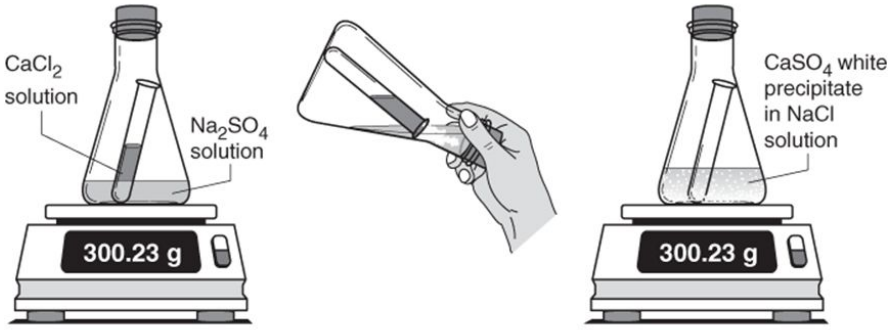
Conservation

MATH1902: Numerical Solution of Differential Equations

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

---

Conservation of mass



mass (g) of reactants = mass (g) of products

Physical laws often require the conservation of some quantity.

Conservation

If a physical law requires that mass be conserved, or that we stay on the surface of the earth, can an ODE solver produce solutions that satisfy this requirement?

1 In real life, systems follow hidden laws

Mathematicians are perfectly happy making up differential equations to solve, even when they have little to do with the real world, For instance, the "humps" ODE was the derivative of a function that I made up.

However, ODE's are important because they can be used to model, understand, and predict the behavior of electrical systems, engineering structures, astronomical bodies, and the absorption of medicines into the body. These systems often obey additional underlying laws that are worth observing.

One of the most useful concepts is that of **conservation**, the idea that although the observed system changes, there may be some hidden quantity whose value nonetheless stays the same. In particularl, physicists learn early on about conservation of mass, momentum, energy, and charge. If a hidden quantity is conserved in real life, we would hope that the same would be true for our ODE model.

Note that conservation is not the same as accuracy. If an ODE is meant to describe our location as we wander around the surface of the earth, then an accurate ODE solver might specify our location to within 100 feet, but might actually place us 100 feet up in the air. On the other hand, a conservative ODE solver might be off by 100 feet, but it would specify a location that was exactly on the surface of the earth, conserving the value of $x^2 + y^2 + z^2$, our distance from the earth's center. I hope you can see that there is a difference between being 100 feet up in the air, or 100 feet away from your desired location (but with your feet safely on the ground), even though both results have the same "accuracy".

2 The SIR ODE model of an epidemic

A model of the evolution of an epidemic disease is known as the Susceptible/Infected/Recovered or **SIR** model. We start with a population of P people, whose health is to be monitored over some time interval $t_0 \leq t \leq tstop$. At any time t, a person's status is denoted by one of three cases:

- **S**: susceptible, healthy but can catch the disease from an infected person;
- **I**: infected, can transmit the disease;
- **R**: recovered, had the disease, recovered, and is immune for some time;

Instead of monitoring the health of each person, we are simply going to keep track of three numbers at each time t, so that $S(t), I(t), R(t)$ count the number of individuals in each health class.

We will not be bothered by the fact that each of these numbers would ordinarily be an integer, whereas our results will be real numbers.

Rather than writing the differential equation immediately, let's begin by thinking of the situation as one in which we classify the health of our N individuals at time intervals of size Δt. Before we make any assumptions about how things change, we have our simplest model, in which things don't change at all:

$$S(t + \Delta t) = S(t)$$
$$I(t + \Delta t) = I(t)$$
$$R(t + \Delta t) = R(t)$$

SIR0 model: nothing changes

However, because we are modeling a real life system, there are several conditions that we can expect to be satisfied at every time t:

- We must have $S(t), I(t), R(t)$ nonnegative;
- We must have $S(t) + I(t) + R(t) = P$;
- If $I(t)$ ever becomes zero, then the epidemic is over;

Notice that the second condition is a conservation law. If we started with P people, our ODE solver must preserve this number.

If the disease could be fatal, then we could imagine losing some of our participants. However, we could add a fourth variable $D(t)$, with the obvious meaniing. In the case of a potentially fatal disease, we would then require $S(t) + I(t) + R(t) + D(t) = P$, so that our model satisfies this adjusted conservation law.

To set up our ODE, we need to make some assumptions about how an epidemic disease works.

We will make some very strong simplifying assumptions about how the disease spreads. Over a typical time interval Δt, we will assume that every pair of people can have an interaction that potentially transmits the disease. Here is a table of all the possible interactions:

```
S*S  S*I  S*R
I*S  I*I  I*R
R*S  R*I  R*R
```

The only "interesting" cases are S*I and I*S, in which a susceptible person meets an infected person. We really only need to count these encounters once, so we can assume that, at time t, over a typical time interval Δt, there are $S(t) * I(t)$ encounters in which the disease could be transmitted. Let α measure the probability

that such an encounter will actually transmit the disease. Then both $S(t)$ and $I(t)$ will change:

$$S(t+\Delta t) = S(t) - \frac{\alpha}{P} * S(t) * I(t) * \Delta t \qquad \text{(subtract newly infected susceptibles)}$$
$$I(t+\Delta t) = I(t) + \frac{\alpha}{P} * S(t) * I(t) * \Delta t \qquad \text{(add newly infected susceptibles)}$$
$$R(t+\Delta t) = R(t)$$

SIR1 model: susceptibles can become infected

But if this is our model, then people stay sick forever. In fact, we want to allow any sick person to recover over a time period Δt, with probability β. This means that our model becomes

$$S(t+\Delta t) = S(t) - \frac{\alpha}{P} * S(t) * I(t) * \Delta t$$
$$I(t+\Delta t) = I(t) + \frac{\alpha}{P} * S(t) * I(t) * \Delta t - \beta * I(t) * \Delta t \qquad \text{(subtract newly recovered infecteds)}$$
$$R(t+\Delta t) = R(t) + \beta * I(t) * \Delta t \qquad \text{(add newly recovered infecteds)}$$

SIR2 model: SIR1 + infected people can recover

Our final adjustment is made to account for the possibility that people who have recovered from the disease might catch it again. If that is the case, then there is a probability γ that any recovered person must move back into the susceptible class. And in that case, our SIR difference equation becomes:

$$S(t+\Delta t) = S(t) - \frac{\alpha}{P} * S(t) * I(t) * \Delta t + \gamma * R(t) * \Delta t \qquad \text{(add newly susceptible recovereds)}$$
$$I(t+\Delta t) = I(t) + \frac{\alpha}{P} * S(t) * I(t) * \Delta t - \beta * I(t) * \Delta t$$
$$R(t+\Delta t) = R(t) + \beta * I(t) * \Delta t - \gamma * R(t) * \Delta t \qquad \text{(subtract newly susceptible recovereds)}$$

SIR3 model: SIR2 + recovered people can become susceptible again.

Now we can rearrange things, and take the limit as $\Delta t \to 0$ to transform our SIR difference equation into an SIR differential equation:

$$\frac{dS}{dt} = -\frac{\alpha}{P} SI + \gamma R$$
$$\frac{dI}{dt} = +\frac{\alpha}{P} SI - \beta I$$
$$\frac{dR}{dt} = +\beta I - \gamma R$$

SIR differential equation model

Note that we can make SIR0, SIR1, SIR2 and SIR3 models from this ODE, simply by zeroing out some of the parameters. Thus, we get an SIR2 model if we set $\gamma = 0$.

There's actually a conservation law that these differential equations are hiding. Define the variable $P(t) = S(t) + I(t) + R(t)$ and note that if must follow that $\frac{dP}{dt} = \frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt}$. If you add up the left hand sides of the SIR ODE, and the right hand sides of the SIR ODE, you get the following differential equation:

$$\frac{dP}{dt} = 0$$

In other words, our SIR ODE model "knows" that the value of P, the total population, must be conserved. That's a mathematical statement. This does not guarantee that there is a corresponding computational statement of conservation! We will monitor the behavior of $P(t)$ over time, as a way to check whether our ODE solutions are "reasonable".

3 *sir_ode.m*: Population conservation

For our SIR problem, we will let set the population to $P = 100$ individuals. We will study the epidemic over the time interval $0 \leq t \leq 5000$. And we will use an initial condition of `[99, 1, 0]`, that is, initially there is just one infected person, and the rest are susceptible. Even though we are solving over a long time period, we can try a fairly small number of time steps, $n = 50$.

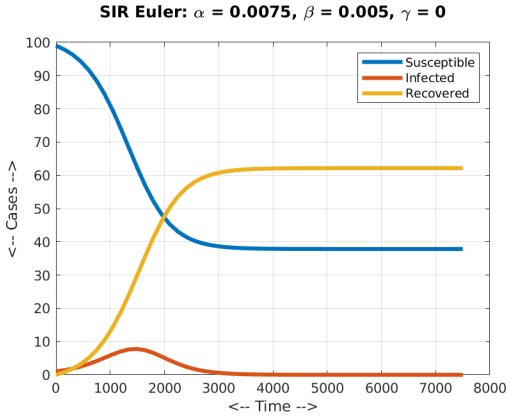
We also need to specify the values of the parameters α, β, γ. Let's start with an SIR2 model, and set $\alpha = 0.0075, \beta = 0.0050, \gamma = 0$.

Now we need to write the function *sir_deriv()* which evaluates the right hand side of the SIR ODE. Since this is a set of 3 equations, the current values of S, I and R will be stored in the vector y of length 3. Here's an outline of what the function might look like:

```
function dydt = sir_deriv ( t, y )

  S = y(1);
  I = y(2);
  R = y(3);
  P = ?;

  alpha = ?
  beta = ?
  gamma = ?

  dSdt = ?
  dIdt = ?
  dRdt = ?

  dydt = [ dSdt; dIdt; dRdt ];

  return
end
```

Listing 1: Outline of sir_deriv.

Once you have gotten this derivative function written, you should know how to write a function *sir_euler()*, which calls the *euler()* ODE solver to compute the solution pair `[ t, y ]`. Then you can use the command `plot ( t, y )` to automatically display all three solution components in one plot.



An SIR2 simulation ($\gamma = 0$).

From the plot, it looks like the epidemic has a brief surge and then dies away. Actually, the number of infected people never reaches zero, so mathematically the epidemic smolders away for all time. In a more

realistic approach, we could declare that once the number of infected people is less than 1, or perhaps less than $\frac{1}{2}$, the epidemic is over (remember that in our model, we allow these numbers to be decimal values.)

To check on conservation, you can compute the value of P over time, by the command `P = sum ( y, 2 )`. The "2" tells MATLAB to sum the rows of y, rather than the default behavior of summing the columns. Then the command `plot(t,P)` will show how well the ODE solver is conserving the total population.

For this ODE, the Euler method does an almost perfect job of conserving the value of P. We will shortly see some more challenging problems in which the conserved quantity involves quadratic terms, for which the Euler method does not do so well!

For a challenge, you can investigate several aspects of the SIR model:

- At the end of our time interval, about 40% of the population has never had the disease. This fact is related to the value of α. Try to determine approximate values of α that would result in 10%, 20%, ..., 90% of the population being unaffected by the ending time.
- The value of β corresponds roughly to how quickly an infected person is no longer infectious, moving to the recovered group. Using our original value of α, try to simulate an improvement in treatment of infected individuals, In other words, increase β. Find an approximate value of β so that the epidemic is reduced so that about 60% of the population has never had the disease by the end time.
- The value of γ controls how quickly a recovered person loses their immunity and becomes susceptible again. In the SIR2 model, we leave γ at 0. Try to approximate a value of γ which means that instead of 40% of the people being unaffected, this rate becomes 20%. You can start with the value $\gamma = 0.001$ and work from there.

4 *sphere_ode*: Distance conservation

The "sphere" differential equation system describes changes in the position (x, y, z) of an object which is moving along a closed loop on the surface of the unit sphere:

$$i1 = 1.6, i2 = 1, i3 = \frac{2}{3}$$
$$\frac{\partial x}{\partial t} = (\frac{1}{i3} - \frac{1}{i2})zy$$
$$\frac{\partial y}{\partial t} = (\frac{1}{i1} - \frac{1}{i3})xz$$
$$\frac{\partial z}{\partial t} = (\frac{1}{i2} - \frac{1}{i1})yx$$

We can imagine that a friend is driving a car on some closed circuit over the face of the earth. We ignore the existence of oceans, assume the earth is perfectly spherical, and that we are using units of measurement so that the earth has a radius of 1.

Unfortunately, we would normally use y to denote the vector of our solution values, but now we want one of those variables to be named y. So for this problem, we will use the name xyz for the full solution vector. We will need to write an ODE right hand side function *sphere_deriv.m* which would look something like this:

```
1  function dxyzdt = sphere_deriv ( t, xyz )
2    x = xyz(1);
3    y = xyz(2);
4    z = xyz(3);
5    i1 = ?
6    i2 = ?
7    i3 = ?
8    dxdt = ?
```

```
9    dydt = ?
10   dzdt = ?
11   dxyzdt = [ dxdt; dydt; dzdt ];
12   return
13 end
```

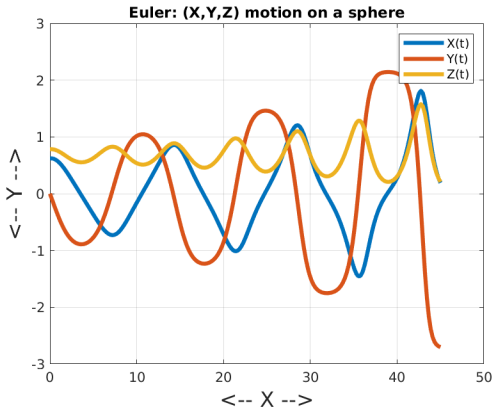
Listing 2: Outline for sphere_deriv.m

Because our position $(x(t), y(t), z(t))$ is always on the surface of the sphere, it must be the case that we conserve a quantity we can call $h(t)$, which we recognize as simply the square of our distance from the center of the sphere:

$$h(t) = x(t)^2 + y(t)^2 + z(t)^2 = 1 \text{ for all time } t$$

Suppose that at time $t0 = 0$ hours, we start our travel at $(x(0), y(0), z(0)) = (\cos(0.9), 0.0, \sin(0.9))$, and that we have enough fuel to travel until $tstop = 45$ hours.

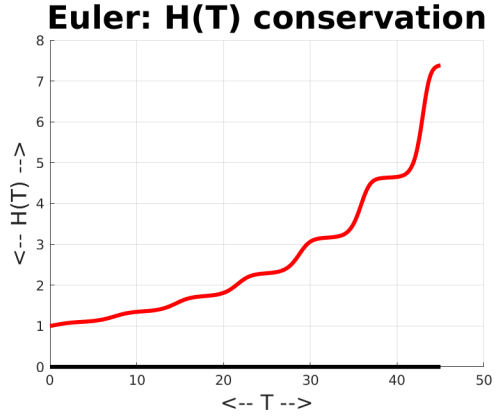
Let's consider approximating our itinerary by using $n = 200$ steps. If we use the Euler method in the usual way, we can plot the values of $x(t), y(t), z(t)$ over time. The graphs of these functions look reasonably smooth and regular:



The Euler solution to the sphere problem.

However, if you look closely at the plot, you might be concerned that the values of all three variables seem to be growing in magnitude. Does this seem reasonable?

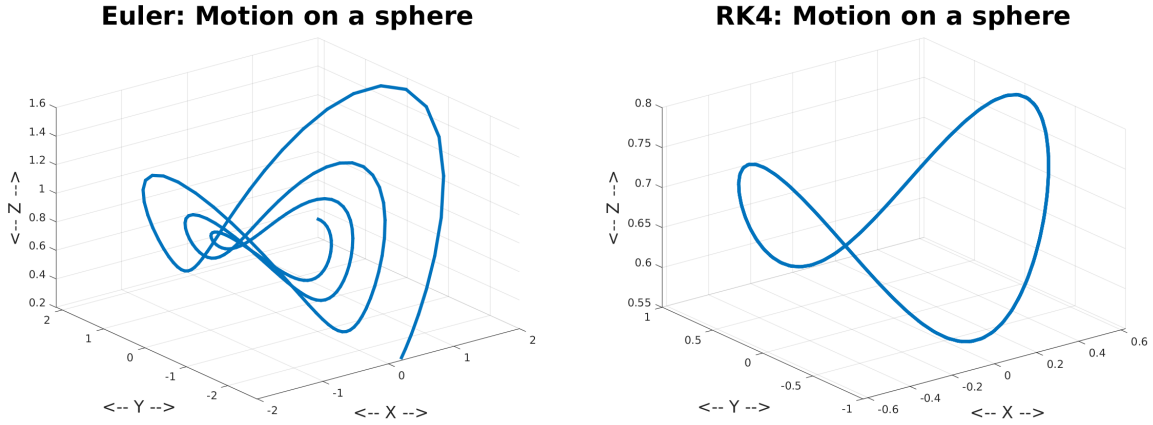
So let's compute the value of $h(t)$ over time and plot that. Remember that this quantity represents our distance from the center of the Earth, and it's rather important that this number always be equal to 1!



The value of $h(t)$ as computed by the Euler method.

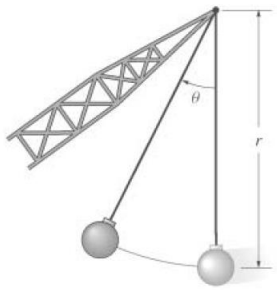
This plot reveals that our Euler solution is worthless. After 45 hours of driving around on the surface of the Earth, Euler thinks we are actually about 7 earth radii out in space!

As we know, when our ODE solution results are unsatisfactory, one thing to try is to take a smaller stepsize. An alternative is to seek a more accurate solver. Here, if we compare 3D plots of the sphere solution for the Euler method and the RK4 solver, we see how the Euler solution is spinning off into space, while the RK4 solution simply cycles around the same path, and stays on the Earth.



3D plots of the Euler and RK4 solutions.

5 *pendulum_ode()*: Energy conservation



The deflection angle of a pendulum satisfies a second-degree ODE.

The pendulum problem models the back and forth swinging motion of a pendulum. The pendulum has length $l = 1$meter and mass $m = 1$kilogram, and is subject to the force of gravity whose strength is $g = 9.8$meters/second/second. The pendulum position at any time is reported by the value of $\theta(t)$, the angle that the pendulum makes with the downward direction, and can be written as

$$m\,l^2\,\frac{d^2\theta}{dt^2} = -m\,g\,l\,\sin(\theta)$$

We replace θ by the vector y: $y_1 = \theta, y_2 = \frac{d\theta}{dt}$, Assuming that θ is small, we can also use the approximation $\sin(\theta) \approx \theta$ to simplify the right hand side. We will also find it convenient to write $\omega = \sqrt{\frac{g}{l}}$. Along with

some algebraic simplifications, we arrive at the following pair of linear ODE's for our pendulum problem:

$$y_1' = y_2$$
$$y_2' = -\omega\, y_1$$

This pair of differential equations will need to be described in the MATLAB file *pendulum_deriv.m*.

Even though our approximation is only good for small θ, we will use the following initial conditions:

$$y_1(0) = \frac{\pi}{3}$$
$$y_2(0) = 0$$

and we will approximate the solution over the interval $0 \le t \le 10$.

Note that an exact solution to this problem is then:

$$y_1(t) = y_1(0)\cos(\omega\,(t - t_0)) + \frac{y_2(0)}{\omega}\sin(\omega\,(t - t_0))$$
$$y_2(t) = -\omega\, y_1(0)\sin(\omega\,(t - t_0)) + y_2(0)\,\cos(\omega\,(t - t_0))$$

which is stored in the file *pendulum_solution.m*.

From this exact formula, we can see that a single period p of our pendulum lasts

$$p = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{1}{9.8}} \approx 2.0071 \text{ seconds}$$

Because the pendulum is a physical system, at every time t it has an energy $E(t)$, which is the sum of its potential energy $m\,g\,h$ and kinetic energy $\frac{1}{2}mv^2$. The height h is $l\,(1 - \cos\theta)$. Using a half angle formula, we see that $1 - \cos(\theta) = 2\sin^2(\theta/2) \approx \frac{\theta}{2}$, and so we can write:

$$E(t) = \frac{1}{2}\frac{m\,g}{l}\,y_1^2(t) + \frac{1}{2}m\,y_2^2(t)$$

In the absence of friction or other disturbances, this energy should be constant. Since we know the solution at the starting time t_0, this means that the energy $E(t)$ at any time should equal that initial value. In other words:

$$\begin{aligned} E(t) &= \frac{1}{2}\frac{m\,g}{l}\,y_1^2(t) + \frac{1}{2}m\,y_2^2(t) \\ &= \frac{1}{2}\frac{m\,g}{l}\,y_1^2(t_0) + \frac{1}{2}m\,y_2^2(t_0) \\ &= \frac{1}{2} * 1 * 9.8/1 * \left(\frac{\pi}{3}\right)^2 + \frac{1}{2} * 1 * 0^2 \\ &\approx 5.3735 \end{aligned}$$

We can evaluate the energy by calling *pendulum_energy()*.

Now we have defined our problem, and the MATLAB functions that can be used by an ODE routine to produce an approximate solution. While the solution is important to us, we are now also very much interested in checking whether we conserve energy.

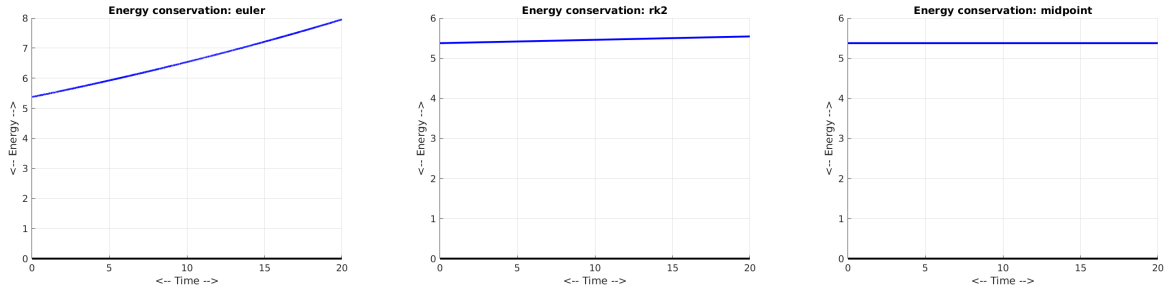
6 Solve pendulum with euler(), rk2(), and midpoint()

We start with a program *pendulum_euler()*, which calls on *euler()* to solve the pendulum ODE. We try using $n = 100, 500, 2000, 5,000, 10,000$. The first computations are a disaster. The pendulum swings further and further, and the energy, which starts out at 5.37 blows up. For the first trial $n = 100$, the energy goes to hundreds, thousands, and millions.

It seems advisable to try a more accurate solver, and so *pendulum_rk2()* switches to our next choice, *rk2()*. For the first trial, with $n = 100$, the energy blows up to about 250 by the ending time. For $n = 500$, the energy increases a bit, to a final value of about 5.6, which we may find acceptable.

Finally, the code *pendulum_midpoint()* tries out a new solver, called *midpoint()*. The midpoint solver has the same degree of accuracy as *rk2()*, that is, the global error decreases as dt^2. However, it has an extra property that is very useful in this case: if a physical system has a quantity that should be conserved, and that quantity can be expressed as a linear or quadratic function of the dependent variables, then the midpoint approximate solution should also preserve that quantity - **exactly**.

Here are the pendulum energy plots for *euler()* at $n = 10,000$, *rk2()* at $n = 500$, and *midpoint()* at $n = 100$.



Pendulum energy of Euler, RK2 and Midpoint solutions.

As mentioned before, the Euler energy plot blows up unless we take a huge number of steps; the RK2 energy plot is only approximately constant; if we took fewer steps, or ran the code for a longer time, the energy would grow; it is only for the results of the Midpoint method that we see essentially perfect conservation, even for a low number of time steps. In the next class, we will look more closely at the midpoint method.

7 *predator_ode*: A hidden conservation law

Recall the predator prey ODE: at any time t, we have populations of $u(t)$ prey and $v(t)$ predators, which might be rabbits and foxes. The names u and v are convenient, but when we work with MATLAB, we need to store both u and v in a single vector y:

$$y(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}$$

The ODEs that model the changes in population are:

$$\frac{du}{dt} = \alpha u - \beta uv$$
$$\frac{dv}{dt} = -\gamma v + \delta uv$$

where $\alpha, \beta, \gamma, \delta$ are positive parameters which control the relationship between the predators and prey.

For our experiments, we used $\alpha = 2, \beta = 1, \gamma = 1, \delta = 1$, an initial time $t_0 = 0$, and initial conditions

$$u(t_0) = 4.0$$
$$v(t_0) = 2.0$$

and we wanted to compute the solution over the range $0 \leq t \leq 250$.

We already saw that the Euler method needed to take about $n = 1,000,000$ steps to get an answer whose values showed regular oscillation, and whose phase plane made what looked like a closed curve. Methods that were more accurate were able to use much fewer steps to get a reasonable solution.

Obviously, the number of rabbits and foxes are not conserved. However, even though it's hard to give it a physical meaning, this system also has a conserved quantity:

$$H(u, v) = \delta(u) - \gamma \ln(u) + \beta v - \alpha \ln(v)$$

For our choice of parameters, this becomes

$$H(u, v) = u - \ln(u) + v - 2 \ln(v)$$

When we apply an ODE solver to compute an approximate solution to the predator prey ODE, we are not guaranteed that $H(u, v)$ will stay constant. We just saw that the RK2 method is accurate enough to do a reasonable job of conservation, and that the midpoint method does an excellent job if the conserved quantity is linear or quadratic. But since $H(u, v)$ involves the logarithm function, we can't guarantee that the midpoint method will conserve perfectly on this problem.

For the given initial conditions, the conserved value of $H(u, v)$ can be computed as:

$$\begin{aligned} H(u(0), v(0)) &= u(0) - \ln(u(0)) + v(0) - 2 \ln(v(0)) \\ &= 4 - \ln(4) + 2 - 2 \ln(2) \\ &\approx 3.2274 \end{aligned}$$

In MATLAB, we will have the solution stored in a vector y and compute a result vector h:

```
function h = predator_conserved ( y )
  u = y(:,1);
  v = y(:,2);
  h = ...
  return
end
```

Listing 3: Outline for computing conserved quantity.

8 Homework #8

Set up MATLAB codes to solve the predator prey ODE using *euler()*, *midpoint()*, and *rk2()*. Make it easy to vary the number of steps n. From the solution vector $y(t)$, compute the value of $h(t)$ and make a plot of $(t, h(t))$ over the time interval $0 \leq t \leq 50$.

For each solver,

1. Compute a solution using $n = 100$ steps, and report the value of h at the end;
2. Increase n as necessary until $h(t)$ satisfies $3 \leq h(t) \leq 3.5$, and report the value of n you needed.

The *conservation* web page supplies the codes *euler()*, *midpoint()*, *rk2()*, *predator_deriv()*, *predator_parameters()* and *predator_conserved()*, which you will need.

Your report might be as simple as:

```
              n = 100           3 < h < 3.5
              final h?          used n = ?
euler         ----------        ----------
midpoint      ----------        ----------
rk2           ----------        ----------
```

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

9 Extra credit

It's hard to explain how the predator-prey conservation law was derived. On the other hand, once we have the formula for the conserved quantity, it's possible to show that it must be true. Do this as follows.

1. Think of the conserved quantity as $h(t)$, a function of time:

$$h(t) = H(u(t), v(t)) = u(t) - \ln(u(t)) + v(t) - 2\ln(v(t))$$

2. Compute $h'(t)$, by differentiating the right hand side:

$$h'(t) = \frac{d}{dt}\left(u(t) - \ln(u(t)) + v(t) - 2\ln(v(t))\right) = ?$$

3. Use the predator prey ODE to replace $u'(t)$ and $v'(t)$ by formulas involving $u(t)$ and $v(t)$:

$$h'(t) = \text{ formula involving } u(t) \text{ and } v(t) = ?$$

4. Use algebra to arrive at the conclusing that

$$h'(t) = 0$$

   so that $h(t)$ is a conserved quantity for the predator prey ODE.