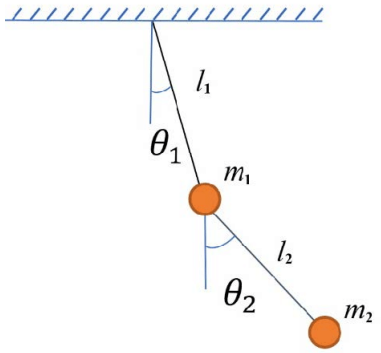
The Double Pendulum

MATH1902: Numerical Solution of Differential Equations

http://people.sc.fsu.edu/~jburkardt/classes/math1902_2020/double_pendulum/double_pendulum.pdf

---



We track the angles θ_1 and θ_2; masses and lengths are parameters

The double pendulum

The double pendulum system can exhibit sensitivity and chaotic solutions.

1 The nonlinear pendulum model

As a warmup, let us begin with the nonlinear pendulum model. The unknown variables are θ and $\frac{d\theta}{dt}$, the deflection angle in radians, and angular velocity in radians/second. For convenience, we will refer to these two variables as y_1 and y_2 respectively. Newton's laws allow us to write the following equations:

$$y_1' = y_2$$
$$y_2' = -\frac{g}{l}\sin(y_1)$$

where time is measured in seconds, the pendulum length l in meters, and $g = 9.8$ meters/seconds is the gravitational constant.

A typical initial condition at time $t_0 = 0$ might be:

$$y_1(t_0) = \frac{\pi}{6}$$
$$y_2(t_0) = 0$$

This starting condition means that the maximum deflection of the pendulum will be $\frac{\pi}{6} \approx 0.5236$. In this nonlinear model, the energy is:

$$E = m\,g\,l\,(1.0 - \cos(y_1)) + 0.5\,m\,y_2^2$$

2 The linear pendulum model

Our initial pendulum deflection is relatively small, and the initial angular velocity is zero. This implies that the pendulum will never rise higher than this initial value. Note further that $\sin(\frac{\pi}{6}) = 0.5$, in other words,

for this problem, the approximation $\sin(\theta) \approx \theta$ will be reasonably close for all the values of angular deflection that we will encounter. It is this fact that allows us to approximate our nonlinear ODE by the following linear pendulum model, replacing $\sin(y_1)$ by y_1:

$$y_1' = y_2$$
$$y_2' = -\frac{g}{l} y_1$$

The ODE model is linear because both right hand sides are linear in the y variables. This model is less accurate than the nonlinear model, because we've made an extra approximation. However, it's beloved by mathematicians because it's exactly solvable. For our initial condition, we see that the linear model has the exact solution:

$$y_1(t) = \frac{\pi}{6} \cos(\omega t)$$
$$y_2(t) = -\omega \frac{\pi}{6} \sin(\omega t)$$

where, for convenience, we define $\omega = \sqrt{\frac{g}{l}}$. This implies that our solution functions will be periodic with period $p = \frac{2\pi}{\omega}$. For our problem, $\omega \approx 3.1337$ and $p \approx 2.0050$.

Thus any solution of the linear pendulum problem will involve simple harmonic motion, a combination of sine and cosine functions with period $\frac{2\pi}{\omega}$, and so a phase plane plot of the solution will be an ellipse (a circle if $\omega = 1$).

In this linear model, the energy is:

$$E = 0.5\, m\, g\, l\, y_1^2 + 0.5\, m\, y_2^2$$

3 Comparing two pendulum models

Because they are simplifications of reality, all models are wrong. We know one way in which our linear pendulum model is wrong, namely in the use of the approximation $\sin(\theta) \approx \theta$, which gives us a mathematical problem for which we can find an exact solution. We expect that our linearized model is acceptable as long as the oscillations are small. But how much have we given up by this change? It's easy to answer that question by setting up the MATLAB code for the nonlinear pendulum problem, running both models with a fairly large initial deflection, and comparing.

Our nonlinear pendulum derivative function is simply:

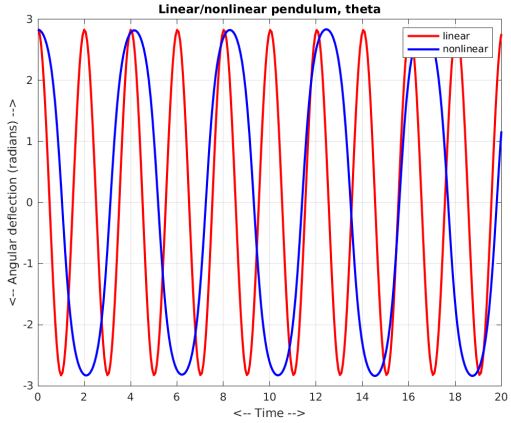
```
1  function dydt = pendulum_nonlinear_deriv ( t, y )
2
3    [ g, l, m, t0, y0 ] = pendulum_parameters ( );
4
5    u = y(1);
6    v = y(2);
7
8    dudt = v;
9    dvdt = - ( g / l ) * sin ( u );  % sin(u), not u!
10
11   dydt = [ dudt; dvdt ];
12
13   return
14 end
```

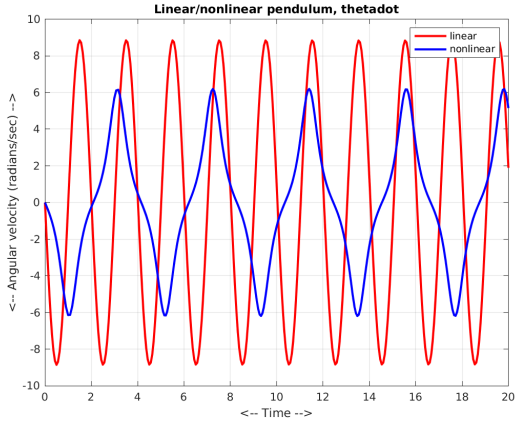
Listing 1: Nonlinear pendulum derivative

and for our initial condition, let's go $\frac{9}{10}$ of the way to the top, where $\theta = \pi$:

$$y_1(t_0) = 0.9 * \pi$$
$$y_2(t_0) = 0$$

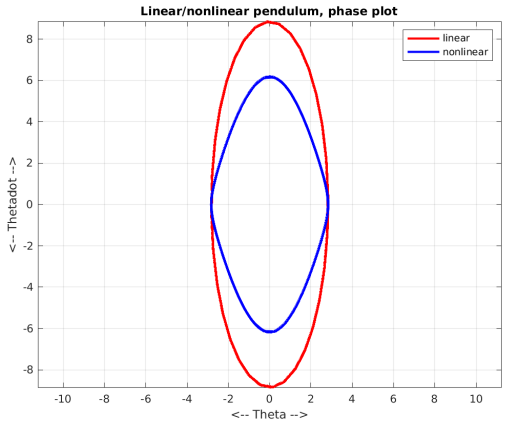
If we plot θ and $\frac{d\theta}{dt}$, we can see that the nonlinear pendulum (blue) still has a regular repeating behavior (but it's not actually a sine or cosine curve!). Instead of the linear period of about $p \approx 2.0050$, we see a longer period of about $p \approx 4$. Moreover, the shape of the $\frac{d\theta}{dt}$ curve is remarkably different for the nonlinear case.

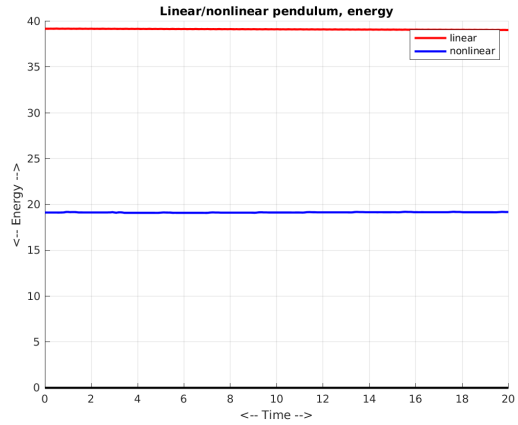




Comparison of solutions for linear and nonlinear pendulums

Moreover, the phase plot shows that the nonlinear pendulum (blue) is not following a circular or elliptical path, but some other shape entirely, and the energy value is very different.





Comparison of solutions for linear and nonlinear pendulums

If we had done this comparison for a problem with a small initial deflection, the two models would have been very close. However, assuming that the nonlinear model is closer to reality, we can see that the linear model is only reliable for small oscillations of the pendulum. It's easy to perform physical experiments that show that the nonlinear model agrees with reality, aside from friction effects.

4 The double pendulum model

Now we will proceed to a related problem which, at first sight, seems only "slightly" more complicated than our previous cases that we understand reasonably well. We start with one pendulum, and attach another on the end of the first.

To be precise, in the double pendulum problem, a rod of length l_1 is fixed at one end (0,0), and forms an angle θ_1 with the downward vertical, so that its endpoint is at $(x_1, y_1) = (l_1 \sin(\theta_1), l_1 \cos(\theta_1))$. (Remember that we measure angles from the downward vertical direction now!) A weight of mass m_1 is attached to the end of the first rod.

A second rod, of length l_2 is also attached to this end of the first rod, and can swing freely at the attachment point. This second rod forms an angle θ_2 with the downward vertical. A weight of mass m_2 is attached to the free end of the second rod. The position of this weight is $(x_2, y_2) = (x_1, y_1) + (l_2 \sin(\theta_2), l_2 \cos(\theta_2))$.

Gravity has a force coefficient of g.

The unknowns are the angles θ_1, θ_2 and their time derivatives. Starting from some initial condition, we wish to predict the locations and angular velocities of the two masses for future time.

We store the unknowns in a vector z:

$$z = \begin{bmatrix} \theta_1 \\ \frac{d\theta_1}{dt} \\ \theta_2 \\ \frac{d\theta_2}{dt} \end{bmatrix}$$

The differential equations become:

$$\begin{aligned}
z_1' &= z_2 \\
z_2' &= -\frac{((g(2m_1 + m_2)\sin(z_1) + m_2(g\sin(z_1 - 2z_3) + 2(l_2 z_4^2 + l_1 z_2^2 \cos(z_1 - z_3))\sin(z_1 - z_3)))}{2l_1(m_1 + m_2 - m_2\cos^2(z_1 - z_3))} \\
z_3' &= z_4 \\
z_4' &= \frac{(((m_1 + m_2)(l_1 z_2^2 + g\cos(z_1)) + l_2 m_2 z_4^2 \cos(z_1 - z_3))\sin(z_1 - z_3))}{l_2(m_1 + m_2 - m_2\cos^2(z_1 - z_3))}
\end{aligned}$$

These derivatives can be evaluated by calling the following function, which is available on the web page.

```
dzdt = double_pendulum_deriv ( t, z );
```

Listing 2: double_pendulum_deriv.m evaluates the ODE.

The system has 5 parameters. To start with, we will assign them the following default values:

- $\mathbf{g} = 9.81 \frac{\text{meters}}{\text{second}^2}$
- $\mathbf{m_1} = 1$ kilogram;
- $\mathbf{m_2} = 1$ kilogram;
- $\mathbf{l_1} = 2$ meters;
- $\mathbf{l_2} = 1$ meter;

These values of the parameters are available in the following function.

```
[ g, m1, m2, l1, l2 ] = double_pendulum_parameters ( );
```

Listing 3: double_pendulum_parameters.m returns parameters.

Time t will be measured in seconds, and angles θ in radians. If you want to run the problem with different values for some of the parameters, it is enough to change the values in *double_pendulum_parameters()*.

5 The system energy

A physical system has an energy. A model of the physical system can report its estimate of this energy at any time. If the model does not include any friction or other dissipative factors, then the total energy should be constant over time. It's important to determine whether a model does express this property, and whether an ODE solver for the model accurately preserves the initial energy value.

We know that the midpoint method is one example of an ODE solver that can exactly conserve energy, if that energy can be expressed as a quadratic expression in terms of the variables. As it turns out, the energy of the double pendulum system is "mostly" quadratic, but includes trigonometric terms which we cannot assume the midpoint method will be able to capture perfectly. On the other hand, many other ODE solvers don't make any claim to energy conservation, so it will be interesting to compute the energy for various solvers and see which ones do best.

Here is the awful expression for the energy E of the double pendulum, for a given set of variables z:

$$\begin{aligned} E = & 0.5\,(m_1 + m_2)\,l_1^2\,z_2^2 \\ & + 0.5\,m_2\,l_2^2\,z_4^2 \\ & + m_2\,l_1\,l_2\,z_2\,z_4\,\cos(z_1 - z_3) \\ & - (m_1 + m_2)\,g\,l_1\,\cos(z_1) \\ & - m_2\,g\,l_2\,\cos(z_3) \end{aligned}$$

Once you have computed the solution values z over a stretch of time, and want to plot it, you can request the value of the energy from the following function:

```
e = double_pendulum_energy ( z );
```

Listing 4: double_pendulum_energy.m returns the energy

In order for an energy plot to be meaningful, it is useful to also include a plot of the x axis, so that the plot has a scale. This can be done with commands like:

```
  h = double_pendulum_energy ( z );  % Compute the energy.
  hold ( 'on' );                     % Allows multiple plots in one picture
    plot ( t, h );                   % Plot energy
    plot ( [t0,tstop], [0.0, 0.0] ); % Add plot of x axis
  hold ( 'off' );                    % Finish the picture.
```

Listing 5: Plotting energy + x axis.

6 Simulate the small perturbation problem with Forward Euler

Our first experiment will be timid. We will do a "small perturbation" problem, for which the double pendulum starts at time $t_0 = 0$ with the initial condition $z(t_0) = [0.25, 0, 0, 0]$. Thus the first pendulum has been deflected to the small angle of $\theta = 0.25$ radians $= 0.25 * \frac{180}{2\pi} \approx 14.3°$ and the second pendulum is hanging straight down. We are interested in studying the system over the time period $t_0 = 0 \leq t \leq 50 =$ tstop. We start with our forward Euler code, available on the web page as *euler.m*.

The data will be numerous, and complicated. In order to have a chance of comprehending what is going on, we will need plots. The solution z is in terms of angles and angular velocities, but we would prefer to see the (x, y) coordinates of the two masses of the pendulum. So once we have computed z, we will convert from angles $z_1 = \theta_1$, $z_3 = \theta_2$, to Cartesian coordinates $(x_1, y_1), (x_2, y_2)$. Because we are measuring angles starting from the vertical downward direction ("6 o'clock") rather than the usual angular origin ("3 o'clock"), these formulas may not look like what you would first guess them to be:

```
  x1 =       l1 * sin(z1);
  y1 =    - l1 * cos(z1);
  x2 = x1 + l2 * sin(z3);
  y2 = y1 - l2 * cos(z3);
```

Listing 6: Converting angles to positions.

Then we will be interested in the following plots:

1. plot (t,x1(t)) and (t,y1(t)), the coordinates of mass m_1 over time;
2. plot (t,x2(t)) and (t,y2(t)), the coordinates of mass m_2 over time;
3. plot (x1(t),y1(t)), the position of m_1;
4. plot (x2(t),y2(t)), the position of m_2;
5. plot (t, e(t)), the energy E(t) over time;

For small perturbations, plots #1 and #2 should seem roughly periodic and roughly regular. For larger perturbations, where one or both pendulums can actually swing through the vertical upright position, we may expect much more complicated behavior. Plot #3 should be a circle, or a portion of a circle for small problems. Plot #4 should be made by circular paths looping along the circle of plot #1. Plot #5 would be a perfectly straight line if our ODE solver was exactly conservative. Instead, we can only hope that the line stays close to its initial value at $t = 0$. In order to judge whether the variations are significant, it's helpful to draw a second line for $e = 0$, so that the plot has an implicit scale.

7 A code to call euler() for the double pendulum

The Forward Euler code uses a fixed stepsize. To get reasonable results over our time interval, it is necessary to use a very large number of steps. Here, we didn't get believable pictures until trying $n = 100,000$ steps. And this is for the small perturbation problem!

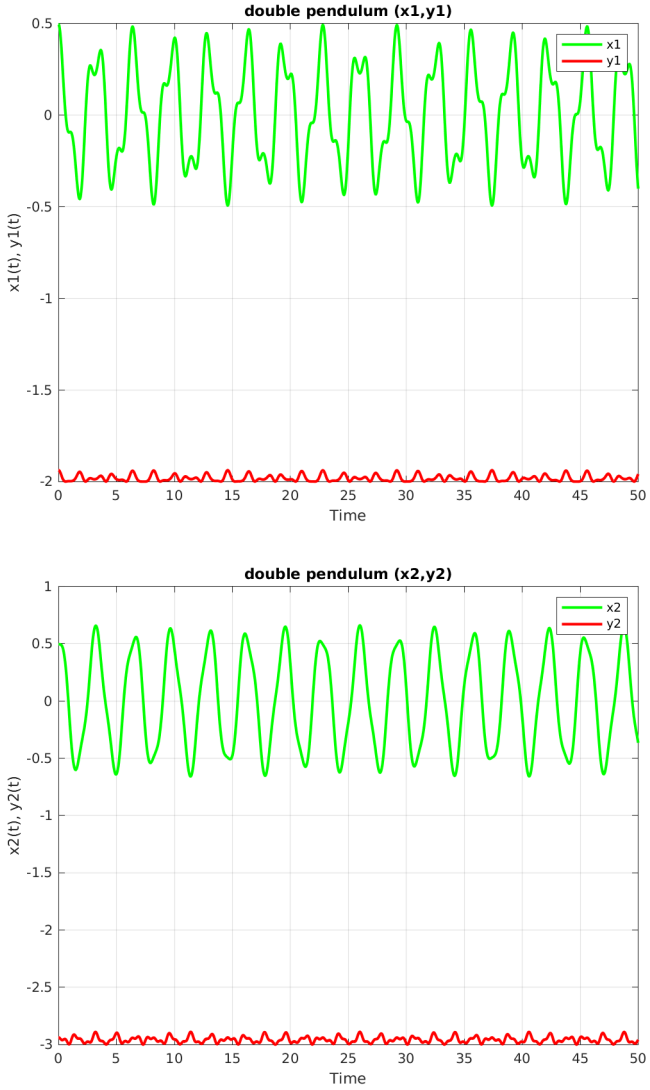
```
  f = @ double_pendulum_deriv;
  t0 = 0.0;
  tstop = 50.0;
  tspan = [ t0, tstop ];
  z0 = [ 0.25; 0.0; 0.0; 0.0 ];
  n = 100000;

  [ t, z ] = euler ( f, tspan, z0, n );
%
%  Compute Cartesian coordinates.
%
  [ g, m1, m2, l1, l2 ] = double_pendulum_data ( );

  x1 =   l1 * sin ( z(1,:) );
  y1 = - l1 * cos ( z(1,:) );
  x2 = x1 + l2 * sin ( z(3,:) );
  y2 = y1 - l2 * cos ( z(3,:) );
%
%  Plot #1.
%
  figure ( 1 );
  plot ( t, x1, 'g', ...
         t, y1, 'r', ...
         'LineWidth', 2 );
  grid ( 'on' );
  xlabel ( 'Time' );
  ylabel ( 'x1(t), y1(t)' );
  legend ( 'x1', 'y1' );
  title ( 'double pendulum (x1,y1)' )
```

```
30    filename = 'double_pendulum_x1y1.png';
31    print ( '-dpng', filename );
32    fprintf ( 1, '  Graphics saved as "%s"\n', filename );
33  %
34  %  More plots...
35  %
```

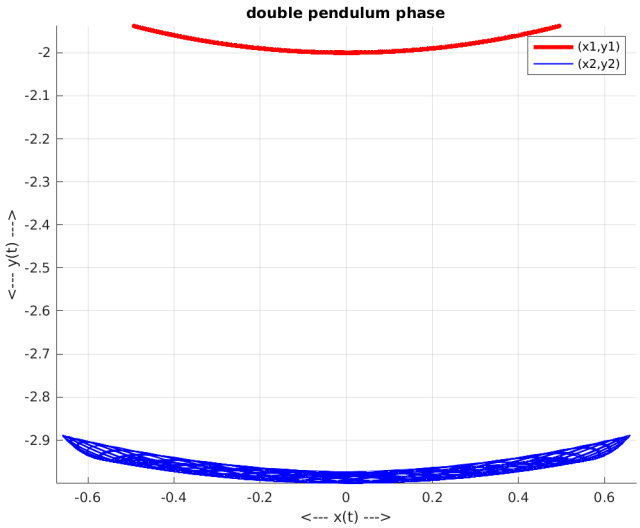
Listing 7: Solving double pendulum with euler method.

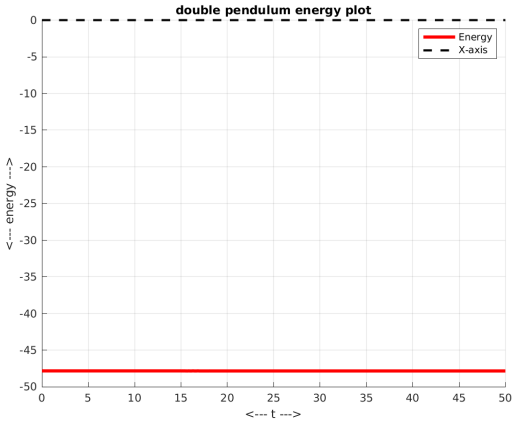
8 Graphic results for the small perturbation problem

The code *double_pendulum_test.m* can use the Euler method to solve our small perturbation problem, and produces plots that summarize the results. Because the pendulum is only slightly swinging, the two components of the pendulum seem to be swinging together almost as one.



Time plots of (t, x_1), (t, y_1) *and* (t, x_2), (t, y_2) *show regularity and periodicity*





The phase plot shows that pendulum #1 stays on a circular arc, and #2 gently loops around it.below it; the energy plot shows conservation.

The graphic results from the small perturbation problem suggest that the solution can exhibit regularity and periodicity for both (x_1, y_1) and (x_2, y_2), the positions of the two pendulum masses. We see that the phase portrait is pretty calm, with the first pendulum tracing a thin line, while the thicker line for the second pendulum mass indicates that its motion is somewhat more complicated - it must actually be jiggling up and down just a bit.

9 Experiments

Our small perturbation seems to have allowed the double pendulum to behave almost as though it were still just a single pendulum. We need to shake things up a bit to see what the system is capable of. Let us consider three sets of initial conditions:

1. *small*: $z = [0.25, 0, 0, 0]$;
2. *medium*: $z = [\pi/3, 0, 0, 2]$;
3. *large*: $z = [\pi, 0, \pi, 4]$;

We have already considered the small perturbation case, so it is up to you to run the medium and large perturbations, make the plots, and try to decide what they are telling you.

You may want to check whether the results would change significantly if you increase the number of steps (for a fixed step method) or tighten the error tolerance (for an adaptive method). If the energy conservation improves, for instance, then you might conclude that the first computation was simply not accurate enough. Otherwise, you might wonder whether this ODE system can be too wild to approximate accurately.

The *double_pendulum_test.m* file has some commented out portions that make experimentation easy. Simply by uncommenting certain lines, you can switch the ODE solver between *euler()*, *rk4()*, *midpoint()* or *ode45()*. Especially for the medium and large initial conditions, these solvers may return results that are significantly different.

The function *double_pendulum_parameters()* sets the values of *g, l1, l2, m1, m2*. Changing any value in this file, will cause *double_pendulum_test()* to use that new value instead of the ones we talked about initially. In particular, you might investigate what happens if

- you set *l2* to 5, or to 1/5;
- you set *m2* to 5, or to 1/5;

The initial condition has a huge influence on the solution. What happens if you choose an initial condition so that

- pendulum 1 is not moving: for example, z0 = [0, 0, 0, 5];
- pendulum 1 and pendulum 2 are at the same angle: for example: z0 = [pi/3, 0, pi/3, 0]
- pendulum 1 and pendulum 2 are at opposite angles: for example: z0 = [pi/3, 0, -pi/3, 0]
- the angular velocity is just large enough that the pendulum will reach the top, and keep going. (Then theta1 and theta2 will not necessarily stay between -pi and pi.)

There is a MATLAB program, written by Alexander Erlich, which can solve the double pendulum problem, display the pendulum system, and make a movie of the results. Watching the animation for various perturbation sizes, it is easy to see how the system can be regular and rhythmic for small perturbation, becoming chaotic when more energy is available, so that the pendulums can both swing full circle.

On the course website, there is a copy of the program, set up to solve the small perturbation problem, in the file *double_pendulum_movie_small.m*, as well as the resulting animation *double_pendulum_movie_small.avi*. To change the code to do the medium or large perturbation problems only requires changing the data in the initial segment of the code.

In that initial code, you can also easily see how to choose different values for the gravitational, mass, and length parameters, or to experiment with other initial conditions.

10 HW # 10

Choose some of the double pendulum experiments above, or set up your own. Write a short report on your results.

Send your report to `trenchea@pitt.edu`.

Professor Trenchea and I have enjoyed presenting this material to you. We hope it has shown you some of the ways that researchers can deal with mathematical problems by taking a computational approach.