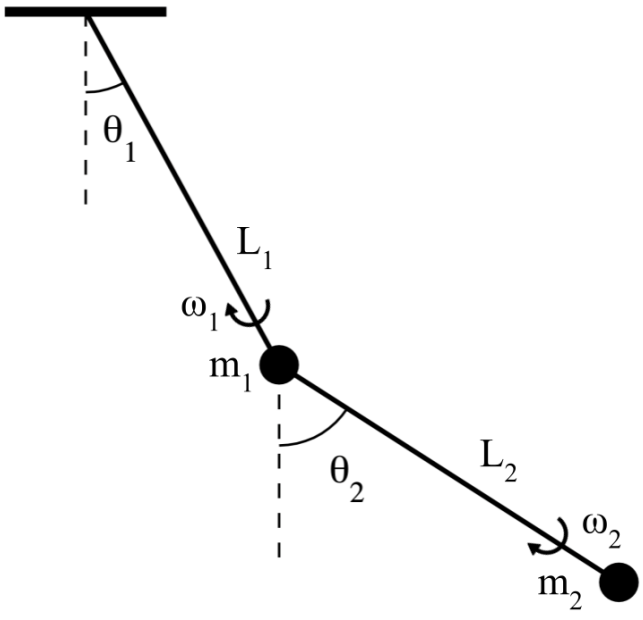
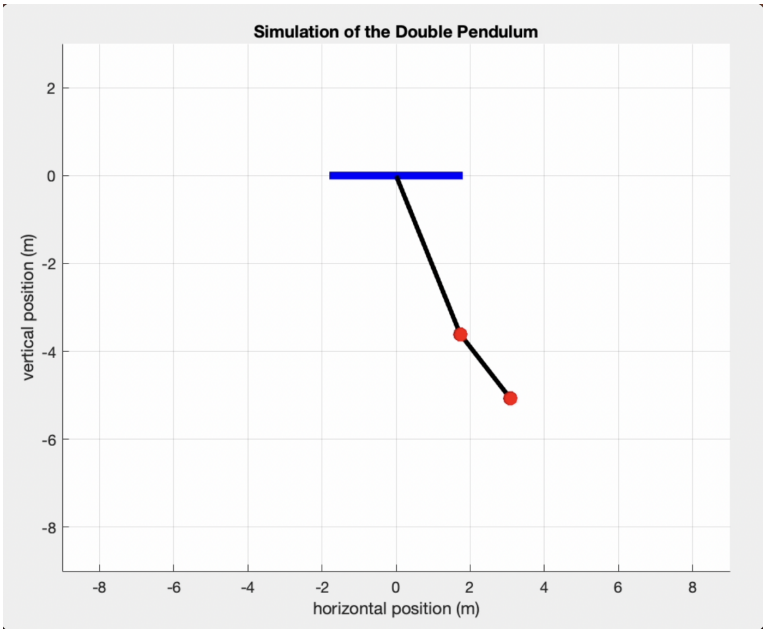
VARDAAN TEKRIWAL

August 2023

1. INTRODUCTION

This project builds a simulation for any closed system containing a free double pendulum (as illustrated below) using numerical methods. The simultaneous equations of motion that we obtain are second order differential equations with respect to the initial angles. We derive the equations, implement the vectorized Runge-Kutta-4 (RK4) method to simulate the physical system (in SI units) using MATLAB, and conduct analysis on the order of convergence through calculating error terms.





(the simulation is animated, a still capture is presented due to formatting restrictions)

Key Words: Differential Equations, Runge-Kutta Methods, Order Reduction, Order of Convergence, Newtonian Kinematics

2. THE SYSTEM OF DIFFERENTIAL EQUATIONS

Either rod may have any real, non-negative length and there exist positive real, non-negative masses at the end of each rod. The closed system may have any gravitational constant. The rods themselves are assumed to be rigid and massless. Either pendulum may have angular displacement and angular velocity.

We thus define the independent variables (i.e. inputs to our simulation):

$L_{1,2} = Lengths\ of\ the\ rods\ (m)$

$\theta_{1,2} = Initial\ angles\ of\ each\ pendulum\ with\ respect\ to\ the\ normal\ (rad)$

$\omega_{1,2} = Angular\ velocities\ of\ the\ point\ masses\ (rad/sec)$

$m_{1,2} = the\ point\ masses\ at\ the\ end\ of\ each\ rod\ (kg)$

$g = the\ gravitational\ constant\ associated\ with\ the\ closed\ system\ (m/s^2)$

Through the derivation provided in **Appendix A**, we obtain the following system:

$$\theta_1'' = \frac{-g(2m_1 + m_2)\sin\theta_1 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 ((\theta_2')^2 L_2 + (\theta_1')^2 L_1 \cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$

$$\theta_2'' = \frac{2\sin(\theta_1 - \theta_2)((\theta_1')^2 L_1(m_1 + m_2) + g(m_1 + m_2)\cos\theta_1 + (\theta_2')^2 L_2 m_2 \cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$

3. ORDER REDUCTION

The physical system has a useful property where $(\theta_{1,2})' = \omega_{1,2}$ since we can reduce the system's order from two to one by doubling the number of equations in the system.

The system of second order differential equations are thus modified into first order ones:

$$\theta_1' = \omega_1$$

$$\theta_2' = \omega_2$$

$$\omega_1' = \frac{-g(2m_1 + m_2)\sin\theta_1 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 (\omega_2^2 + \omega_1^2 \cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$

$$\theta_2'' = \frac{2\sin(\theta_1 - \theta_2)(\omega_1^2 L_1(m_1 + m_2) + g(m_1 + m_2)\cos\theta_1 + \omega_2^2 L_2 m_2 \cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$

4. NUMERICAL IMPLEMENTATION FUNCTIONS

```
function ydot = fpend(y, L, m, g)
%fpend is the system of functions
% y is the input from the previous timestep (variable)
           %i.e. theta (rad) and omega (rad/sec)
% L is the vector of rod lengths (constant, in meters)
% m is the vector of point masses (constant, in kilograms)
% g is the gravitational constant (constant, in meters/second^2)

ydot = [y(3), ...
       y(4), ...
       (-g*(2*m(1)+m(2))*sin(y(1))-m(2)*g*sin(y(1)-2*y(2))-2*sin(y(1) ...
            - y(2))*m(2)*(y(4)^2*L(2) + y(3)^2*L(1)*cos(y(1) - y(2))) ...
             / (L(1)*(2*m(1) + m(2) - m(2)*cos(2*y(1) - 2*y(2)))), ...
       (2*sin(y(1) - y(2))*(y(3)^2 * L(1)*(m(1) + m(2)) + g*(m(1) + ...
            m(2))*cos(y(1)) + y(4)^2*L(2)*m(2)*cos(y(1) - y(2))) ...
            / (L(2)*(2*m(1) + m(2) - m(2)*cos(2*y(1) - 2*y(2))))];
end

function w = pendrk4(a, b, alpha, L, m, g, h)
%a is the initial time step
%b is the final time step
```

```
%alpha is the vector of initial conditions (thetas (rad), omegas (rad/sec))
% L is the vector of rod lengths (constant, in meters)
% m is the vector of point masses (constant, in kilograms)
% g is the gravitational constant (constant, in meters/second^2)
% h is the delta per time step

t = (a:h:b)';
N = (b - a)/(h);
w = zeros(N+1, length(alpha));
w(1,:) = alpha;
for i = 1:N
   k1 = h * fpend(w(i,:), L, m, g);
   k2 = h * fpend(w(i,:) + k1/2, L, m, g);
   k3 = h * fpend(w(i,:) + k2/2, L, m, g);
   k4 = h * fpend(w(i,:) + k3, L, m, g);

   w(i+1, :) = w(i,:) + (1/6) .* (k1 + 2*k2 + 2*k3 + k4);
end
```

5. MATLAB SIMULATION FUNCTION

```
function pendsim(a, b, y, L, m, g, h)
%a is the initial time step
%b is the final time step
%alpha is the vector of initial conditions (i.e thetas (rad), omegas (rad/sec))
% L is the vector of rod lengths (constant, in meters)
% m is the vector of point masses (constant, in kilograms)
% g is the gravitational constant (constant, in meters/second^2)
% h is the delta per time step
%in case input y only consists of initial thetas

alpha = zeros(1, 4);

for i=1:length(y)
   if length(y) > length(alpha)
       return
   end
```

```
   alpha(i) = y(i);
end

%solving the system of differential equation
w = pendrk4(a, b, alpha, L, m, g, h);

%extracting the angles at time steps
th1 = w(:, 1);
th2 = w(:, 2);

%converting angles to cartesian coordinates
L_sum = sum(L);
x1 = L(1)*sin(th1);
y1 = -L(1)*cos(th1);
x2 = x1 + L(2)*sin(th2);
y2 = y1 - L(2)*cos(th2);

%Plotting the data
nth1 = numel(w(:, 1));
nth2 = numel(w(:, 1));

if all([nth1,nth2] > 1)
   if nth1 ~= nth2, error('Vector inputs must be equal lengths'); end
   for i = 1:nth1
       hold off
       cla, grid on,
       axis([-1.5*L_sum,1.5*L_sum,-1.5*L_sum, 0.5*L_sum])
       xlabel("horizontal position (m)"), ylabel("vertical position (m)")
       title("Simulation of the Double Pendulum")
       line([-.3*L_sum,.3*L_sum], [0,0], 'linewidth',5, 'color','b')
       line([0,x1(i),x2(i)], [0,y1(i),y2(i)], 'linewidth',3, 'color','k')
       line([x1(i),x2(i)], [y1(i),y2(i)], 'linestyle','none','marker','.', ...
       'markersize',32, 'color','r')
       pause(h)
   end
   return
end
```

6. EXAMPLE SIMULATION

We select a time frame from 0 to 5 seconds with a time-delta of 0.1, with our initial conditions:

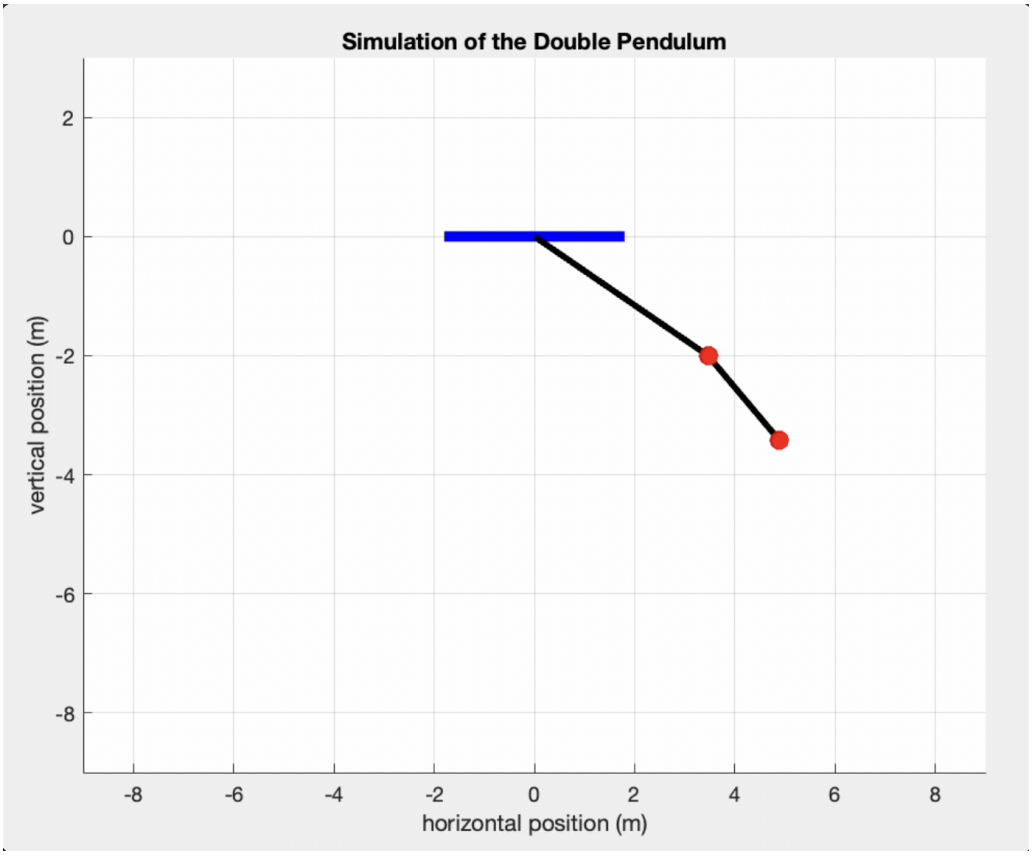
$\theta_1, \theta_2, \omega_1, \omega_2 = (\frac{\pi}{3}, \frac{\pi}{4}, 0, 0)$

$L_1, L_2 = (4, 2)$

$m_1, m_2 = (1, 3)$

$g = 9.81$

```
>> pendsim(0, 5, [pi/3, pi/4, 0, 0], [4, 2], [1, 3], 9.81, 0.1)
```



(Still capture of the initial frame
of the simulation)

7. ORDER OF CONVERGENCE

Given the case that all independent variables of the system are 1 except for the angular velocities, ω, (which is set to 0), the order of convergence can be estimated by analyzing the error at different time-deltas, h. The set of time-deltas is selected strategically to provide the estimated local truncation error on a log-log plot of the errors and h. Starting with $h = 0.001$, the five time steps include $h = \frac{0.05}{2^{(k-1)}}$ for $k = 1, 2, 3, 4$. The time scale runs from $[0, 100]$.

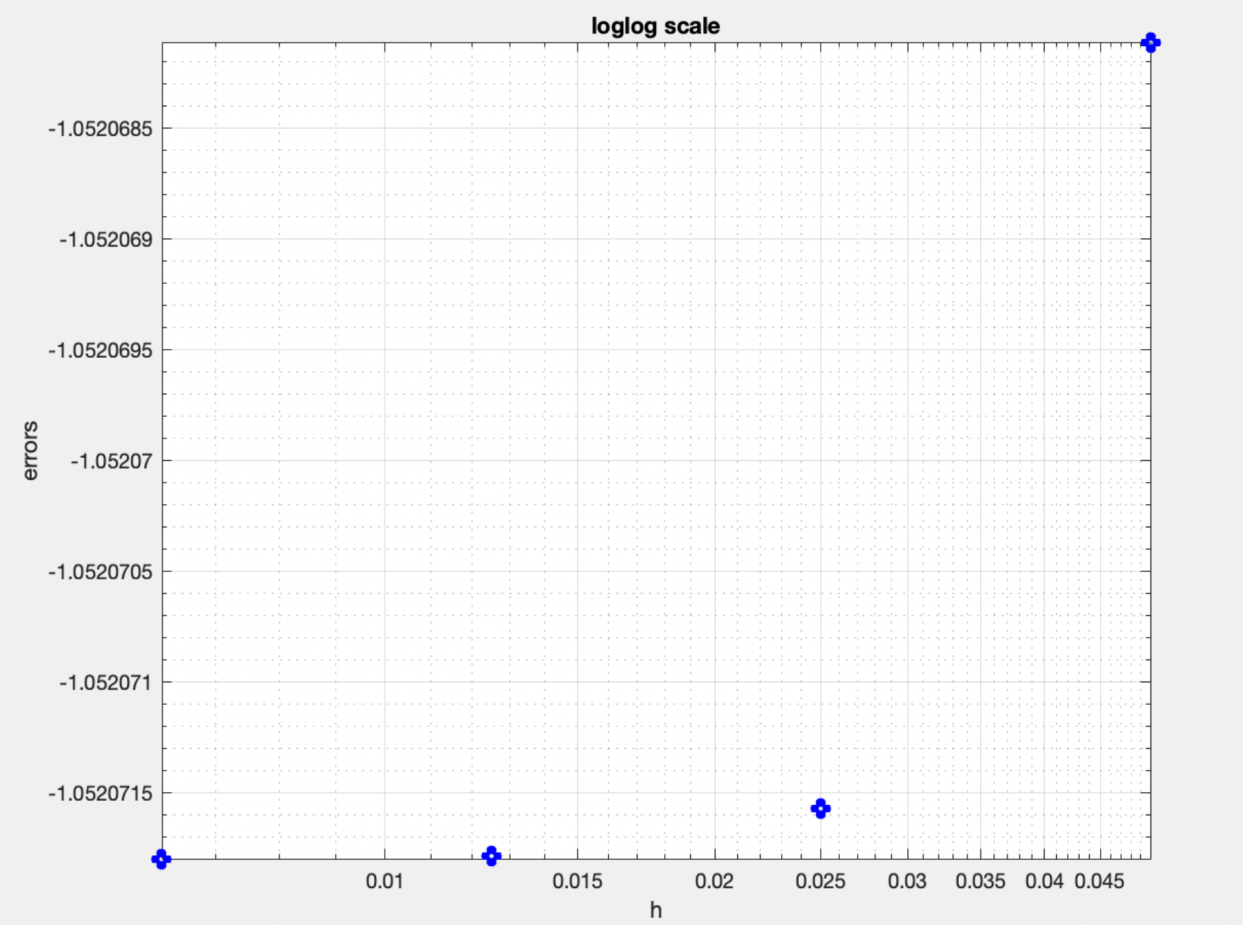
The following is the function and selected output:

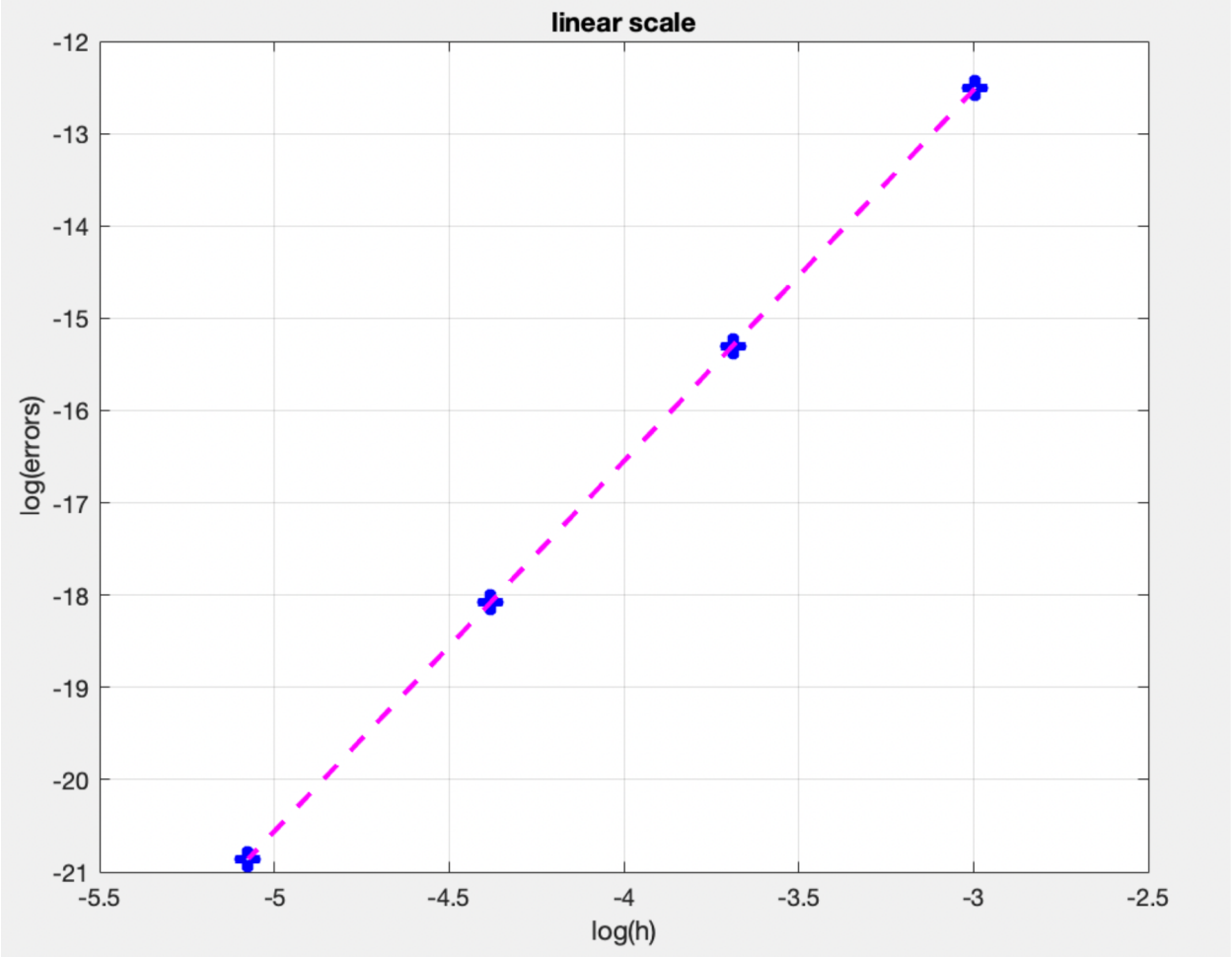
```
function [vals, exact, err, hs, slope] = error_analysis(a, b, alpha, L, m, g)
%a is the initial time step
%b is the final time step
%alpha is the vector of initial conditions (i.e thetas (rad), omegas (rad/sec))
% L is the vector of rod lengths (constant, in meters)
% m is the vector of point masses (constant, in kilograms)
% g is the gravitational constant (constant, in meters/second^2)

w_e = pendrk4(a, b, alpha, L, m, g, 0.001);
exact = w_e(length(w_e),2);
vals = zeros(1, 4);
hs = zeros(1, 4);
for k = 1:4
   w_k = pendrk4(a, b, alpha, L, m, g, 0.05/(2^(k-1)));
   vals(k) = w_k(length(w_k),2);
   hs(k) = 0.05/(2^(k-1));
end
err = vals - exact;
figure(1)
hold off, grid on
loglog(hs, vals, 'ob', 'LineWidth', 4)
xlabel('h'); ylabel('errors')
title('loglog scale')
figure(2)
hold on, grid on
```

```
plot(log(hs), log(err), 'ob', 'LineWidth', 5)
xlabel('log(h)')
ylabel('log(errors)')
title('linear scale')
plot(log(hs), log(err), '--m', 'LineWidth', 3)
xlabel('log(h)')
ylabel('log(errors)')
title('linear scale')

logerrors = log(err);
loghs = log(hs);
slope = (logerrors(4) - logerrors(1))/(loghs(4) - loghs(1));
```






```
slope =

    4.0154
```

Since the slope of the plot on the linear scale is approximately 4, we can conclude that the order of convergence is of $O(h^4)$. This matches the theoretical local truncation error of the RK4 numerical method, which also is $O(h^4)$.

APPENDIX A: THE DERIVATION OF THE EQUATIONS OF MOTION

Other variables that will help with the derivation of the system of differential equations:

$x_{1,2} = \ horizontal\ coordinates\ of\ the\ point\ masses\ (m)$

$y_{1,2} = vertical\ coordinates\ of\ the\ point\ masses\ (m)$

$T_{1,2} = the\ tensile\ force\ in\ each\ rod\ (N)$

$x_1 = L_1 \sin\theta_1$

$y_1 = -L_1 \cos\theta_1$

$x_2 = x_1 + L_1 \sin\theta_1$

$y_2 = y_1 - L_1 \cos\theta_1$

$x_1' = \theta_1' L_1 \cos\theta_1$

$y_1' = \theta_1' L_1 \sin\theta_1$

$x_2' = x_1' + \theta_2 L_2 \cos\theta_2$

$y' = y_1' + \theta_2 L_2 \sin\theta_2$

$x_1'' = -(\theta_1')^2 L_1 \sin\theta_1 + \theta_1'' L_1 \cos\theta_1$

$y_1'' = (\theta_1')^2 L_1 \cos\theta_1 + \theta_1'' L_1 \sin\theta_1$

$x_2'' = x_1'' - (\theta_2')^2 L_2 \sin\theta_2 + \theta_2'' L_2 \cos\theta_2$

$y_2'' = (\theta_2')^2 L_2 \cos\theta_2 + \theta_2'' L_2 \sin\theta_2$

$m_1 x_1'' = -T_1 \sin\theta_1 + T_2 \sin\theta_2$

$m_1 y_1'' = T_1 \cos\theta_1 - T_2 \cos\theta_2 - m_1 g$

$m_2 x_2'' = -T_2 \sin\theta_2$

$m_2 y_2'' = T_2 \cos\theta_2 - m_2 g$

Substituting and Rearranging (using Wolfram):

$$\theta_1'' = \frac{-g(2m_1 + m_2)\sin\theta_2 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 ((\theta_2')^2 L_2 + (\theta_1')^2 L_1 \cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$

$$\theta_2'' = \frac{2\sin(\theta_1 - \theta_2)((\theta_1')^2 L_1(m_1 + m_2) + g(m_1 + m_2)\cos\theta_1 + (\theta_2')^2 L_2 m_2 \cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)}$$