Table of Contents


```
% This script runs a verification for the non-equilibrium trajectory
% generated from a 4th order polynomial:
```

Simulation Parameters:

```
T   = 5;      % Time horizon/final time.
dt  = 0.01;   % Step size.
s   = 0:dt:T; % Time series vector.
g   = 9.81;   % acceleration due to gravity (kgm/s^2).

% Solved input:
u   = 0.0048*s.^4 + 0.144*s.^3 - 1.32*s.^2 + 2.4*s + (g/2);
% Rotor thrust as a function of time.

% System characteristics and state vector initialization:
% The drone starts from rest. Then climbs 10m vertically and then
 slows
% down into a hover. The time horzon for this to happen is 2seconds.

% Positions:
q_theta = zeros(1,length(s));
q_h = zeros(1,length(s));
q_v = zeros(1,length(s));

% Velocity:
q_theta_t = zeros(1,length(s));
q_h_t = zeros(1,length(s));
q_v_t = zeros(1,length(s));

% Acceleration:
q_theta_tt = zeros(1,length(s));
q_h_tt = zeros(1,length(s));
q_v_tt = zeros(1,length(s));
```

Non-linear system model:

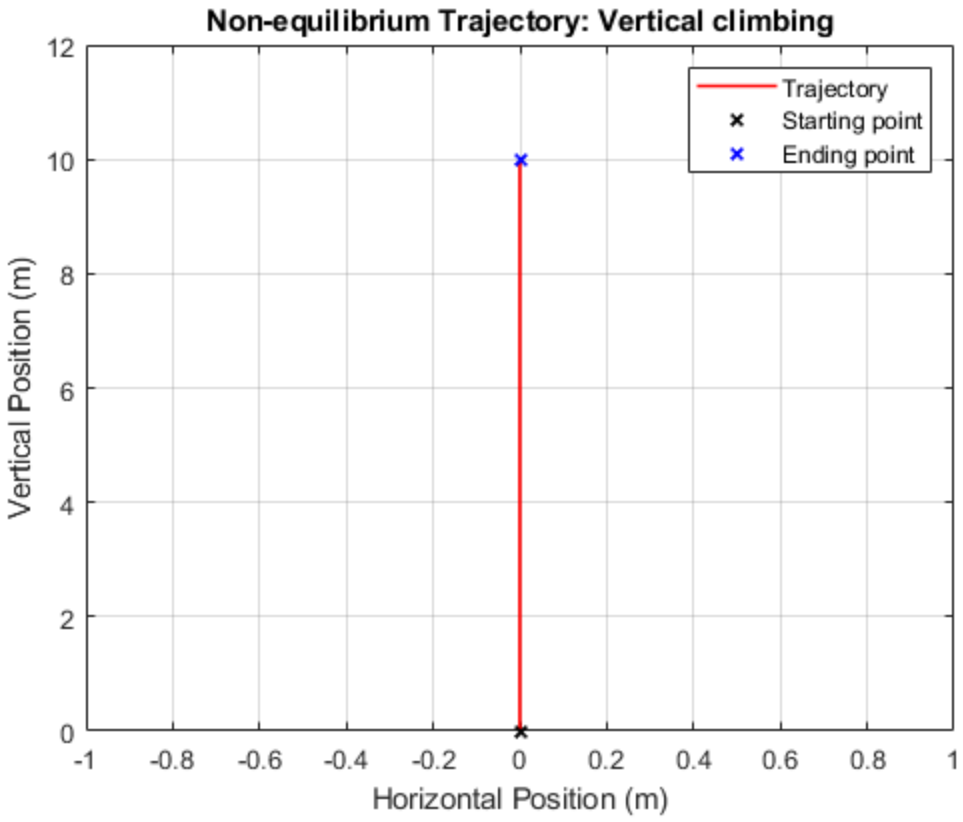
```
for i = 1:length(s)
    if i == 1
        % Calculate acceleration terms at initial position:
        q_theta_tt(i) = (u(i) - u(i))*100;
        q_h_tt(i)     = 2*u(i)*sin(q_theta(i)) - 0.1*q_h_t(i);
```

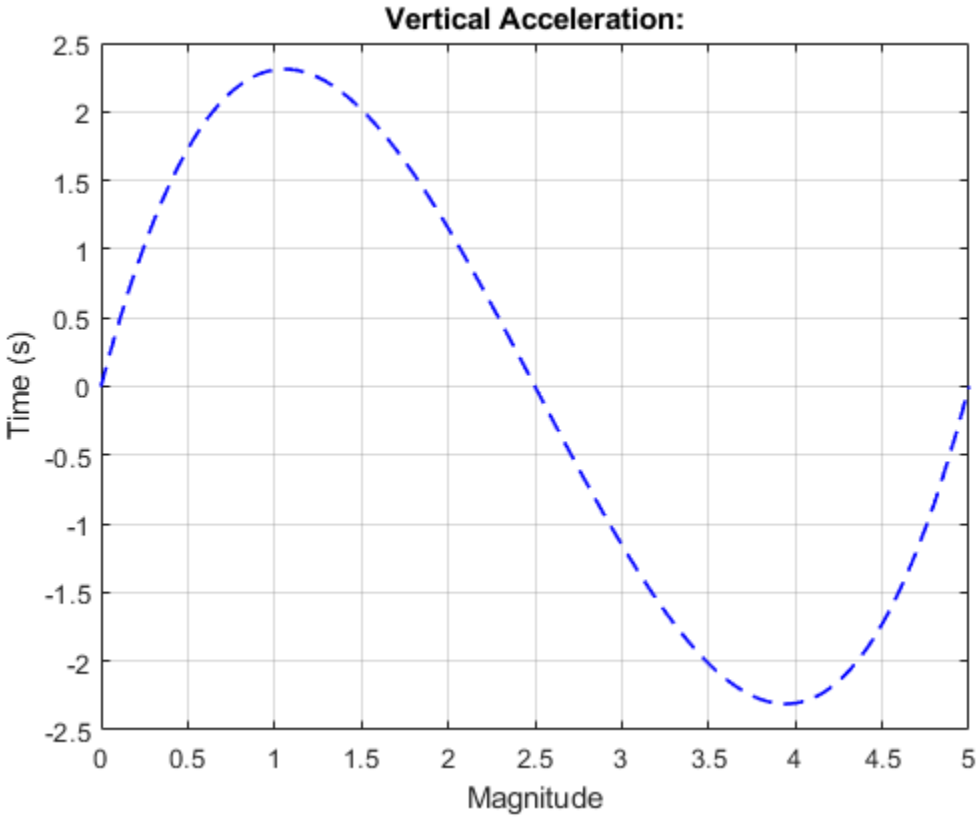
```
        q_v_tt(i)     = 2*u(i)*cos(q_theta(i)) - 0.1*q_v_t(i) -g;
    else
        % Update states:
        % Angular position:
        q_theta(i) = q_theta(i-1) + q_theta_t(i-1)*dt +
 0.5*q_theta_tt(i-1)*dt^2;
        q_theta_t(i) = q_theta_t(i-1) + q_theta_tt(i-1)*dt;
        % Horizontal position:
        q_h(i) = q_h(i-1) + q_h_t(i-1)*dt + 0.5*q_h_tt(i-1)*dt^2;
        q_h_t(i) = q_h_t(i-1) + q_h_tt(i-1)*dt;
        % Vertical position:
        q_v(i) = q_v(i-1) + q_v_t(i-1)*dt + 0.5*q_v_tt(i-1)*dt^2;
        q_v_t(i) = q_v_t(i-1) + q_v_tt(i-1)*dt;
        % Calculate accelerations:
        q_theta_tt(i) = (u(i) - u(i))*100;
        q_h_tt(i)     = 2*u(i)*sin(q_theta(i)) - 0.1*q_h_t(i);
        q_v_tt(i)     = 2*u(i)*cos(q_theta(i)) - 0.1*q_v_t(i) -g;
    end
end
```

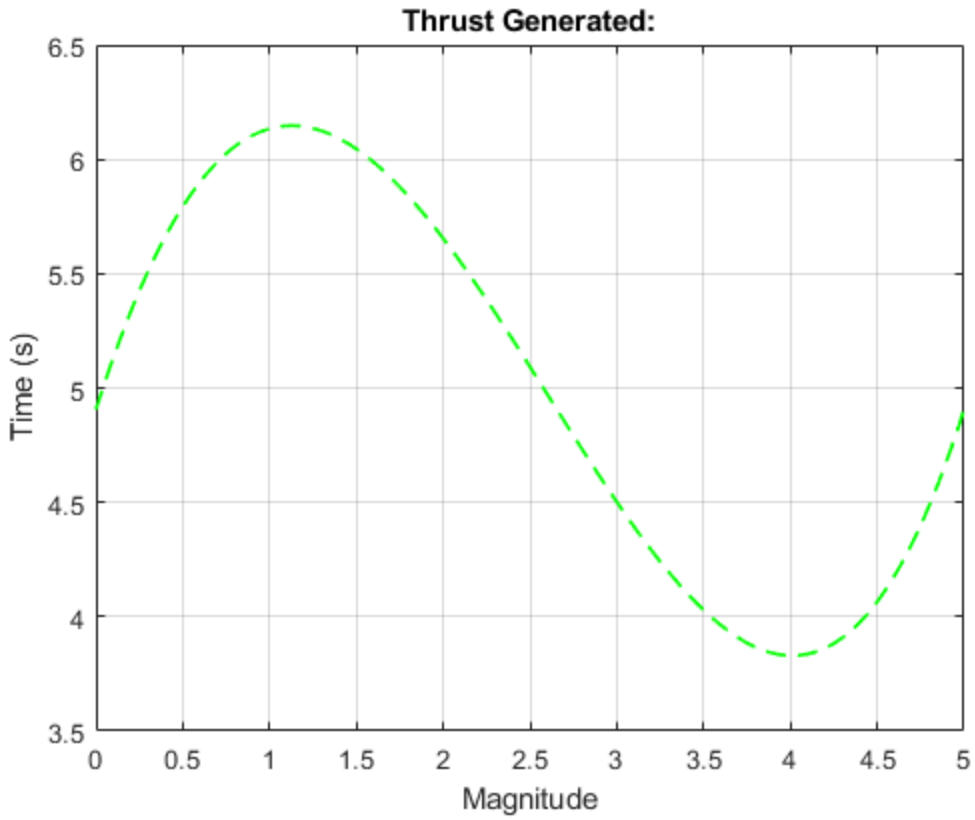
Plotting results:

Trajectory plot:

```
figure()
plot(q_h,q_v,'r','LineWidth',1.2);
hold on; grid on;
plot(q_h(1),q_v(1),'kx','LineWidth',1.2);
plot(q_h(end),q_v(end),'bx','LineWidth',1.2);
xlabel('Horizontal Position (m)');
ylabel('Vertical Position (m)');
title('Non-equilibrium Trajectory: Vertical climbing');
legend('Trajectory','Starting point','Ending point');

% Vertical acceleration plot:
figure()
plot(s,q_v_tt,'b--','LineWidth',1.2);
grid on;
xlabel('Magnitude');
ylabel('Time (s)');
title('Vertical Acceleration:');

% Input Magnitude plot:
figure()
plot(s,u,'g--','LineWidth',1.2);
grid on;
xlabel('Magnitude');
ylabel('Time (s)');
title('Thrust Generated:');

% End.
```


Non-equilibrium Trajectory: Vertical climbing
Trajectory
Starting point
Ending point
Vertical Position (m)
Horizontal Position (m)
12
10
8
6
4
2
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Vertical Acceleration:
Time (s)
Magnitude
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5



Published with MATLAB® R2018b