

---

```

function rk4_anim_matlab
% Solving the Double Pendulum with 4th order Runge-Kutta Integration
%=====
% A 4th order Runge-Kutta method is implemented to solve the      %
% double pendulum equations of motion. This method approximates  %
% a function f(x) on a given interval [a,b] by expanding the      %
% function into a 4th order Taylor series at the midpoint of      %
% [a,b]. The 2nd order double pendulum equations of motion are    %
% broken into four 1st order ODE's and are integrated by the 4th %
% order Runge-Kutta method. To analyze the behavior of the      %
% double pendulum, phase space and power spectrum plots are made. %
%                                                                    %
% Code design for rk4 integration from "A Survey of Computational  %
% Physics" by Landau, Paez, Bordeianu.                             %
%=====
% ABW 2015 %
%=====
clc;

pend_animation = 'off'; % Turn pendulum animation on/off
phase_plot = 'off'; % Turn phase space plot on/off
pspec_plot = 'off'; % Turn power spectrum plot on/off

% Initialize parameters
neq = 4; % Number of equations
g = 9.807; % Acceleration of gravity [m/s^2]
a = 0.0; % Lower limit of integration [a,b]
b = 100; % Upper limit of integration [a,b]
h = 0.04; % Step size
npts = (b - a)/h; % Number of points to integrate
tb = 10; % Duration of animation, 0.0 < tb <= b
x_range = 0.4; % Range of x in power spec. plot, 0.0 < x_range <= pi

% Preallocate vectors
y = zeros(1,neq);
ymod = zeros(1,npts);
th1 = zeros(1,npts);
th2 = zeros(1,npts);
thldot = zeros(1,npts);
th2dot = zeros(1,npts);
omega = zeros(1,npts);

% Initial conditions of ODE
m1 = 1.0; % Mass of upper pendulum
m2 = 1.0; % mass of lower pendulum
L1 = 1.0; % Length of upper pendulum
L2 = 1.0; % Length of lower pendulum

y(1) = pi/4.0; % Angle of upper pendulum
y(2) = 1.0; % Angular velocity of upper pendulum
y(3) = 0.0; % Angle of lower pendulum
y(4) = 1.0; % Angular velocity of lower pendulum

```

---

---

```

i = 1;
t = a;
% Integrate with Runge-Kutta
while t < b
    th1(i) = y(1);
    th2(i) = y(3);
    thldot(i) = y(2);
    th2dot(i) = y(4);
    %   fprintf('%d %21.16f %21.16f\n',i,y(1),y(2))

    y = rk4(neq,h,y,t,g,m1,m2,L1,L2);
    t = t + h;
    i = i + 1;
end

% Make phase space plot
if strcmp(phase_plot,'on') == 1
    figure(1)
    scatter(th1,thldot,4,'k.');
```

`xlabel('\theta_1','FontSize',14)`, `ylabel('d\theta_1/dt','FontSize',14)`  
`title('Phase Space','FontSize',14)`  
`print('phase_space_plot','-dpdf')`

```

end

% Fast Fourier transform data into frequency space
yk = fft(th1,npts);

i = 1;
omega_it = 0.0;
while omega_it < x_range
    omega(i) = pi*((2*i)/npts);
    omega_it = omega(i);
    ymod(i) = real(yk(i))^2 + imag(yk(i))^2;
    i = i + 1;
end

% Plot power spectrum to show dominant frequencies
if strcmp(pspec_plot,'on') == 1
    figure(2)
    scatter(omega,ymod,3,'ro');
```

`axis([0 x_range 0.01 100000000])`  
`set(gca,'yscale','log')`  
`xlabel('\omega','FontSize',14)`, `ylabel('|Y_1(\omega)|','FontSize',14)`  
`title('Power Spectrum','FontSize',14)`  
`print('power_spec_plot','-dpdf')`

```

end

% Animate double pendulum and save animation as 'pend_anim.gif'
if strcmp(pend_animation,'on') == 1
    xj = [L1*sin(th1(:)), L1*sin(th1(:)) + L2*sin(th2(:))];
    yj = [-L1*cos(th1(:)), -L1*cos(th1(:)) - L2*cos(th2(:))];

    figure(3)

```

---

---

```

imgpos = get(gcf, 'Position');
im_width = imgpos(3);
im_height = imgpos(4);
ia = 1;
ta = a;
anim = zeros(im_height,im_width,1,tb/h,'uint8');
while ta < tb
    ap = plot([0,xj(ia,1);xj(ia,1),xj(ia,2)], ...
        [0,yj(ia,1);yj(ia,1),yj(ia,2)], ...
        '-','MarkerSize', 25, 'LineWidth', 2);
    set(ap,{'Color'},{'blue';'red'})
    axis equal; axis([-L1-L2 L1+L2 -L1-L2 L1+L2]);
    title(sprintf('Time: %0.2f / %0.2f sec',ta,tb));
    drawnow;

    cur_frame = getframe(gcf);
    if ia == 1
        [anim(:,:,1,ia),col_map] = rgb2ind(cur_frame.cdata,256, ...
            'nodither');
    else
        anim(:,:,1,ia) = rgb2ind(cur_frame.cdata,col_map,'nodither');
    end
    ta = ta + h;
    ia = ia + 1;
end
imwrite(anim,col_map,'pend_anim.gif','LoopCount',inf,'DelayTime',0)
end
end

% Function to hold equations of motion
function dy = dy(i,x,y,g,m1,m2,L1,L2)
num11 = -m2*L2*y(2)*y(2)*sin(y(1) - y(3))*cos(y(1) - y(3));
num12 = g*m2*sin(y(3))*cos(y(1) - y(3));
num13 = -L2*m2*y(4)*y(4)*sin(y(1) - y(3)) - (m1 + m2)*g*sin(y(1));
den11 = L1*(m1 + m2) - m2*L1*cos(y(1) - y(3))*cos(y(1) - y(3));

num21 = m2*L2*y(4)*y(4)*sin(y(1) - y(3))*cos(y(1) - y(3));
num22 = g*(m1 + m2)*(sin(y(1))*cos(y(1) - y(3)) - sin(y(3)));
num23 = L1*(m1 + m2)*y(2)*y(2)*sin(y(1) - y(3));
den21 = L2*(m1 + m2) - m2*L2*cos(y(1) - y(3))*cos(y(1) - y(3));

switch i
    case 1
        dy = y(2);
    case 2
        dy = (num11 + num12)/den11 + num13/den11;
    case 3
        dy = y(4);
    case 4
        dy = (num21 + num22)/den21 + num23/den21;
end
end

% Function to perform rk4 integration

```

---

---

```
function y = rk4(n,h,y,x,g,m1,m2,L1,L2)
f1 = zeros(1,n);
f2 = zeros(1,n);
f3 = zeros(1,n);
f4 = zeros(1,n);
x1 = zeros(1,n);
x2 = zeros(1,n);
x3 = zeros(1,n);

for i = 1:n
    f1(i) = h*dy(i,x,y,g,m1,m2,L1,L2);
    x1(i) = y(i) + f1(i)/2.0;
end
for i = 1:n
    f2(i) = h*dy(i,x+(h/2.0),x1,g,m1,m2,L1,L2);
    x2(i) = y(i) + f2(i)/2.0;
end
for i = 1:n
    f3(i) = h*dy(i,x+(h/2.0),x2,g,m1,m2,L1,L2);
    x3(i) = y(i) + f3(i);
end
for i = 1:n
    f4(i) = h*dy(i,x+(h/2.0),x3,g,m1,m2,L1,L2);
    y(i) = y(i) + (f1(i) + 2.0*f2(i) + 2.0*f3(i) + f4(i))/6.0;
end
end
```

*Published with MATLAB® 7.14*