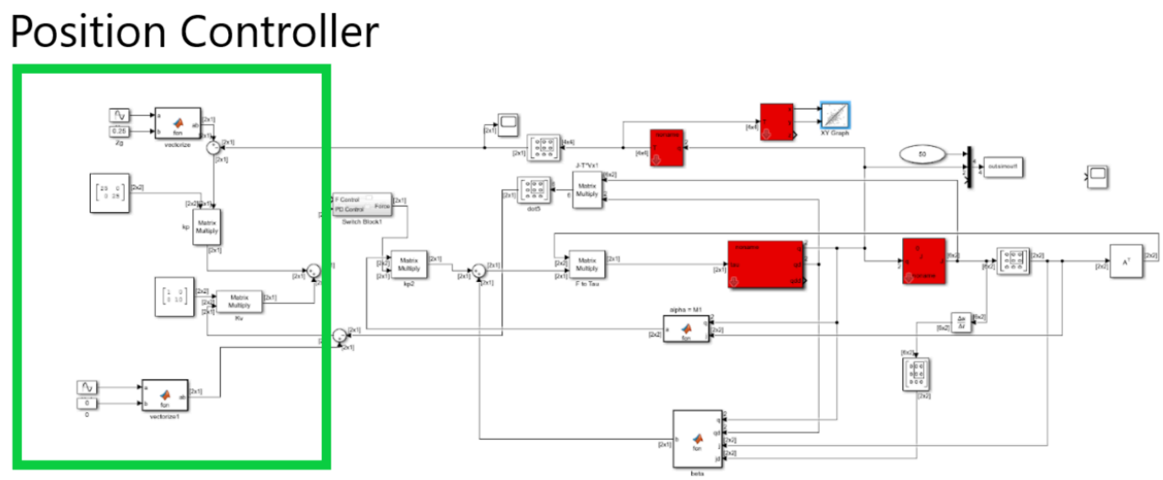
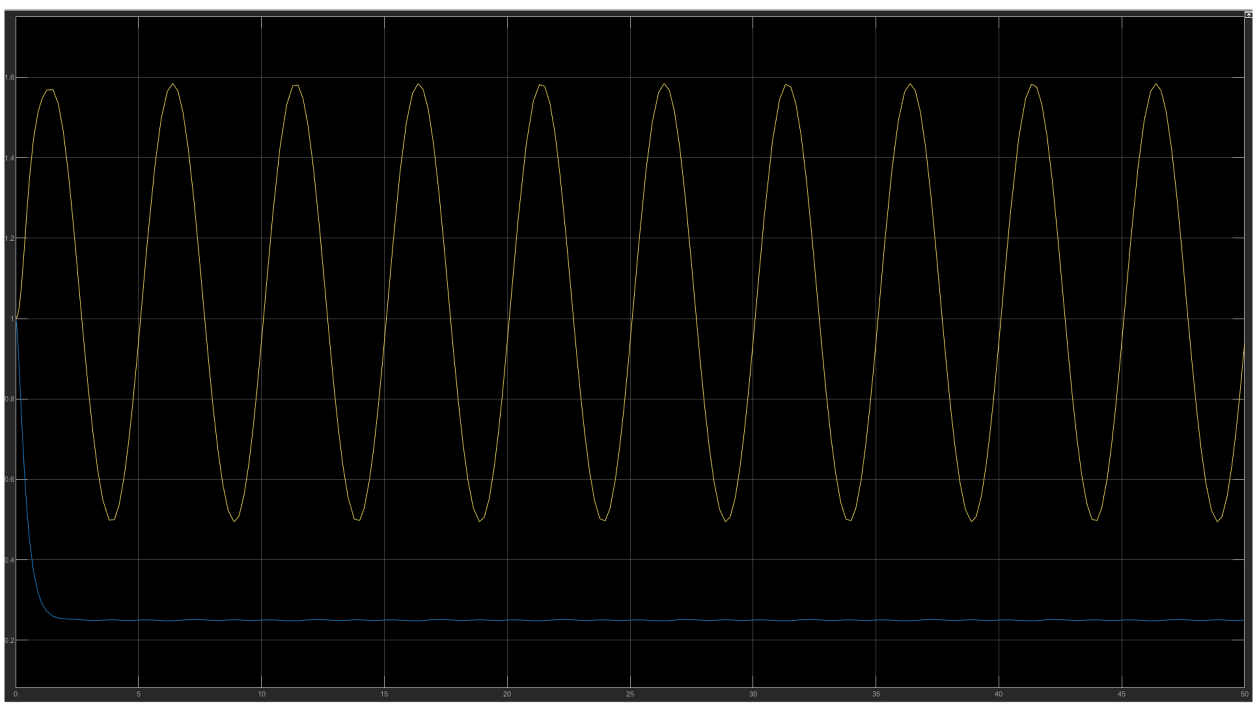
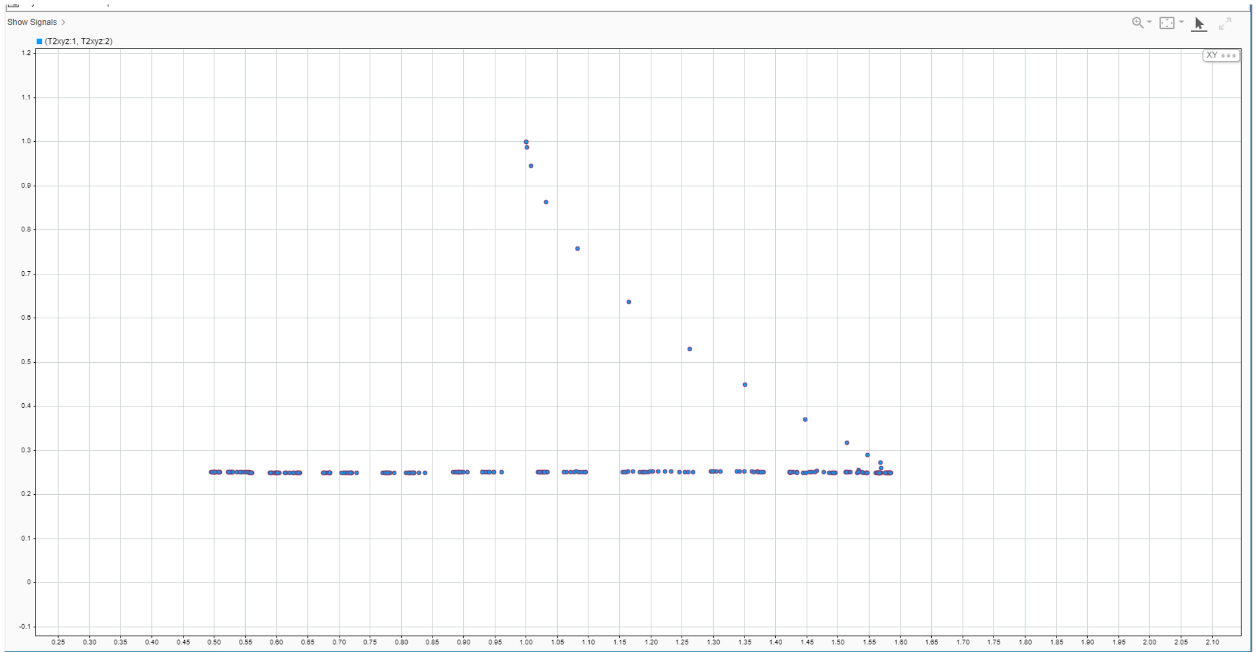
Aly Khater - Robotics Final Project
Part a - Nonlinear 2-DOF Position Control



Simulink Model - Left to right functions (top to bottom)
Vectorize - Alpha = M1 - Beta

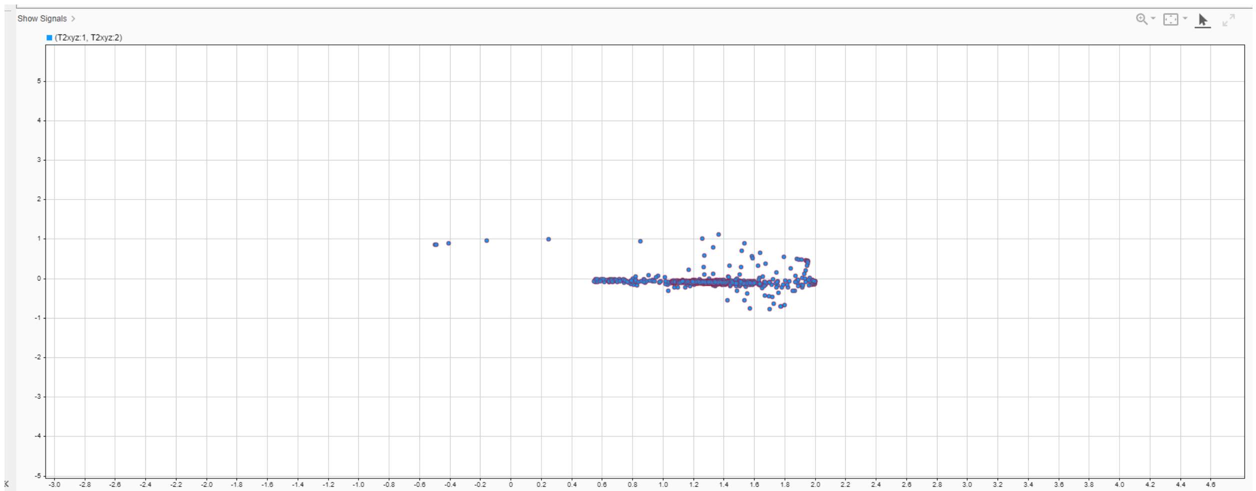


Endpoint time response - Position vs Time

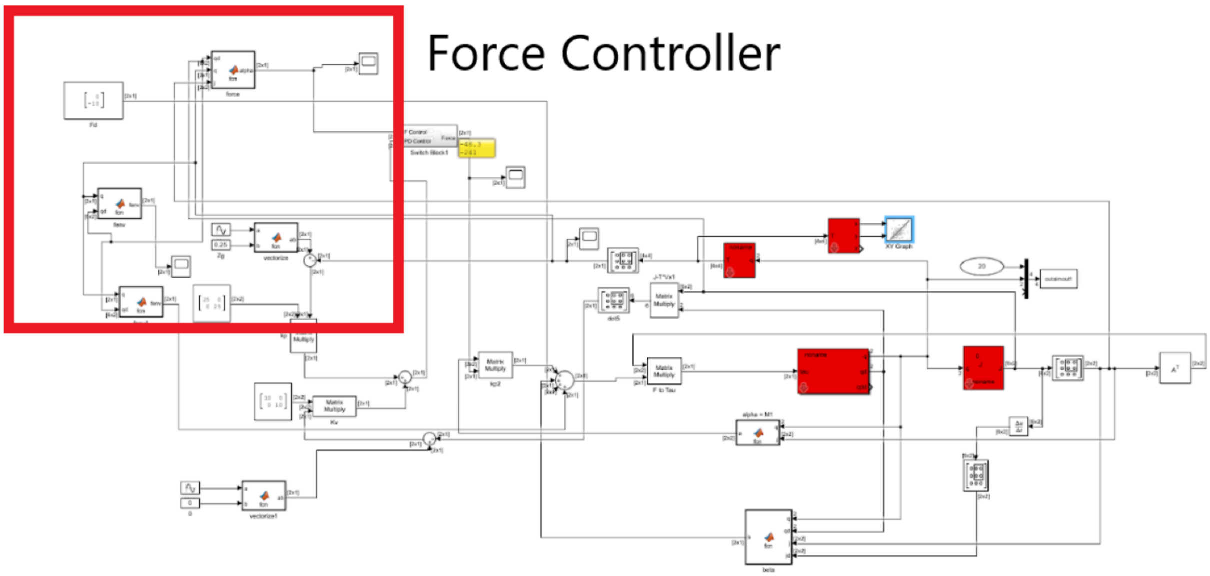


Endpoint position over time. X vs Z

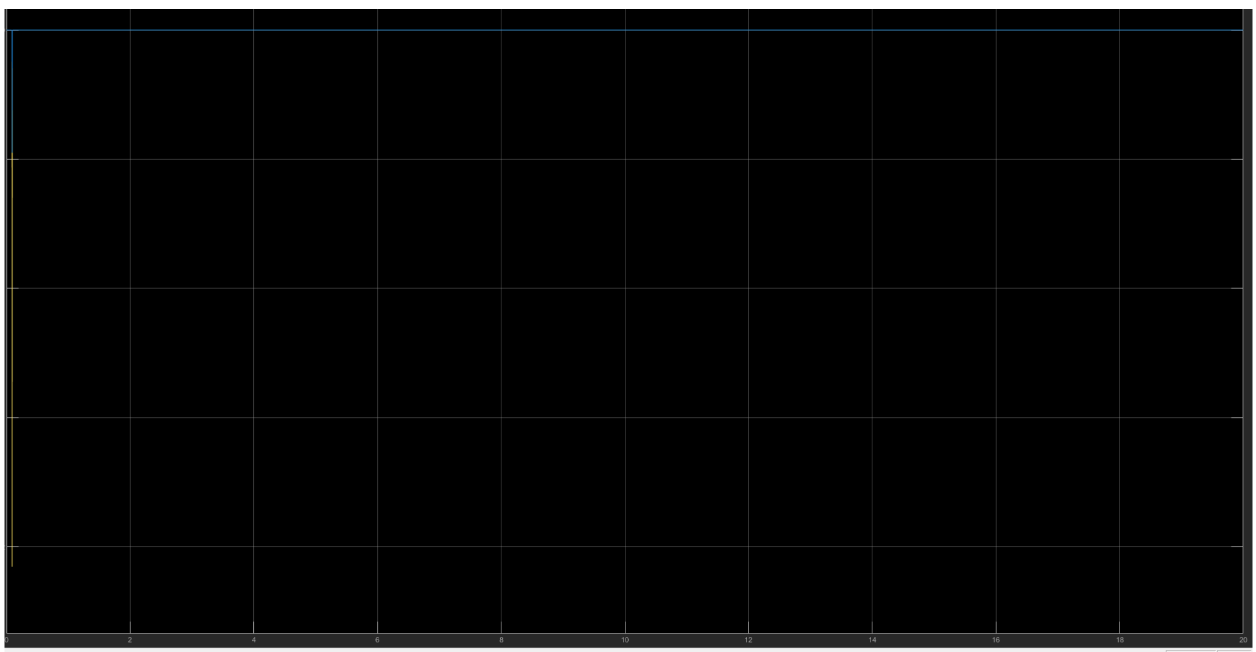
Part b - Nonlinear 2-DOF Force Control

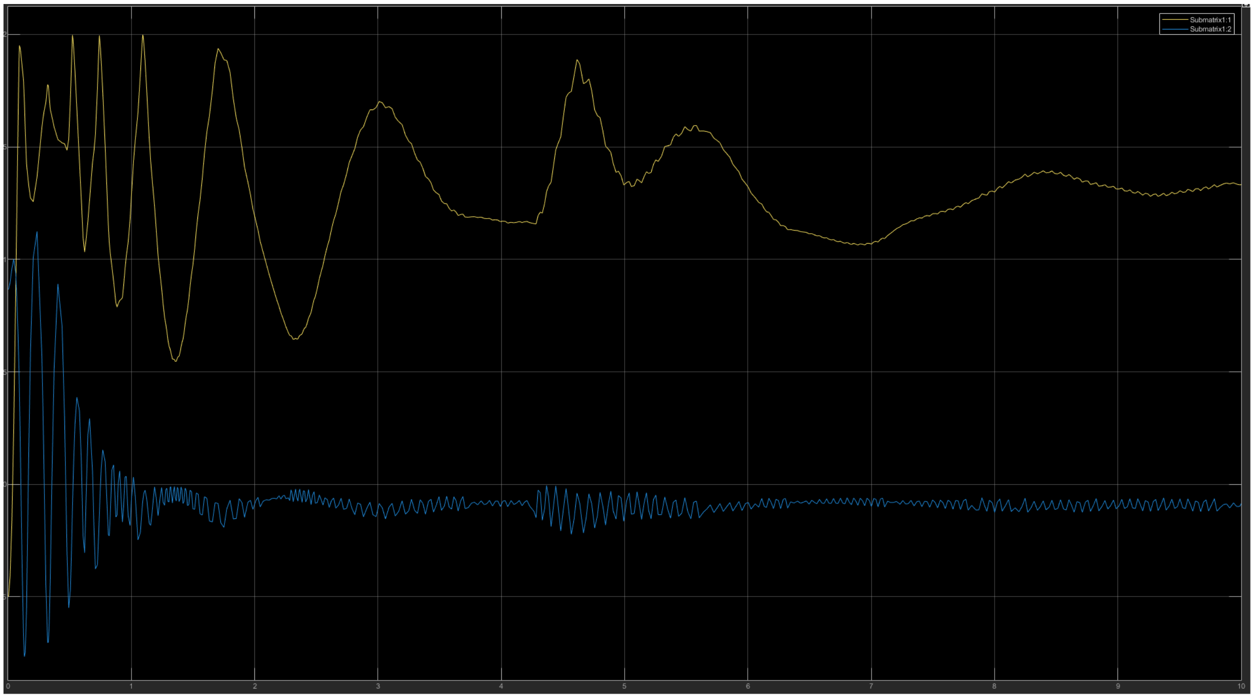


Endpoint position over time. X vs Z



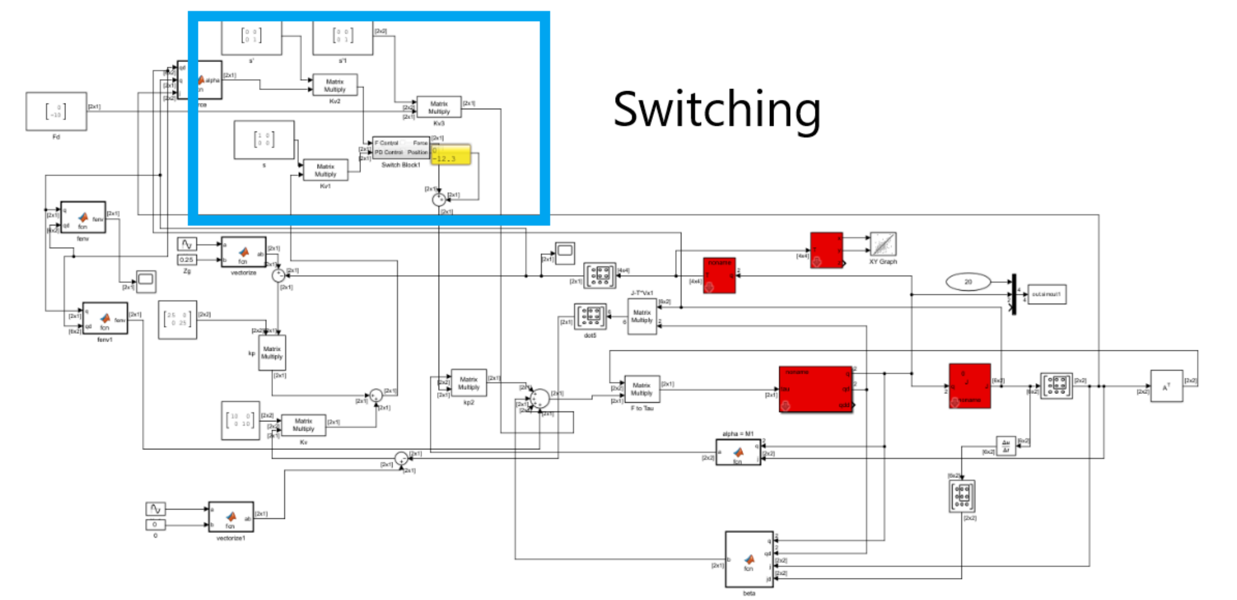
Simulink Model - Left to right functions (top to bottom)
Fenv - Force - Vectorize - Alpha - Beta



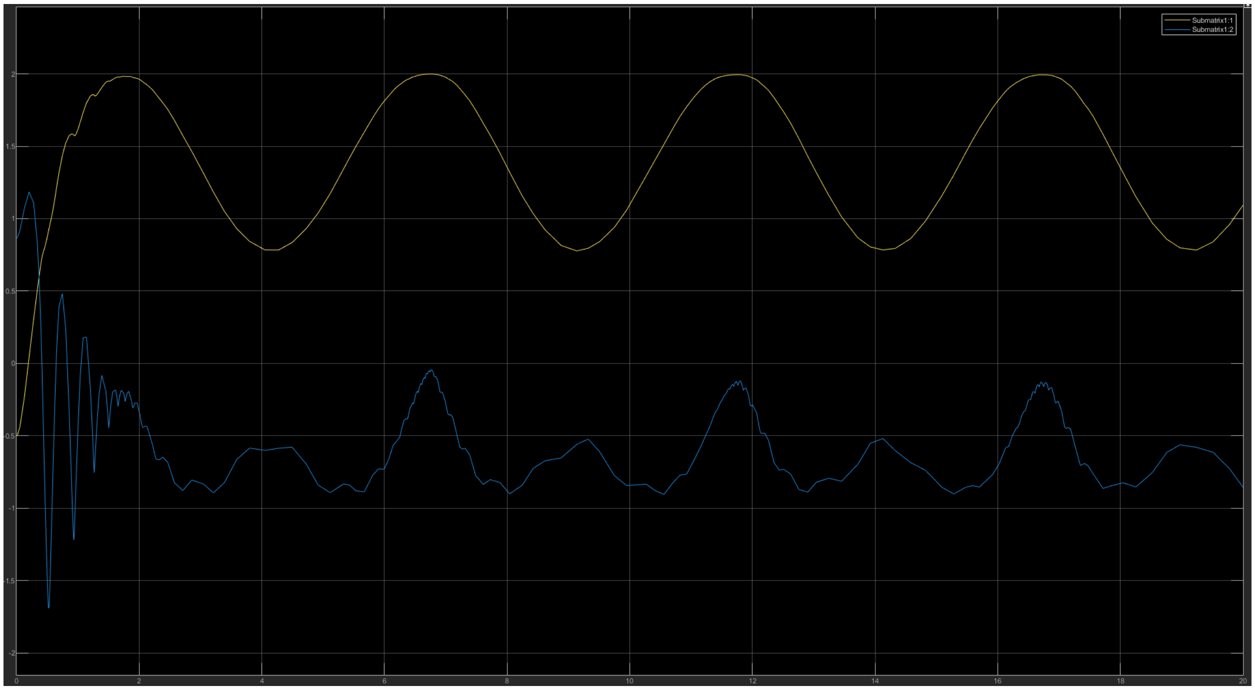


Force response over time - Force vs Time

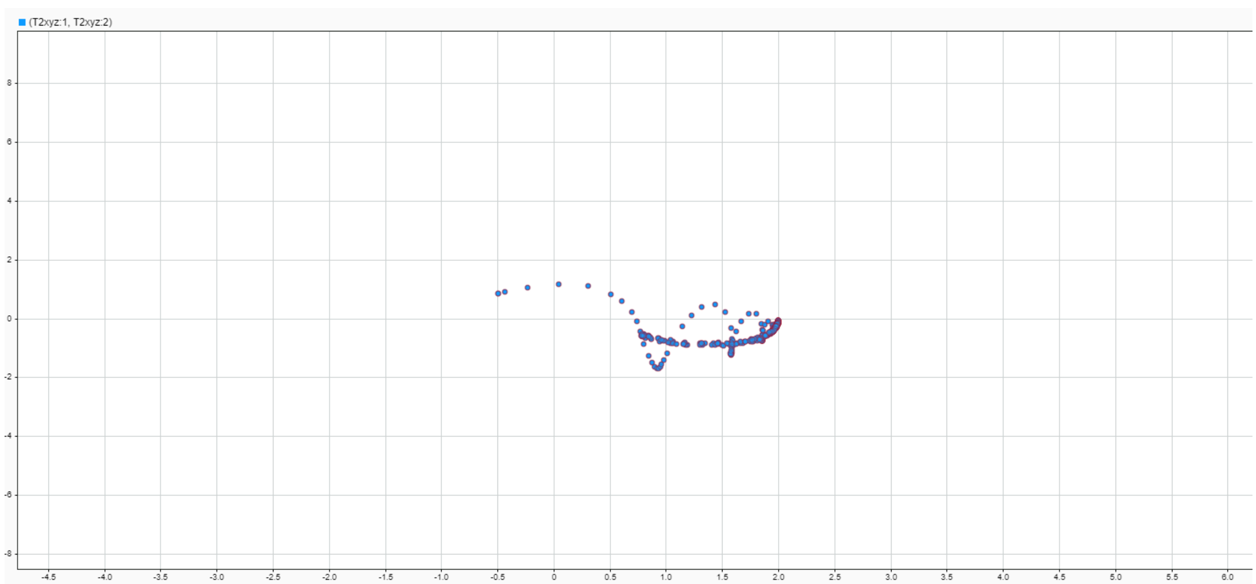
Part c - Nonlinear Hybrid Control



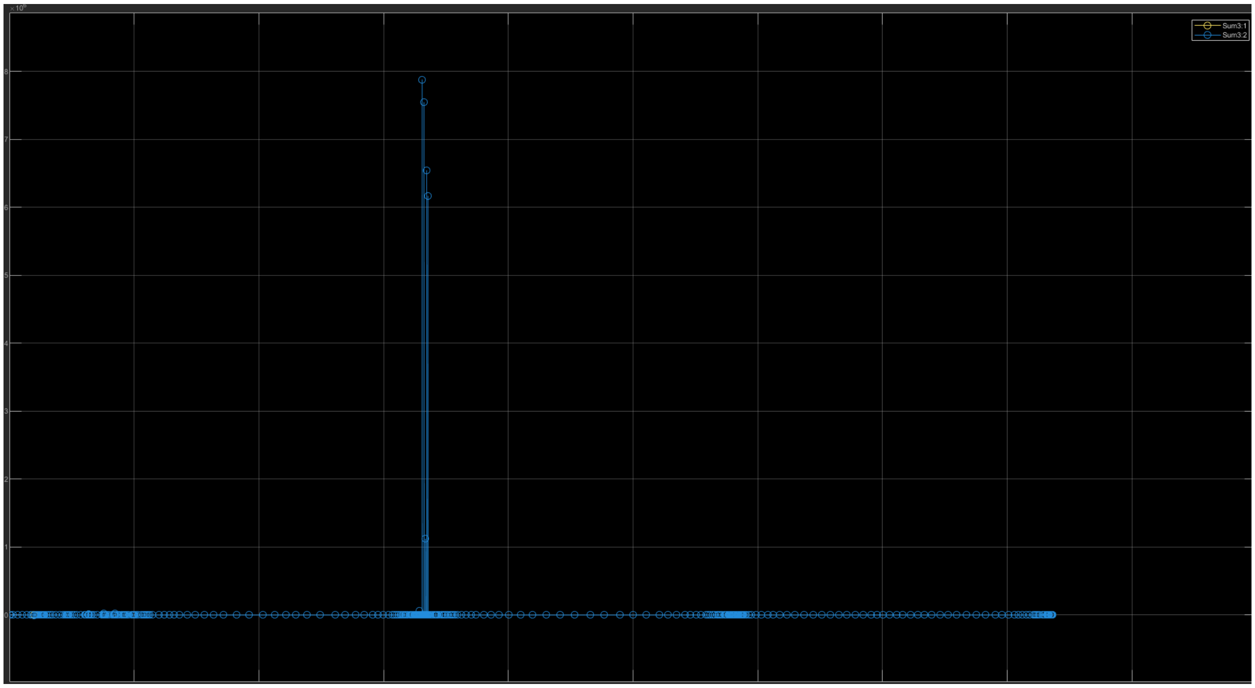
Simulink Model - Left to right functions (top to bottom)
Fenv - Force - Vectorize - Alpha - Beta



Force response over time - Force vs Time



Endpoint position over time. X vs Z



Code

```
function a = fcn(q,j)
%Variable Definitions
l1 = 1;
l2 = 1;
m1 = 3;
m2 = 3;
c2 = cos(q(2));
%Jacobian Definitions
ji = inv(j);
jit = ji';

%Mass matrix Joint Space
m = [l2^2*m2+2*l1*l2*m2*c2+l1^2*(m1+m2), l2^2*m2+l1*l2*m2*c2;l2^2*m2+l1*l2*m2*c2,l2^2*m2];
%Mass Matrix Cartesian
a = jit*m*ji;
end
```

Alpha function, utilizing Mx function from the book.

```
function b = fcn(q,qd,j,jd)
%Variable definitions
l1 = 1;
l2 = 1;
m1 = 3;
m2 = 3;
g = 9.81;
b1 = 3;
b2 = 3;
c1 = cos(q(1));
c2 = cos(q(2));
s1 = sin(q(1));
s2 = sin(q(2));
c12 = cos(q(1)+q(2));
s12 = sin(q(1)+q(2));
t1 = qd(1);
t2 = qd(2);
%jacobian Definitions
ji = inv(j);
jit = ji';
%Mass
m = [l2^2*m2+2*l1*l2*m2*c2+l1^2*(m1+m2), l2^2*m2+l1*l2*m2*c2;l2^2*m2+l1*l2*m2*c2,l2^2*m2];
%Velocity
v = [-m2*l1*l2*s2*t2^2 - 2*m2*l1*l2*s2*t1*t2; m2*l1*l2*s2*t1^2];
vx = jit*(v-m*ji*jd*qd);
%Gravity
g =[m2*l2*g*c12 + (m1+m2)*l1*g*c1; m2*l2*g*c12];
gx = jit*g;
%friction
f = [t1*b1 ; t2*b2];
fx = ji'*f*ji;

%Beta
b = vx+gx+fx;

end
```

Beta function, utilizing Vx, Gx, Fx functio from the book.

ProjectccccccHOLYSHITYES ▸ vectorize1

```matlab
function ab = fcn(a,b)

ab = [a;b];
end
```

Function to vectorize values

```matlab
function fenv = fcn(q,qd)
    ke = 1000; %Stiffness
    z = 0;
    if q(2) <= z  %Checks if position has hit the wall
        fenv = [0; ke*q(2)]; %Wall exerts a force
    else
        fenv = [0;0]; %No force exerted from wall
    end
end
```

Force from the environment

```
function force = fcn(qd, q, j)
    %Variable Definitions
    %High gain values for overshoot
    kpf = [1000 0; 0 1000];
    kvf = [60 0; 0 60];
    m1 = 3;
    m2 = 3;
    l1 = 1;
    l2 = 1;
    fd = [0; 0];
    ke = 1000;
    ke_inv = 1 / ke;
    t1 = q(1);
    t2 = q(2);
    c2 = cos(q(2));
    %Jacobian Definitions
    j1 = inv(j);
    jjt = j * j';

    %Mass Matrix
    M = [m2*l2^2 + 2*l1*l2*m2*c2 + (m1 + m2)*l1^2, m2*l2^2 + l1*l2*m2*c2;
         m2*l2^2 + l1*l2*m2*c2, m2*l2^2];
    mx = jjt * M * j_inv;
    %Fenvironment
    if q(2) <= 0
        fenv = [0; ke * t2];
    else
        fenv = [0; 0];
    end

    x_dot = [qd(1); qd(2)];
    %Error between desired force and environmental force
    ef = fd - fenv;
    %Force equation from the book
    force = mx * (kpf * ke_inv * ef - kvf * x_dot) + fd;
end
```

Force function

SCURRTWOLINK.m | slaccel.m

```matlab

%clear all

L{1} = link([0 1 0 0 0], 'standard');
L{2} = link([0 1 0 0 0], 'standard');
L{1}.m = 3;
L{2}.m = 3;
L{1}.r = [0 0 0];
L{2}.r = [0 0 0];
L{1}.I = [0 0 0 0 0 0];
L{2}.I = [0 0 0 0 0 0];
L{1}.Jm = 0;
L{2}.Jm = 0;
L{1}.G = 1;
L{2}.G = 1;
L{1}.B = 3;
L{2}.B = 3;

%useful poses
qz=[0 pi/2];    %zero angle

r = robot(L);

r.gravity = [0; 9.81; 0];
```

Robot Definition