dx; dq; d2X];**

**end**

**Separation Principle – i.e., putting observers to use!**

Using U = Kr\*R – Kx\*X:

HeatObs.m

**% 20-stage RC Filter**

**% Lecture #16**

**% Servo Compensators at DC**

**%% System description**

**R = 0.2; Cap = 0.2;**

**A = zeros(20,20);**

**for i=1:19**

**A(i,i) = -2/(R\*Cap);**

**A(i,i+1) = 1/(R\*Cap);**

**A(i+1,i) = 1/(R\*Cap);**

**end**

**A(20,20) = -1/(R\*Cap);**

**B = zeros(20,1);**

**B(1) = 1/(R\*Cap);**

**C = zeros(1,20);**

**C(20) = 1;**

**D = 0;**

**% 4th Order Approximation**

**Rapprox = 5\*R; Capprox = 5\*Cap;**

**aa = 1/(Rapprox\*Capprox);**

**Aapprox = zeros(4,4);**

**for i=1:3**

**Aapprox(i,i) = -2\*aa;**

**Aapprox(i,i+1) = aa;**

**Aapprox(i+1,i) = aa;**

**end**

**Aapprox(4,4) = -aa;**

**Bapprox = zeros(4,1);**

**Bapprox(1) = aa;**

**Capprox = zeros(1,4);**

**Capprox(4) = 1;**

**Dapprox = D;**

**%% Simple Compensator**

**des\_poles = [-1, -2, -3, -4];**

**[Kx, Kr] = placePoles(Aapprox, Bapprox, Capprox, des\_poles);**

**%% Observer**

**des\_poles\_H = [-3,-4,-5,-6];**

**H = transpose(placePoles(Aapprox',Capprox',Capprox,des\_poles\_H));**

**Ae = Aapprox; Be = Bapprox; Ce = Capprox; De = D;**

**%% Linear Simulation**

**% U = Kr\*R - Kx\*X**

**Aaug = [Ae, -Be\*Kx; H\*Ce, Ae-H\*Ce-Be\*Kx]; Baug = [Be;Be]\*Kr;**

**Caug = [Ce, Ce\*0; 0\*Ce, Ce]; Daug = [0;0];**

**t = transpose(linspace(0,15,1001)); R = 0\*t + 1;**

**X0 = [zeros(4,1); rand(4,1)];**

**Y = step3(Aaug,Baug,Caug,Daug, t,X0,R);**

**plot(t,Y,'LineWidth',2);**

**legend('y(t)','y\_{est}(t)');**

**grid on; title('Linear Simulation: y vs y\_{est}'); xlabel('Time (s)');**

**pause;**

**%% Nonlinear Simulation**

**V = zeros(20,1); Z = 0; Xe = zeros(4,1);**

**dt = 100e-6; T\_end = 20; t = 0;**

**Ref = 1;**

**N = (T\_end / dt) + 1;**

**DATA = zeros(N,10);**

**i=1;**

**tic**

**while(t < T\_end)**

**V0 = Kr\*Ref - Kx\*Xe;**

**% V0 = -Kz\*Z - Kx\*V([5,10,15,20]);**

**% V0 = 1 + sin(2\*t);**

**dV = A\*V + B\*V0;**

**% dZ = V(20) - Ref;**

**dXe = Ae\*Xe + Be\*V0 + H\*(V(20)-Ce\*Xe);**

**V = V + dV \* dt;**

**% Z = Z + dZ \* dt;**

**Xe = Xe + dXe \* dt;**

**t = t + dt;**

**DATA(i,:) = [V([5,10,15,20])', Xe', V0, V(20)];**

**i = i+1;**

**% plot([0:20], [V0;V], 'b.-', t, 0, 'b+',20,Ref,'b+');**

**% ylim([0,3\*VMAX]);**

**% xlim([0,20]);**

**% grid on;**

**% pause(100e-6);**

**end**

**toc**

**t = [1:length(DATA)]' \* dt; %#ok<NBRAK>**

**DATAds = downsample(DATA,10);**

**tds = downsample(t,10);**

**subplot(1,2,1);**

**plot(t,DATA(:,[1:4]), t,DATA(:,[5:8]), '--');**

**grid on;**

**title('Simulated Response of Heat System: State vs Observer State'); xlabel('Time (s)');**

**subplot(1,2,2);**

**plot(t,DATA(:,9), t,DATA(:,10),'LineWidth',2);**

**legend('u(t)','y(t)');**

**grid on;**

**title('Output of System'); xlabel('Time (s)');**

