A DOUBLE PENDULUM SIMULATOR WITHIN A UNIFORM MAGNETIC FIELD

CHAD MUSICK

1. Mathematics of the double pendulum without the magnetic field

An idealized double pendulum is a system of two stiff massless rods of fixed length attached end-to-end at a frictionless joint with the free end of one of the rods attached to a stationary frictionless joint and operating under the effects of gravity alone. Figure 1 displays a labeled diagram of such a system.

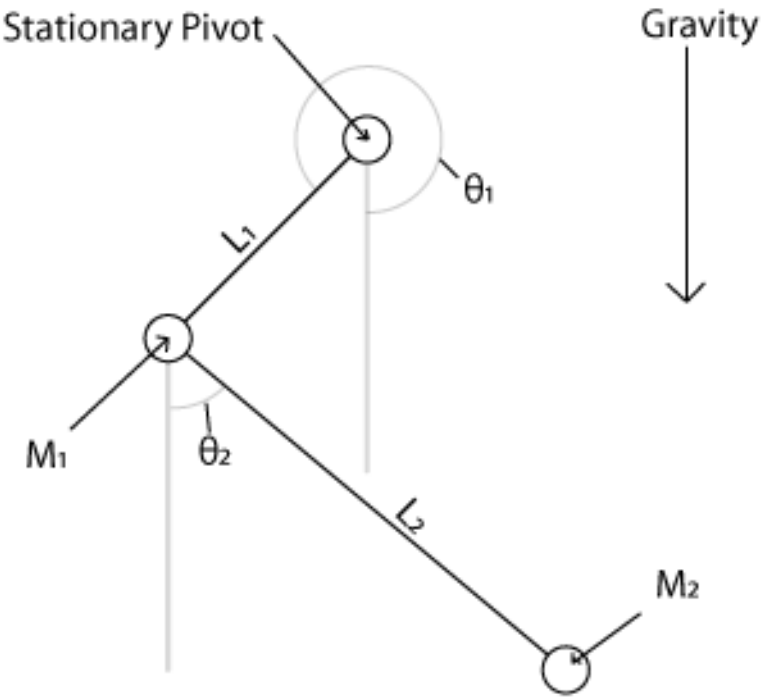

Figure 1. An idealized double pendulum

If the stationary pivot is placed at the origin, then the locations of the point masses M_1 and M_2 are given by

$$M_1 = (L_1 \sin\theta_1, -L_1 \cos\theta_1)$$
$$M_2 = M_1 + (L_2 \sin\theta_2, -L_2 \cos\theta_2).$$

If the masses of M_1 and M_2 are m_1 and m_2 respectively and gravity by g (with units appropriate to the units of the lengths and masses), then the system is modeled by the differential equations

$$\ddot{\theta}_1 = \frac{-g(2m_1 + m_2)\sin\theta_1 - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 (\dot{\theta}_2^2 L_2 - \dot{\theta}_1^2 L_1 \cos(\theta_1 - \theta_2))}{L_1(2m_1 + m_2 - m_2 \cos(2(\theta_1 - \theta_2))}$$

$$\ddot{\theta}_2 = \frac{2\sin(\theta_1 - \theta_2)(\dot{\theta}_1^2 L_1 (m_1 + m_2) + g(m_1 + m_2)\cos\theta_1 + \dot{\theta}_2^2 L_2 m_2 \cos(\theta_1 - \theta_2))}{L_2(2m_1 + m_2 - m_2 \cos(2(\theta_1 - \theta_2)))}$$

This can be solved numerically to find θ_1 and θ_2 as a function of time.

2. The addition of magnetic force

Suppose that the system described above is operating in a uniform magnetic field, such as one generated by an infinitely long wire with steady current in a solenoid around the system. If the mass M_2 carries a point charge, then this mass will, when moving, be subject to a force due to the magnetic field. Suppose that the direction of the current through the wire is such that the force is in the clockwise direction and that it is centered at the stationary pivot. Figure 2 shows such a figure with the magnetic force lines displayed in red and the point charge in blue.

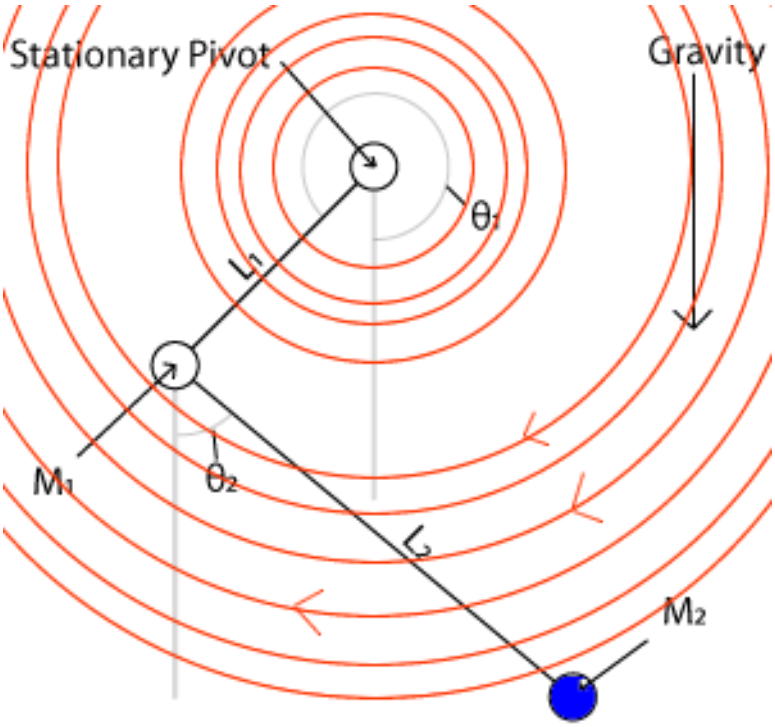

Figure 2. An idealized double pendulum with magnetic force lines

We can choose a unit of measurement such that the strength of the magnetic field is 1. Then by Lorenz's law,

$$F_m = P\left(\begin{pmatrix}\dot{M}_{2,x}\\ \dot{M}_{2,y}\\ 0\end{pmatrix}\times\begin{pmatrix}0\\0\\1\end{pmatrix}\right) = P\left(\begin{pmatrix}\dot{M}_{2,y}\\ -\dot{M}_{2,x}\\ 0\end{pmatrix}\right)$$

where P is the charge on M_2 in some appropriate unit and $\dot{M}_{2,x}, \dot{M}_{2,y}$ are the instantaneous velocities of M_2 in the x and y direction respectively. From Newton's second law, the acceleration of M_2 due to the field is $F_{magnetic}/m_2$, the mass of M_2.

The linear velocity of M_2 may be calculated as the sum of the linear velocity due to $\dot{\theta}_1$ and the linear velocity due to $\dot{\theta}_2$. So that:

$$\begin{aligned}\dot{M}_{2,x} &= 2\pi\sqrt{(L_1\sin(\theta_1)+L_2\sin(\theta_2))^2+(L_1\cos(\theta_1)+L_2\cos(\theta_2))^2}*\cos(\theta_1)*\dot{\theta}_1\\ &\quad + L_2*\cos(\theta_2)*\dot{\theta_2}\\ \dot{M}_{2,y} &= 2\pi\sqrt{(L_1\sin(\theta_1)+L_2\sin(\theta_2))^2+(L_1\cos(\theta_1)+L_2\cos(\theta_2))^2}*\sin(\theta_1)*\dot{\theta}_1\\ &\quad + L_2*\sin(\theta_2)*\dot{\theta_2}\end{aligned}$$

We modify our prior equations to account for this new force. Letting $L_{O,2}$ be the distance from the stationary pivot to M_2, we can modify our prior equations to account for a point charge of magnitude 1:

$$\begin{aligned}\ddot{\theta}_1 &= \frac{-g(2m_1+m_2)\sin\theta_1 - m_2 g\sin(\theta_1-2\theta_2) - 2\sin(\theta_1-\theta_2)m_2(\dot{\theta}_2^2L_2-\dot{\theta}_1^2L_1\cos(\theta_1-\theta_2))}{L_1(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2))}\\ &\quad + \frac{\dot{M}_{2,x}\sin\theta_1 + \dot{M}_{2,x}\sin(\theta_1-2\theta_2) + \dot{M}_{2,y}\cos\theta_1 + \dot{M}_{2,y}\cos(\theta_1-2\theta_2)}{L_1(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2))}\\ \ddot{\theta}_2 &= \frac{2\sin(\theta_1-\theta_2)(\dot{\theta}_1^2L_1(m_1+m_2)+g(m_1+m_2)\cos\theta_1+\dot{\theta}_2^2L_2m_2\cos(\theta_1-\theta_2))}{L_2(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))}\\ &\quad + \frac{2\sin(\theta_1-\theta_2)(-\dot{M}_{2,y}\cos\theta_1)+2\cos(\theta_1-\theta_2)(\dot{M}_{2,x}\sin\theta_1)}{L_2(2m_1+m_2-m_2\cos(2(\theta_1-\theta_2)))}\end{aligned}$$

3. The Effects of the Changes

A series of figures generated by the included source code demonstrate clearly one of the primary effects that the magnetic field has on the motion of the double pendulum — the field inhibits the natural swing of the pendulum as the lower bob tries to drift in a circular pattern, a phenomena called ion cyclotron resonance.

Figure 3 is the initial state of the system — both the gravity only and gravity/magnetism states are identical. Figure 4 shows the two systems before much difference has occurred, since the direction of motion has been almost entirely with the field, little change has occurred. Figure 5 shows the system after 10 swings of the pendulum. Drift is clearly visible in the magnetized system.

In the simulator, the times of the two figures are locked so that the pendulums always represent the same time after the same initial system. It is therefore possible to visually compare the positions of the bobs to see the net effect that the field has had on the system.

4. The MATLAB Code

```
function varargout = magpendulum(varargin)
% pendulum Plots a double pendulum with adjustable values for all
%       state values.

% These data items are necessary for all components to know since they
% describe the state of the system and store past data values (for
% graphing).
mL              = 500; % The length of the storage vectors.
```

```
if nargin > 0 % Modify this length if requested.
    mL = max([1, round(str2double(varargin(1)))]);
end
mOutputArgs     = {}; % Storage for output when GUI returns
mTheta1         = zeros(1, mL) + rand()*pi; % The previous mL values
mTheta2         = zeros(1, mL) + rand()*pi; % The previous mL values
mPivotsX        = zeros(1, mL); % The last mL X positions of the pivot
mPivotsY        = zeros(1, mL); % The last mL Y positions of the pivot
mPivot          = NaN; % The line handle of the inner pivot.
mBobsX          = zeros(1, mL); % The last mL X positions of the bob
mBobsY          = zeros(1, mL); % The last mL Y positions of the bob
mBob            = NaN; % The line handle of the outer mass (the bob).
mTrail          = NaN; % The line handle of the path trail.
mLinks          = NaN; % The line handle for the origin-pivot-bob links.
mThetaSpeed1    = rand()*6-3; % The value of theta 1 speed
mThetaSpeed2    = rand()*6-3; % The value of theta 2 speed
mATheta1         = mTheta1; % The previous mL values
mATheta2         = mTheta2; % The previous mL values
mAPivotsX        = zeros(1, mL); % The last mL X positions of the pivot
mAPivotsY        = zeros(1, mL); % The last mL Y positions of the pivot
mAPivot          = NaN; % The line handle of the inner pivot.
mABobsX          = zeros(1, mL); % The last mL X positions of the bob
mABobsY          = zeros(1, mL); % The last mL Y positions of the bob
mABob            = NaN; % The line handle of the outer mass (the bob).
mATrail          = NaN; % The line handle of the path trail.
mALinks          = NaN; % The line handle for the origin-pivot-bob links.
mRevolve        = 1; % The index of the circular index of the vectors.
mAThetaSpeed1    = mThetaSpeed1; % The value of theta 1 speed
mAThetaSpeed2    = mThetaSpeed2; % The value of theta 2 speed
mMass1          = rand()*3+1; % The mass of object 1
mMass2          = rand()*3+1; % The mass of object 2
mGravity        = rand()*10+5; % The gravitational constant.
mLength1        = rand()+0.1; % The length of pendulum 1.
mLength2        = 2-mLength1; % The length of pendulum 2.
mRunning        = 0; % Whether or not the simulation is running.
mStop           = 0; % Should this stop?

% The UI objects.
hMainFigure     = figure(...    % The overall window
                    'Units', 'pixels', ...
                    'Position', [100, 100, 1200, 700], ...
                    'MenuBar', 'figure', ...
                    'HandleVisibility', 'callback', ...
                    'Name', 'The Double Pendulum', ...
                    'NumberTitle', 'off', ...
                    'Color', get(0,'defaultuicontrolbackgroundcolor'));
hPlotAxes       = axes(...      % The axes for displaying the pendulum
                    'Parent', hMainFigure, ...
                    'Units', 'normalized', ...
                    'HandleVisibility', 'callback', ...
                    'GridLineStyle', 'none', ...
                    'XTickLabel', '', ...
                    'XTick', [], ...
```

```
                    'YTickLabel', '', ...
                    'YTick', [], ...
                    'Box', 'on', ...
                    'XLim', [-2.2, 2.2], ...
                    'YLim', [-2.2, 2.2], ...
                    'Position', [0.05, 0.05, 0.35, 0.75]);
hAxesText       = uicontrol(..., % The text for the regular axes
                    'Parent', hMainFigure, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.10, 0.01, 0.2, 0.03], ...
                    'Style', 'text', ...
                    'String', 'Gravity Alone');
hMPlotAxes       = axes(...      % The axes for displaying mag pendulum
                    'Parent', hMainFigure, ...
                    'Units', 'normalized', ...
                    'HandleVisibility', 'callback', ...
                    'GridLineStyle', 'none', ...
                    'XTickLabel', '', ...
                    'XTick', [], ...
                    'YTickLabel', '', ...
                    'YTick', [], ...
                    'Box', 'on', ...
                    'XLim', [-2.2, 2.2], ...
                    'YLim', [-2.2, 2.2], ...
                    'Position', [0.45, 0.05, 0.35, 0.75]);
hMAxesText       = uicontrol(..., % The text for the magnetic field axes
                    'Parent', hMainFigure, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.52, 0.01, 0.2, 0.03], ...
                    'Style', 'text', ...
                    'String', 'Gravity and Magnetism');
hRunButton      = uicontrol(..., % The button to stop/start the plotting
                    'Parent', hMainFigure, ...
                    'Units', 'normalized', ...
                    'Position', [0.82, 0.05, 0.16, 0.13], ...
                    'FontSize', 16, ...
                    'HandleVisibility', 'callback', ...
                    'String', 'Run', ...
                    'Style', 'togglebutton', ...
                    'TooltipString', 'Push to swing pendulums', ...
                    'Callback', @hRunButtonCallback);
hConstPanel     = uipanel(..., % A panel to contain the constants.
                    'Parent', hMainFigure, ...
                    'FontSize', 12, ...
                    'Title', 'Constants', ...
                    'TitlePosition', 'centertop', ...
                    'BorderType', 'beveledin', ...
                    'Position', [0.05, 0.82, 0.75, 0.16]);
hGravityText    = uicontrol(..., % The text for gravity
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
```

```
                    'FontSize', 14, ...
                    'Position', [0.05, 0.6, 0.2, 0.35], ...
                    'Style', 'text', ...
                    'String', 'Gravity');
hGravity        = uicontrol(..., % The control to set/show gravity
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.05, 0.05, 0.2, 0.55], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mGravity), ...
                    'TooltipString', 'The gravitational constant', ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hGravityCallback);
hPendulumText   = uicontrol(..., % The text for the pendulum length
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.29, 0.6, 0.2, 0.35], ...
                    'Style', 'text', ...
                    'String', 'Pendulum Length');
hPendulum       = uicontrol(..., % The control to set/show pendulum length
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.29, 0.05, 0.2, 0.55], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mLength2 / mLength1), ...
                    'TooltipString', ['The length of the outer ',...
                        'pendulum, relative to the inner'], ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hPendulumCallback);
hMass1Text      = uicontrol(..., % The text for the mass 1
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.53, 0.6, 0.2, 0.35], ...
                    'Style', 'text', ...
                    'String', 'Mass 1');
hMass1          = uicontrol(..., % The control to set/show mass 1
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.53, 0.05, 0.2, 0.55], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mMass1), ...
                    'TooltipString', 'The mass of the inner body', ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hMass1Callback);
hMass2Text      = uicontrol(..., % The text for the mass 2
```

```
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.77, 0.6, 0.2, 0.35], ...
                    'Style', 'text', ...
                    'String', 'Mass 2');
hMass2          = uicontrol(..., % The control to set/show mass 1
                    'Parent', hConstPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.77, 0.05, 0.2, 0.55], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mMass2), ...
                    'TooltipString', 'The mass of the outer body', ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hMass2Callback);
hVolPanel       = uipanel(..., % A panel to contain the constants.
                    'Parent', hMainFigure, ...
                    'FontSize', 12, ...
                    'Title', 'State', ...
                    'TitlePosition', 'centertop', ...
                    'BorderType', 'beveledin', ...
                    'Position', [0.82, 0.20, 0.16, 0.78]);
hTheta1Text     = uicontrol(..., % The text for Theta 1
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.05, 0.8, 0.9, 0.18], ...
                    'Style', 'text', ...
                    'String', 'Theta 1');
hTheta1         = uicontrol(..., % The control to set/show Theta 1
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.05, 0.75, 0.9, 0.18], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mTheta1(mRevolve)), ...
                    'TooltipString', 'Angle of the inner pendulum', ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hTheta1Callback);
hThetaSpd1Text  = uicontrol(..., % The text for Theta 1 Speed
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.05, 0.56, 0.9, 0.18], ...
                    'Style', 'text', ...
                    'String', 'Inner Speed');
hTheta1Spd      = uicontrol(..., % The control to set/show Theta 1 Speed
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.05, 0.51, 0.9, 0.18], ...
```

```
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mThetaSpeed1), ...
                    'TooltipString', ['Angular velocity of the ', ...
                        'inner pendulum'], ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hTheta1SpdCallback);
hTheta2Text     = uicontrol(..., % The text for Theta 2
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.05, 0.32, 0.9, 0.18], ...
                    'Style', 'text', ...
                    'String', 'Theta 2');
hTheta2         = uicontrol(..., % The control to set/show Theta 2
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.05, 0.27, 0.9, 0.18], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mTheta2(mRevolve)), ...
                    'TooltipString', 'Angle of the outer pendulum', ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hTheta2Callback);
hThetaSpd2Text  = uicontrol(..., % The text for Theta 2 Speed
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'FontSize', 14, ...
                    'Position', [0.05, 0.08, 0.9, 0.18], ...
                    'Style', 'text', ...
                    'String', 'Outer Speed');
hTheta2Spd      = uicontrol(..., % The control to set/show Theta 2 Speed
                    'Parent', hVolPanel, ...
                    'Units', 'normalized', ...
                    'Position', [0.05, 0.03, 0.9, 0.18], ...
                    'HandleVisibility', 'callback', ...
                    'Style', 'edit', ...
                    'FontSize', 16, ...
                    'String', num2str(mThetaSpeed2), ...
                    'TooltipString', ['Angular velocity of the ', ...
                        'outer pendulum'], ...
                    'BackgroundColor', 'w', ...
                    'Callback', @hTheta2SpdCallback);

% Define default output and return it if it is requested by users
mOutputArgs{1} = hMainFigure;
if nargout>0
    [varargout{1:nargout}] = mOutputArgs{:};
end

initDrawPendulum();
```

```
%-------------------------------------------------------------------------
function hRunButtonCallback(hObject, eventdata)
% Callback function run when the run button is toggled.
    press = get(hObject, 'Value');
    if press == get(hObject, 'Max') % Toggled On
        set(hObject, 'String', 'Pause');
        set(hObject, 'TooltipString', 'Push to stop pendulums');
        stepPendulum();
    else % Toggled Off
        set(hObject, 'String', 'Run');
        set(hObject, 'TooltipString', 'Push to swing pendulums');
        mStop = 1;
    end
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hGravityCallback(hObject, eventdata)
% Callback function run when the gravity is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d <= 0
        errordlg('Only positive numeric values', 'Bad Input', 'modal')
        set(hObject, 'String', num2str(mGravity));
        return
    end
    mGravity = d;
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hPendulumCallback(hObject, eventdata)
% Callback function run when the pendulum length is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d <= 0
        errordlg('Only positive numeric values', 'Bad Input', 'modal')
        set(hObject, 'String', num2str(mLength1 / mLength2));
        return
    end
    % The total length of the pendulums is 2. This sets relative
    % length.
    mLength2 = d*mLength1;
    scale = (mLength2 + mLength1) / 2;
    mLength1 = mLength1 / scale;
    mLength2 = mLength2 / scale;
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hMass1Callback(hObject, eventdata)
% Callback function run when mass 1 is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d <= 0
```

```
        errordlg('Only positive numeric values', 'Bad Input', 'modal')
        set(hObject, 'String', num2str(mMass1));
        return
    end
    mMass1 = d;
    % Make the point size reflect the mass (within a limited extent).
    set(mPivot, 'linewidth', max([1, min([6, mMass1])]));
    set(mAPivot, 'linewidth', max([1, min([6, mMass1])]));
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hMass2Callback(hObject, eventdata)
% Callback function run when mass 2 is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d ≤ 0
        errordlg('Only positive numeric values', 'Bad Input', 'modal')
        set(hObject, 'String', num2str(mMass2));
        return
    end
    mMass2 = d;
    % Make the point size reflect the mass (within a limited extent).
    set(mBob, 'linewidth', max([1, min([6, mMass2])]));
    set(mABob, 'linewidth', max([1, min([6, mMass2])]));
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hTheta1Callback(hObject, eventdata)
% Callback function run when theta 1 is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d > 2*pi || d < 0
        errordlg('Only numeric values between 0 and 2pi', ...
            'Bad Input', 'modal')
        set(hObject, 'String', num2str(mTheta1(mRevolve)));
        return
    end
    mTheta1(mRevolve) = d;
    mATheta1(mRevolve) = d;
    refreshDrawing();
end

%-------------------------------------------------------------------------
function hTheta1SpdCallback(hObject, eventdata)
% Callback function run when theta 1's speed is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d)
        errordlg('Only numeric values', ...
            'Bad Input', 'modal')
        set(hObject, 'String', num2str(mThetaSpeed1));
        return
    end
    mThetaSpeed1 = d;
```

```
    mAThetaSpeed1 = d;
    refreshDrawing();
end

%--------------------------------------------------------------------------
function hTheta2Callback(hObject, eventdata)
% Callback function run when theta 2 is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d) || d > 2*pi || d < 0
        errordlg('Only numeric values between 0 and 2pi', ...
            'Bad Input', 'modal')
        set(hObject, 'String', num2str(mTheta2(mRevolve)));
        return
    end
    mTheta2(mRevolve) = d;
    mATheta2(mRevolve) = d;
    refreshDrawing();
end

%--------------------------------------------------------------------------
function hTheta2SpdCallback(hObject, eventdata)
% Callback function run when theta 2's speed is updated.
    d = str2double(get(hObject, 'String'));
    if isnan(d)
        errordlg('Only numeric values', ...
            'Bad Input', 'modal')
        set(hObject, 'String', num2str(mThetaSpeed2));
        return
    end
    mThetaSpeed2 = d;
    mAThetaSpeed2 = d;
    refreshDrawing();
end

%--------------------------------------------------------------------------
function [pivot, bob, pivotM, bobM] = thetaToPositions(N)
% Obtain the positions of the pendulum ends given the thetas.
    pivot = [mLength1*sin(mTheta1(N)), -mLength1*cos(mTheta1(N))];
    bob = [mLength2*sin(mTheta2(N)), -mLength2*cos(mTheta2(N))];
    bob = bob + pivot;
    pivotM = [mLength1*sin(mATheta1(N)), -mLength1*cos(mATheta1(N))];
    bobM = [mLength2*sin(mATheta2(N)), -mLength2*cos(mATheta2(N))];
    bobM = bobM + pivotM;
end

%--------------------------------------------------------------------------
function refreshDrawing()
% Refresh the drawing with the positions of the pivot and bob.
    [pivot, bob, pivotM, bobM] = thetaToPositions(mRevolve);
    mPivotsX(mRevolve) = pivot(1);
    mPivotsY(mRevolve) = pivot(2);
    mBobsX(mRevolve) = bob(1);
    mBobsY(mRevolve) = bob(2);
```

```
    mAPivotsX(mRevolve) = pivotM(1);
    mAPivotsY(mRevolve) = pivotM(2);
    mABobsX(mRevolve) = bobM(1);
    mABobsY(mRevolve) = bobM(2);
    set(mPivot, 'xdata', mPivotsX(mRevolve), ...
        'ydata', mPivotsY(mRevolve));
    set(mBob, 'xdata', mBobsX(mRevolve), 'ydata', mBobsY(mRevolve));
    set(mAPivot, 'xdata', mAPivotsX(mRevolve), ...
        'ydata', mAPivotsY(mRevolve));
    set(mABob, 'xdata', mABobsX(mRevolve), 'ydata', mABobsY(mRevolve));

    % Use the array of positions as a circular list.
    intmdt = mRevolve;
    lst = length(mBobsX);
    if intmdt == lst
        bobX = mBobsX;
        bobY = mBobsY;
        bobXM = mABobsX;
        bobYM = mABobsY;
    else
        bobX = cat(2, mBobsX(intmdt+1:lst), mBobsX(1:intmdt));
        bobY = cat(2, mBobsY(intmdt+1:lst), mBobsY(1:intmdt));
        bobXM = cat(2, mABobsX(intmdt+1:lst), mABobsX(1:intmdt));
        bobYM = cat(2, mABobsY(intmdt+1:lst), mABobsY(1:intmdt));
    end

    % Update the data used for the drawing.
    set(mTrail, 'xdata', bobX, 'ydata', bobY);
    set(mLinks, 'xdata', [0, mPivotsX(mRevolve), mBobsX(mRevolve)],...
        'ydata', [0, mPivotsY(mRevolve), mBobsY(mRevolve)]);
    set(mATrail, 'xdata', bobXM, 'ydata', bobYM);
    set(mALinks, 'xdata', [0, mAPivotsX(mRevolve), mABobsX(mRevolve)],...
        'ydata', [0, mAPivotsY(mRevolve), mABobsY(mRevolve)]);

    set(hTheta1, 'String', sprintf('%s, %s', ...
        num2str(mTheta1(mRevolve)), num2str(mATheta1(mRevolve))));
    set(hTheta2, 'String', sprintf('%s, %s', ...
        num2str(mTheta2(mRevolve)), num2str(mATheta2(mRevolve))));
    set(hTheta1Spd, 'String', sprintf('%s, %s', ...
        num2str(mThetaSpeed1), num2str(mAThetaSpeed1)));
    set(hTheta2Spd, 'String', sprintf('%s, %s', ...
        num2str(mThetaSpeed2), num2str(mAThetaSpeed2)));
    % Force a draw.
    drawnow;
end

%-------------------------------------------------------------------------
function initDrawPendulum()
% Create the handles for the pivot, bob, and trail and set up.
    [pivot, bob, pivotM, bobM] = thetaToPositions(mRevolve);
    line('parent', hPlotAxes, 'marker', '*', 'xdata', 0, ...
        'ydata', 0, 'linewidth', 4, 'color', 'black');
    line('parent', hMPlotAxes, 'marker', '*', 'xdata', 0, ...
```

```
        'ydata', 0, 'linewidth', 4, 'color', 'black');
    mPivotsX = mPivotsX + pivot(1);
    mAPivotsX = mAPivotsX + pivotM(1);
    mPivotsY = mPivotsY + pivot(2);
    mAPivotsY = mAPivotsY + pivotM(2);
    mBobsX = mBobsX + bob(1);
    mABobsX = mABobsX + bobM(1);
    mBobsY = mBobsY + bob(2);
    mABobsY = mABobsY + bobM(2);
    mTrail = line('parent', hPlotAxes, 'color', 'green', ...
        'xdata', mBobsX, 'ydata', mBobsY, 'linewidth', 1.5);
    mATrail = line('parent', hMPlotAxes, 'color', 'green', ...
        'xdata', mABobsX, 'ydata', mABobsY, 'linewidth', 1.5);
    mLinks = line('parent', hPlotAxes, 'color', 'black', ...
        'linewidth', 1, ...
        'xdata', [0,pivot(1),bob(1)], 'ydata', [0,pivot(2),bob(2)]);
    mALinks = line('parent', hMPlotAxes, 'color', 'black', ...
        'linewidth', 1, ...
        'xdata', [0,pivot(1),bobM(1)], 'ydata', [0,pivotM(2),bobM(2)]);
    mPivot = line('parent', hPlotAxes, 'marker', '*', ...
        'linewidth', mMass1, 'color', 'blue', 'xdata', pivot(1), ...
        'ydata', pivot(2));
    mAPivot = line('parent', hMPlotAxes, 'marker', '*', ...
        'linewidth', mMass1, 'color', 'blue', 'xdata', pivotM(1), ...
        'ydata', pivotM(2));
    mBob = line('parent', hPlotAxes, 'marker', '*', ...
        'linewidth', mMass2, 'color', 'red', 'xdata', bob(1), ...
        'ydata', bob(2));
    mABob = line('parent', hMPlotAxes, 'marker', '*', ...
        'linewidth', mMass2, 'color', 'red', 'xdata', bobM(1), ...
        'ydata', bobM(2));
    drawnow;
end

%-------------------------------------------------------------------------
function U = calcDiffs(U, mag)
% Calculate the second derivative (wrt time) of the angular momentum
% of the inner pendulum.
    t1 = U(3);
    t2 = U(4);
    dt1 = U(1);
    dt2 = U(2);
    g = mGravity;
    m1 = mMass1;
    m2 = mMass2;
    L1 = mLength1;
    L2 = mLength2;
    LOB = sqrt((L1 * sin(t1) + L2 * sin(t2))^2 + ...
        (L1 * cos(t1) + L2 * cos(t2))^2);
    M2x = 2 * pi * LOB * cos(t1) * dt1 + L2 * cos(t2) * dt2;
    M2y = 2 * pi * LOB * sin(t1) * dt1 + L2 * sin(t2) * dt2;

    U(3) = U(1); % d/dt Position = Velocity
```

```
    U(4) = U(2);
    % Acceleration equations for pivot (U(1)) and bob (U(2)).
    U(1) = (-g*(2*m1 + m2)*sin(t1) - m2*g*sin(t1 - 2*t2) - ...
        2*sin(t1 - t2)*m2*(dt2^2*L2 - dt1^2*L1*cos(t1 - t2)));
    if mag
        U(1) = U(1) + M2x * sin(t1) + M2x*sin(t1 - 2*t2) + ...
            M2y * cos(t1 - 2*t2);
    end
    U(1) = U(1) / (L1*(2*m1 + m2 - m2*cos(2*(t1 - t2))));
    U(2) = (2*sin(t1-t2)*(dt1^2*L1*(m1+m2)+ g*(m1+m2)*cos(t1) + ...
        dt2^2*L2*m2*cos(t1 - t2)));
    if mag
        U(2) = U(2) + 2 * sin(t1 - t2) * (-M2y * cos(t1)) + ...
            2 * cos(t1 - t2) * (M2x * sin(t1));
    end
    U(2) = U(2) / (L2*(2*m1 + m2 - m2*cos(2*(t1 - t2))));
end

%-------------------------------------------------------------------------
function stepPendulum()
% Iterate one time step forward. A time step of .01 is used.

    if mRunning || mStop
        return;
    end

    mRunning = 1; % Prevent double calculation of time steps.

    while ~mStop % Setting mStop to 1 will cause the loop to terminate.
        ot1 = mTheta1(mRevolve);
        ot2 = mTheta2(mRevolve);
        ots1 = mThetaSpeed1;
        ots2 = mThetaSpeed2;
        otM1 = mATheta1(mRevolve);
        otM2 = mATheta2(mRevolve);
        otsM1 = mAThetaSpeed1;
        otsM2 = mAThetaSpeed2;

        % Runge-Kutta fourth order solver.
        U = [ots1, ots2, ot1, ot2];
        V1 = calcDiffs(U, 0);
        V2 = calcDiffs(U + (.01*V1/2), 0);
        V3 = calcDiffs(U + (.01*V2/2), 0);
        V4 = calcDiffs(U + (.01*V3), 0);
        U = U + .01*(V1 + 2*V2 + 2*V3 + V4) / 6;

        % Update time-step.
        mRevolve = mod(mRevolve, mL) + 1;
        mThetaSpeed1 = U(1);
        mThetaSpeed2 = U(2);
        mTheta1(mRevolve) = mod(U(3), 2*pi);
        mTheta2(mRevolve) = mod(U(4), 2*pi);
```

```
            % Repeat for magnetized version.
            U = [otsM1, otsM2, otM1, otM2];
            V1 = calcDiffs(U, 1);
            V2 = calcDiffs(U + (.01*V1/2), 1);
            V3 = calcDiffs(U + (.01*V2/2), 1);
            V4 = calcDiffs(U + (.01*V3), 1);
            U = U + .01*(V1 + 2*V2 + 2*V3 + V4) / 6;

            mAThetaSpeed1 = U(1);
            mAThetaSpeed2 = U(2);
            mATheta1(mRevolve) = mod(U(3), 2*pi);
            mATheta2(mRevolve) = mod(U(4), 2*pi);

            refreshDrawing();
        end
        mStop = 0;
        mRunning = 0;
    end

end % end of magpendulum
```

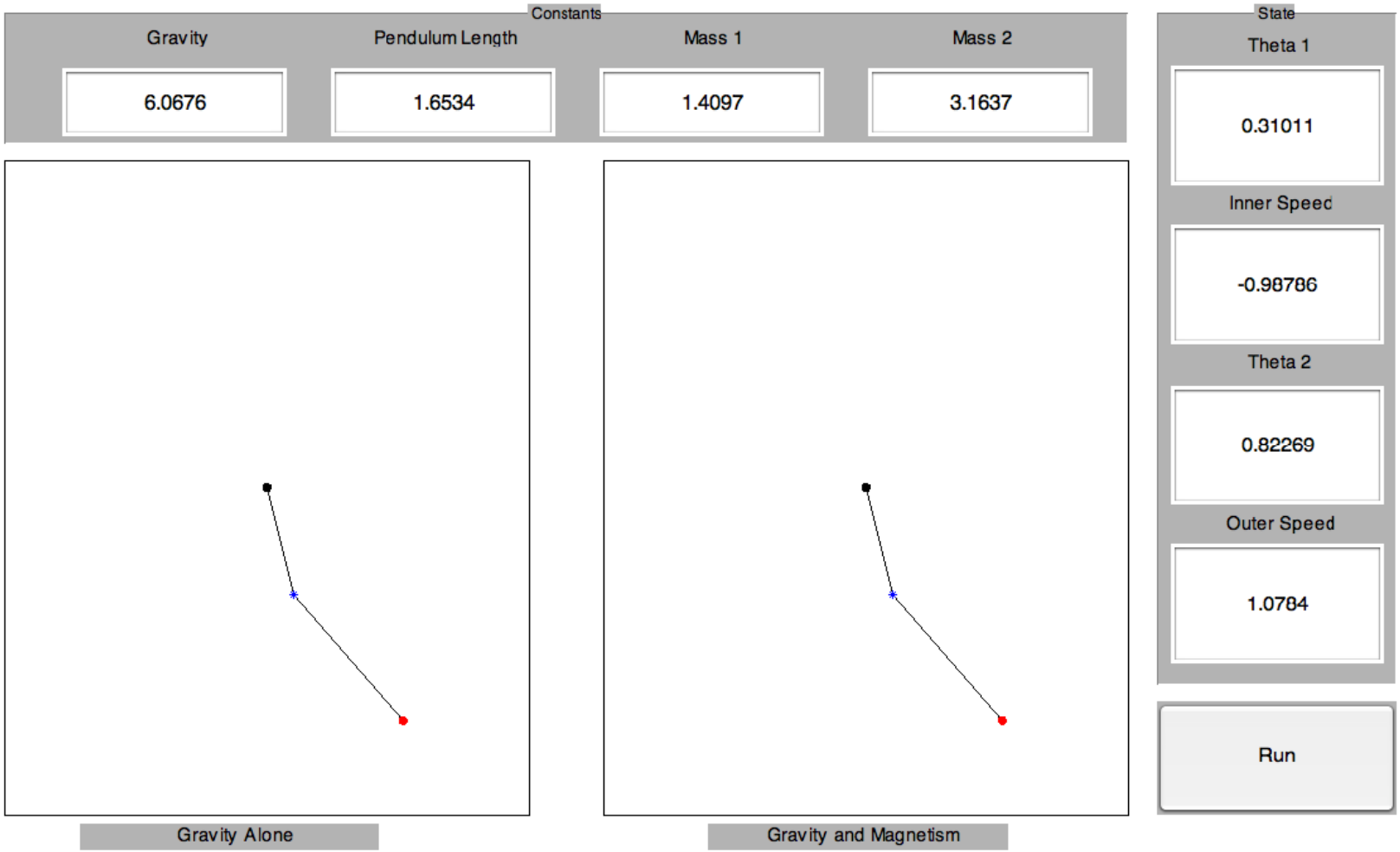

FIGURE 3. Initial state of the system

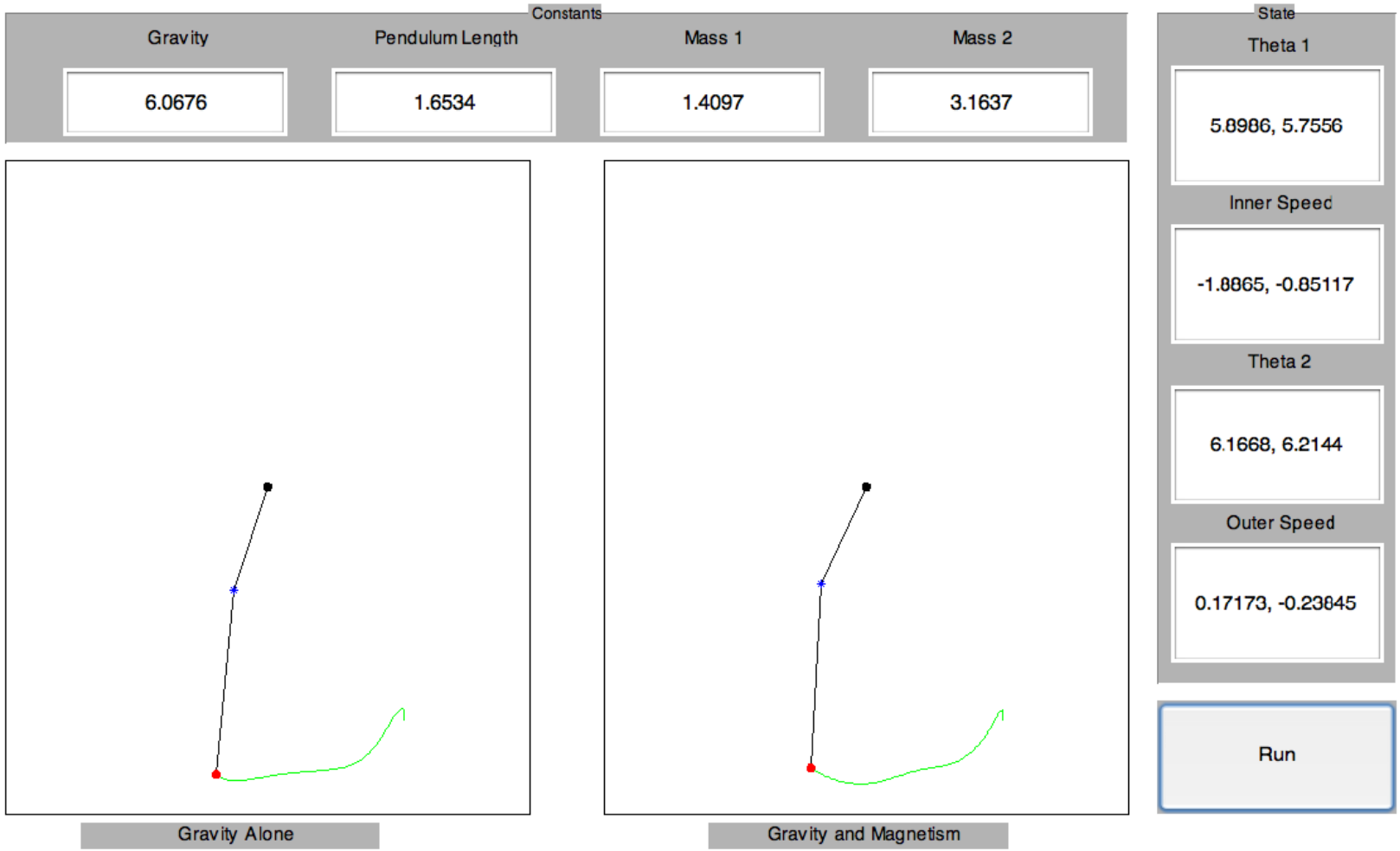

FIGURE 4. Swinging alone force lines

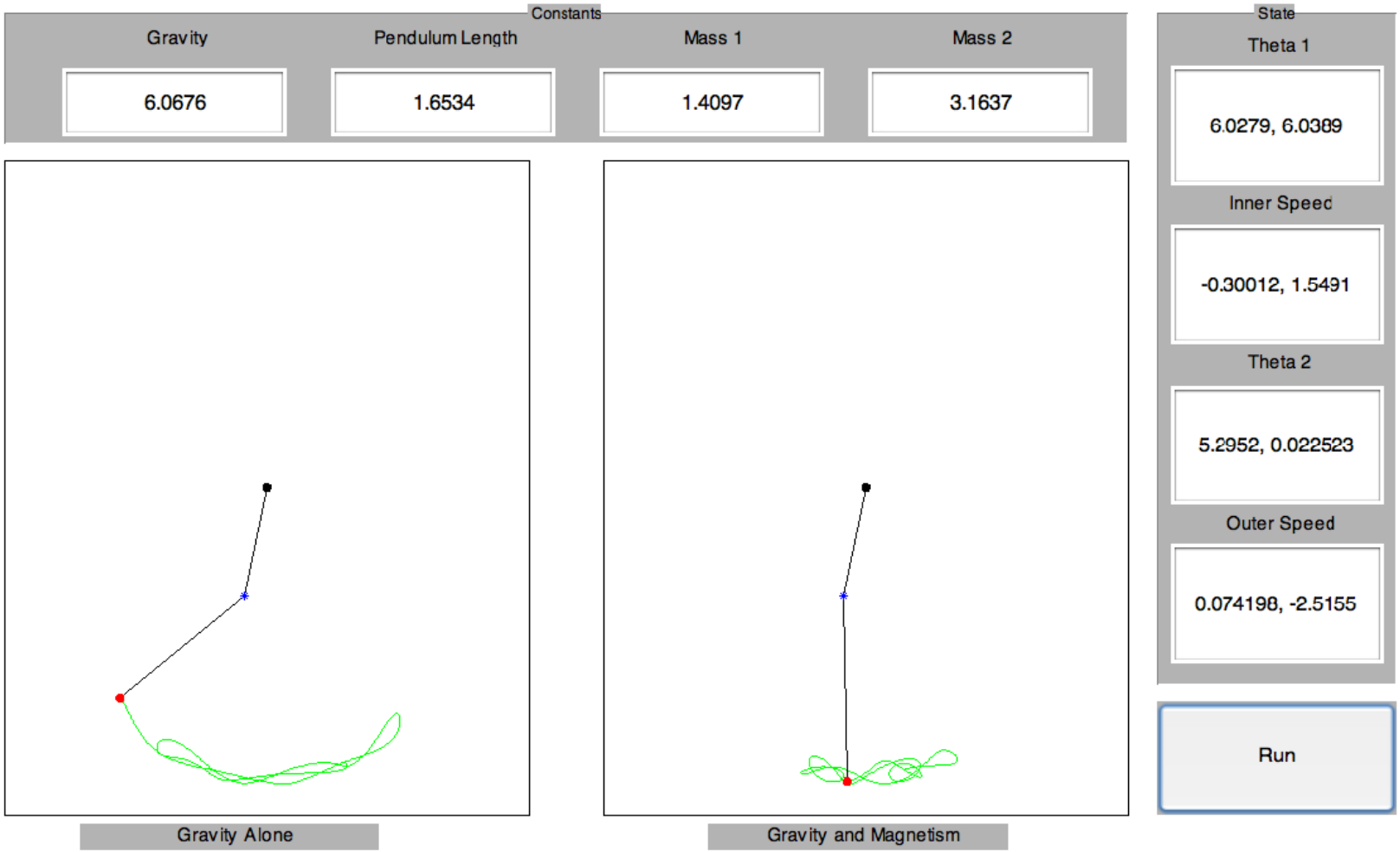

FIGURE 5. Opposed by the magnetic field and drifting

5. References

Halliday, Resnick, & Crane (2001). *Physics, Volume 2.* Wiley.