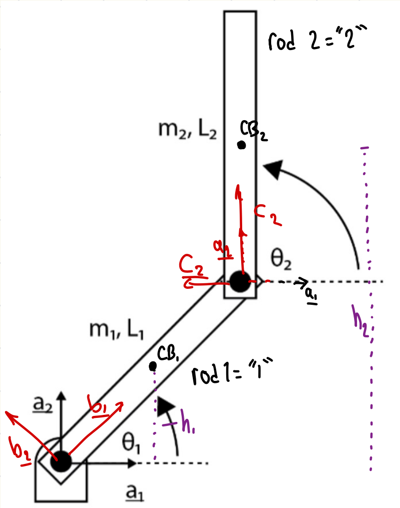


Finding the EOMs using the Lagrange Method

These are Known → $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = Q_i$ $KE - PE = L$

① Draw FBD, & draw fixed frame. Identify D.O.F.s → θ_1 & θ_2. Draw additional frames where needed. We won't use them here, but they're good to put down anyways.

② We will make Matlab do the majority of the work, but we do need to do some parts by hand. First, the position vectors for the centers of mass of both rods (2.5)

We keep in terms of A frame for simplicity (2.5)

$$r^{OCB_1} = \frac{L_1}{2}\cos\theta_1 \underline{a_1} + \frac{L_1}{2}\sin\theta_1 \underline{a_2}$$

$$r^{OCB_2} = L_1\cos\theta_1 \underline{a_1} + L_1\sin\theta_1 \underline{a_2} + \frac{L_2}{2}\cos\theta_2 \underline{a_1} + \frac{L_2}{2}\sin\theta_2 \underline{a_2}$$

We let Matlab take the derivative of these for us to get our velocities.

We want to plug the position vectors into the known equation for kinetic energy of a rigid body → $KE_i = \frac{1}{2}m_i\left({}^A\underline{V}^{C_i}\right) + \frac{1}{2} I^{B/C_B}\left({}^A\underline{w}^B\right)$

For COM of rod → $I^{B/CB} = \frac{1}{12}mL^2$

For ICR @ end of rod → $I^{B/ICR} = \frac{1}{3}mL^2$

We have positive thetas, so → $\left({}^A\underline{w}^B\right)^2 = (\dot{\theta}_i)^2 = \dot{\theta}_i^2$

③ Next, we find the Potential energy of the system. To do this, sum the PEs of all the bodies in the system. Since we only have the two rods, we only have to account for their potentials. PE of a rigid body = mgh

$$PE_1 = m_1 g h_1 \rightarrow \underset{\text{height of COM for 1}}{h_1 = \frac{L_1}{2}\sin\theta_1} \rightarrow PE_1 = m_1 g \frac{L_1}{2}\sin\theta_1$$

$$PE_2 = m_2 g h_2 \rightarrow \underset{\text{height of COM for 2}}{h_2 = L_1\sin\theta_1 + \frac{L_2}{2}\sin\theta_2} \rightarrow PE_2 = m_2 g L_1 \sin\theta_1 + m_2 g \frac{L_2}{2}\sin\theta_2$$

$$PE_{Total} = m_1 g \frac{L_1}{2}\sin\theta_1 + m_2 g L_1 \sin\theta_1 + m_2 g \frac{L_2}{2}\sin\theta_2$$

④ Since there are no non-conservative forces in the system, $Q_i = 0$.
We now can plug what we know into Matlab & let it do the rest.

```
% Michael Grady
% Dynamics Final Project, solving for EOMs of a double-pendulum system.
% I will comment on each line/section and explain what it is doing. This is a
% continuation of the hand-written portion that will also be submitted.

close all
clear
clc %cleaning up

syms m1 m2 L1 L2 g theta1(t) theta2(t)
%Declaring what our symbolic variables are going to be for the remainder of
%the script. We are also establishing theta1 and theta2 as our degrees of
%freedom that change as a function of time.

dtheta1 = diff(theta1,t);
dtheta2 = diff(theta2,t);
%Creating theta1dot and theta2dot for ease of use down the line.

r_OCB1a1 = .5*L1*cos(theta1);
%a1 component of position vector to COM of rod 1 from the origin
r_OCB1a2 = .5*L1*sin(theta1);
%a2 component of position vector to COM of rod 1 from the origin

r_OCB2a1 = L1*cos(theta1) + .5*L2*cos(theta2);
%a1 component of position vector to COM of rod 2 from the origin
r_OCB2a2 = L1*sin(theta1) + .5*L2*sin(theta2);
%a2 component of position vector to COM of rod 2 from the origin
%This is the reason that we left our position vectors in terms of the A
%frame instead of in terms of the B and C frames - so that this is simpler
%to use on Matlab.

V1a1 = diff(r_OCB1a1,t);
V1a2 = diff(r_OCB1a2,t);
V2a1 = diff(r_OCB2a1,t);
V2a2 = diff(r_OCB2a2,t);
%Each of these is just the derivative with respect to time of the above
%associated position vector, still defined in its specific direction. We do
%this for quality of life in the following lines.

inertia1 = (1/12)*m1*L1^2;
inertia2 = (1/12)*m2*L2^2;
%As stated in the written portion, this is the moment of inertia about the
%center of mass of a thin rod. It can be derived, but its okay to just
%remember it and use it. We create two variables, one for each rod.

KE1 = .5*inertia1*dtheta1^2 + .5*m1*V1a1^2 + .5*m1*V1a2^2;
%This is the Kinetic energy of rod 1. Following the known equation, we add
%half the mass times the velocity squared to half the inertia about its COM
%times the angular velocity squared. First term is inertia, second and third
%term are accounting for each direction of the linear velocity of the center
%of mass.
KE2 = .5*inertia2*dtheta2^2 + .5*m2*V2a1^2 + .5*m2*V2a2^2;
% This is the Kinetic energy of rod 2. It follows the exact same idea as the
```

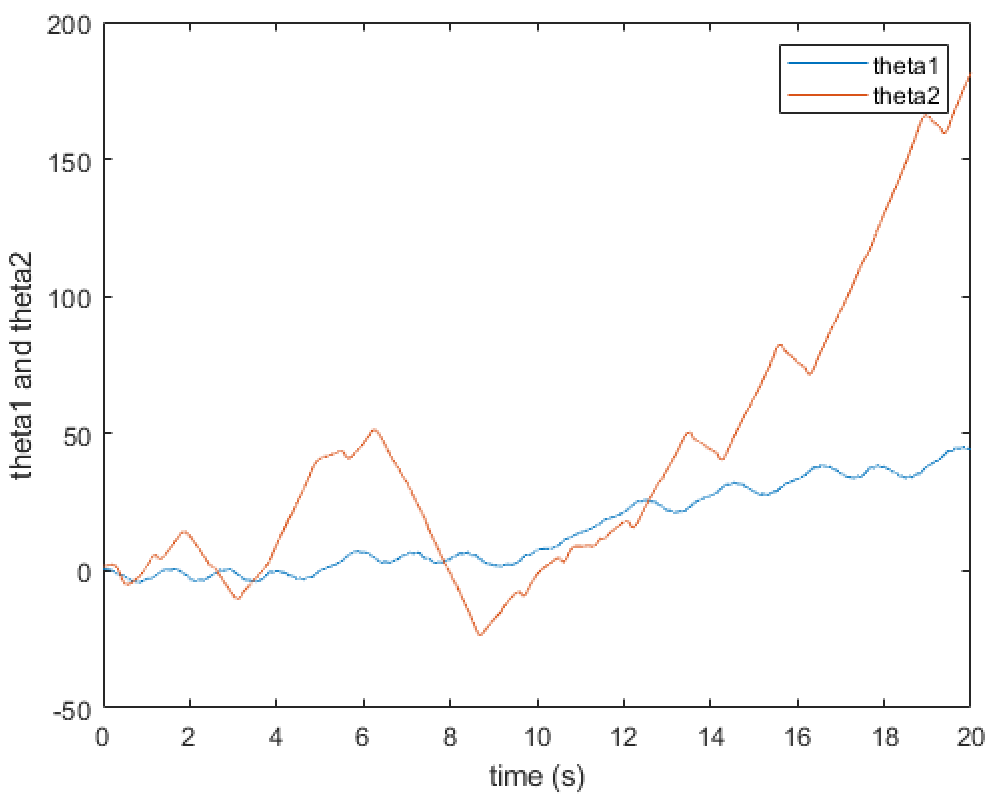
```
%KE of rod one, but obviously accounting for the different mass(m2 instead
%of m1).
KE = KE1 + KE2;
%Creating the total mass by summing all the kinetic energies of our system,
%in this case just the two rods'.

PE = .5*m1*g*L1*sin(theta1) + m2*g*L1*sin(theta1) + .5*m2*g*L2*sin(theta2);
%This is the total potential energy of the system, which is the sum of the
%two potential energies that we have. As shown in the written portion, we
%add the PE of rod 1 with the PE of rod 2 to get our total that we have
%here. The potential energy of a body is the mass times the acceleration due
%to gravity times the height of the center of mass off the "ground" - the
%ground in this case being the "lowest point that the body can go."

Lagrangian = KE-PE;
%Following the definition of what the Lagrangian equation is, L = KE - PE.
%This is going to be used to create an EOM for each of our degrees of
%freedom. So, in this case, two EOMS - one for theta1 and one for theta2.

eom1 = simplify(diff(diff(Lagrangian,dtheta1),t)-diff(Lagrangian,theta1)) ==
0;
eom2 = simplify(diff(diff(Lagrangian,dtheta2),t)-diff(Lagrangian,theta2)) ==
0;
%Here we are plugging in our variables into the larger Euler-Lagrange
%equation that is defined in the written portion. We use this to solve for
%the equations of motion. We take the time derivative of the partial
%derivative of L with respect to each generalized coordinate's first
%derivative(dtheta1 and dheta2), and  subtract the partial of L with respect
%to each generalized coordinate(theta1 and theta2). We set that equal to
%zero because there are no non-conservative forces in our system. If there
%were, we would set everything equal to the sum of forces cross the partials
%of the postion vector of the location of the force with respect to the
%specific generalized coordinate.

disp(eom1)
disp(eom2)
%Display the EOMs so that we can easily copy them. For an easier time doing
%this, we can also use pretty_equation(eom1) and pretty_equation(eom2) to
%make the output more readable.
```

```
(L1^2*m1*diff(theta1(t), t, t))/3 + L1^2*m2*diff(theta1(t), t,
t) + (L1*g*m1*cos(theta1(t)))/2 + L1*g*m2*cos(theta1(t)) +
(L1*L2*m2*sin(theta1(t) - theta2(t))*diff(theta2(t), t)^2)/2 +
(L1*L2*m2*cos(theta1(t) - theta2(t))*diff(theta2(t), t, t))/2 == 0
symbolic function inputs: t

(L2^2*m2*diff(theta2(t), t, t))/3 + (L2*g*m2*cos(theta2(t)))/2
- (L1*L2*m2*sin(theta1(t) - theta2(t))*diff(theta1(t), t)^2)/2 +
(L1*L2*m2*cos(theta1(t) - theta2(t))*diff(theta1(t), t, t))/2 == 0
symbolic function inputs: t
```

Published with MATLAB® R2023b

```
% Michael Grady
% Dynamics Final Project, graphing the EOMS.
clear all
close all
clc

syms m1 m2 L1 L2 g theta1(t) theta2(t)
%Declaring what our symbolic variables are going to be for the
%remainder of the script. We are also establishing theta1 and
%theta2 as our degrees of freedom that change as a function of time.

dtheta1 = diff(theta1,t);
dtheta2 = diff(theta2,t);
ddtheta1 = diff(dtheta1,t);
ddtheta2 = diff(dtheta2,t);
%Creating theta1dot theta2dot theta1doubledot and theta2doubledot
%for ease of use later on in the script. Merely a quality of life
%improvement.

eom1 = L1^2*m1*ddtheta1/3 + L1^2*m2*ddtheta1 + L1*g*m1*cos(theta1)/2 + ...
       L1*g*m2*cos(theta1) + .5*L1*L2*m2*sin(theta1-theta2)*dtheta2^2 + ...
       .5*L1*L2*m2*cos(theta1-theta2)*ddtheta2 == 0;
eom2 = m2*L2^2*ddtheta2/3 + L1*m2*cos(theta1-theta2)*L2*ddtheta1*.5 - ...
       .5*L1*m2*sin(theta1-theta2)*L2*dtheta1^2 + g*m2*cos(theta2)*L2*.5 ==
0;
%Here we are simply defining our EOMs as we have already solved them. We use
%the pretty_equation output from the previous script to plug it in here. And
%again, since there are no non-conservative forces, we set them equa to 0.

eom1 = subs(eom1, [L1 L2 m1 m2 g], [.3 .2 1 .75 9.8]);
eom2 = subs(eom2, [L1 L2 m1 m2 g], [.3 .2 1 .75 9.8]);
%These lines are substituting the numerical answers that the professor
%provided in for each of our symbolic variables so that we can actually
%graph them and get an output.

[newEqs,newVars] = reduceDifferentialOrder([eom1 eom2], [theta1 theta2]);
%This is reducing the order of the differential equations that we provided
%into a set that we can use for our graphs. We start with two second order
%differential equations and reduce them into first orders that are easier to
%solve - newEqs.

[MM,F] = massMatrixForm(newEqs,newVars);
%This is converting the first order diff eqs that we just got from the
%previous line into a mass matrix multiplied by the vector of forces. MM is
%the coefficients of ddtheta1 and ddtheta2, and the force vector is
%everything else from the right-hand side.

f = MM\F;
%Gets the rates of change of the dtheta1 and dtheta2 with respect to
%time.(acceleration) (ddtheta1 and ddtheta2)

odefun = odeFunction(f,newVars);
%odeFunction converts our systems of symbolic expressions into a function
```

```
%that can be used as the input for ode45.

simulationTime = 20; %in seconds
%how long the simulation runs for

[tout, yout] = ode45(odefun, 0:.001:simulationTime, [pi/4 pi/2 0 0]);
%This is the solver of the differential equations, with the input of the
%function of differential equations that we just created, odefun. It
%integrates from 0 over to whatever our simulationTIme is, so in this case,
%20 seconds. The initial conditions  are in the order of newVars, so theta1,
%theta2, dtheta1, then dtheta2.

figure(1)
plot(tout, yout(:,1),tout, yout(:,2))
xlabel('time (s)');
ylabel('theta1 and theta2')
legend('theta1', 'theta2')
%This is just how you plot on Matlab. yout(:,1) represents theta1, and
%yout(:,2) represents theta2. Create the x and y axis labels, and the
%accompanying legend.
```



Published with MATLAB® R2024b