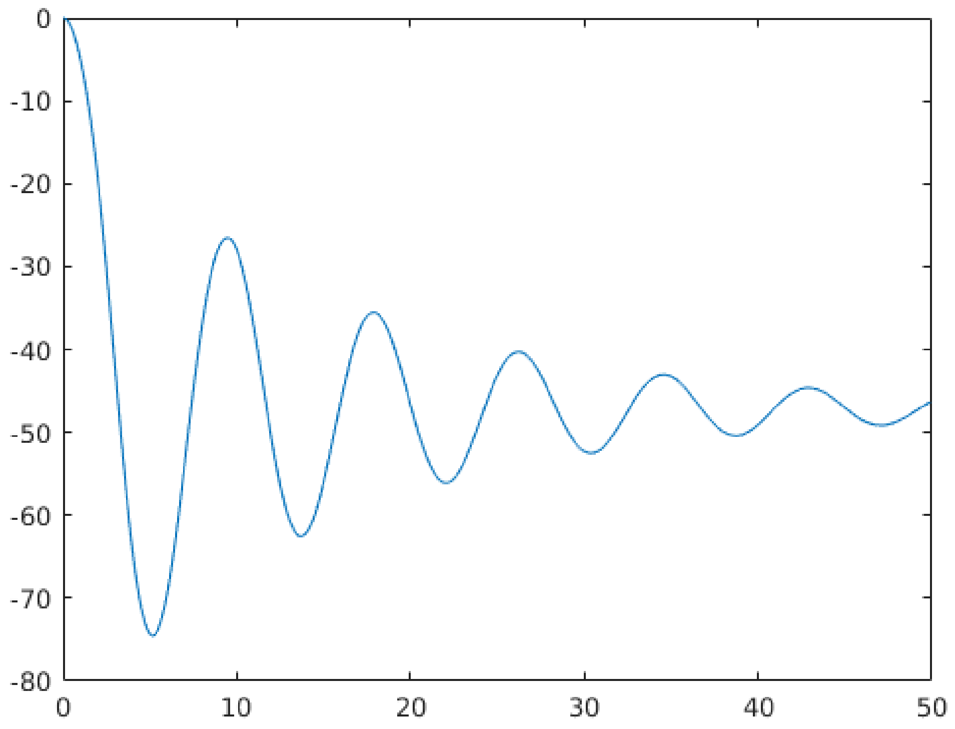
```
% problem 1 - bungee jumper

type freefall.m

% set up initial conditions
y0 = 0;
v0 = 0;
t0 = 0;
s = [y0, v0];
t = t0;

% parameters needed
tau = 0.1; % time step
m = 70; % kg
Cd = 0.5;
rho = 1.2; % kg/m^3
A = 0.5; % m
L = 30; % m, rest length of cord
beta = 0.057; % 1/sec
k = 40; % N / m
params = [m, Cd, rho, A, beta, k, L];
tf = 50; % sec

% find position vs time
steps = (tf - t0) / tau;
yff = zeros(1, steps);
tff = zeros(1, steps);
for i = 1:steps
    % add current height and time to accumulators
    yff(i) = s(1);
    tff(i) = t;
    % get new state from rk4
    s = rk4(s, t, tau, 'freefall', params);
    t = t + tau;
end

plot(tff, yff)


function [ds] = freefall(s, t, param)
% [ds] = freefall(s, t, param)
% rhs of differential equation for free fall
% s = state vector [v, x]
% t = current time
% param = [m, Cd, rho, A, beta, k, L]
% ds = derivative of s

% get state variables from vector
y = s(1);
v = s(2);
% get parameters
m = param(1);
```

```
Cd = param(2);
rho = param(3);
A = param(4);
beta = param(5);
k = param(6);
L = param(7);
omega = sqrt(k / m);
g = 9.81;
% forces on jumper
Fg = -g * m;
Fd = -Cd * rho * A * norm(v) / 2 * v;
if abs(y) > 30
    x = sign(y) * (abs(y) - abs(L));
    F_cord = -2*m*beta*v - m*omega^2*x;
else
    F_cord = 0;
end
% return rhs
a = (Fg + Fd + F_cord) / m;
dv = a; % derivative of velocity
dy = v; % derivative of position
ds = [dy, dv];
end
```



Published with MATLAB® R2019a