The Euler Method for the Initial Value Problem

http://people.sc.fsu.edu/~jburkardt/isc/week10_lecture_18.pdf

.....

ISC3313:
Introduction to Scientific Computing with C++
Summer Semester 2011

.....

John Burkardt
Department of Scientific Computing
Florida State University

Last Modified: 12 July 2011

The Euler Method for the Initial Value Problem

- **Introduction**
- Approximating Change
- Example Differential Equations
- Euler's Method
- A Function for Euler's Method
- Detecting and Dealing with Error
- Lab Exercise #9

INTRO: Schedule

Next Class:

- More About Differential Equations

Assignment:

- Today: in-class lab exercise #9
- Thursday: Programming Assignment #7 is due.

INTRO: Warning

Some recent homework submissions look to me as though they have been copied from other people.

It's always permissible to talk about the problem, and to give advice about how to get something done. But when homework looks as though it has been created using cut and paste, a line has been crossed.

Please understand that your homework submissions represent your own work. You should type in every letter that appears in your program. You should be able to explain every statement in your program.

I have asked Detelina to bring any further suspicious assignm to my attention.

INTRO: Measuring Change

The name planets means *wandering stars*; since ancient times, people realized that these lights moved through the sky in a way that was different from the other stars.

Every night, a given planet would have moved slightly. By keeping records, people were able to trace the orbit, or trajectory, followed by a planet. Kepler showed that this was an ellipse.

Because a planet had a new position each night, it was possible to take the quotient of the change in position over change in time to estimate velocity.

$$\frac{d\vec{x}}{dt} \approx \frac{\vec{x}(\text{today}) - \vec{x}(\text{yesterday})}{\text{today} - \text{yesterday}} = \frac{\Delta\vec{x}}{\Delta t}$$

INTRO: Differential Equations Model Change

It's important to realize that a *velocity estimate* is computed from a change in position over a time interval, while the *velocity* is understood to be the limit of such estimates as the time interval shrinks to zero, that is, the derivative.

Using the idea of derivatives, it was possible to propose good mathematical models to describe many physical systems that change over time. Because such models involve derivatives, they are usually known as *differential equations*.

INTRO: Initial Value Problems

An initial value problem (IVP) is a differential equation that: *Describes something that changes by specifying an initial state, and giving a rule for how it changes over time.*

Thus, a simple IVP would state that at time $t = 2.0$, the value of x is 7, and that thereafter, x changes according to the rule $\frac{dx}{dt} = e^x$.

Starting with planets, the differential equation approach has been applied to chemical reactions, disease spread, economic changes, the interaction of herds of wild animals, the heating of a star core, the movement of water through Florida's underground channels.

Calculus tells us how to solve certain simple differential equations; scientific computing gives us methods for automatically produ[illegible] approximate solutions for almost any system described this way.

The Euler Method for the Initial Value Problem

- Introduction
- **Approximating Change**
- Example Differential Equations
- Euler's Method
- A Function for Euler's Method
- Detecting and Dealing with Error
- Lab Exercise #9

CHANGE: Simple IVP's

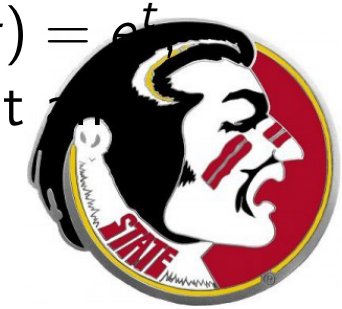
Here are some initial value problems, where we assume u(0) = 1:

$$\frac{du}{dt} = 0; \quad \frac{du}{dt} = 1; \quad \frac{du}{dt} = t;$$

$$\frac{du}{dt} = e^t; \quad \frac{du}{dt} = u; \quad \frac{du}{dt} = u * t;$$

The first four equations are easy. The fifth one is only easy if you recognize it; I am sure you have no idea whether the sixth equation has a solution or not.

In mathematics, we think of a solution as a formula. The mathematical solution to both equations #4 and #5 is $u(t) = e^t$, but the form of equation #5 makes it harder to think about [illegible] solve.



CHANGE: Approximate Solutions

The problems for which exact solutions are known are very few. If we can't find a formula to solve an initial value problem, what can we do?

Computational approximation is possible, but we need to understand the difference. We will not be getting a formula.

If we think of the graph of the solution, then a computational solution is, a sequence of values, that is points $(t1, u1), (t2, u2)$, and so on, that we hope are close to, the graph of the true solution.

Because each step depends on the previous one, we can also expect that errors gradually increase as we move from the initial data

CHANGE: An Initial Value Problem

To see what is going on, let us suppose that in 1980 a prospector discovered a cave containing 10 tons of uranium.

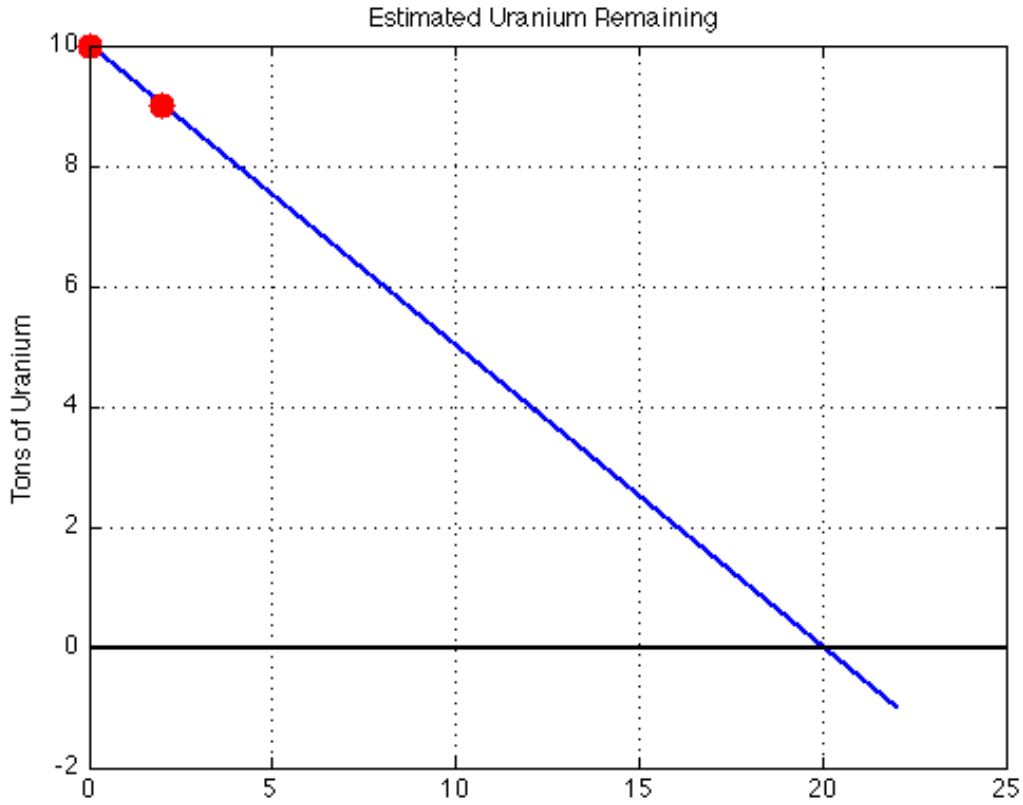
He and a friend come back in 1982, but this time only measure 9 tons. Uranium, of course, gradually decays into other elements.

He and his friend figure they must extract the uranium as soon as possible, or it will all "melt away". But they are captured by bandits and held until the year 2000. When they escape, the prospector says there's no point in returning to the cave, because if the uranium lost half a ton every year, then after 20 years it's all gone.

CHANGE: A Crude Estimate of Change

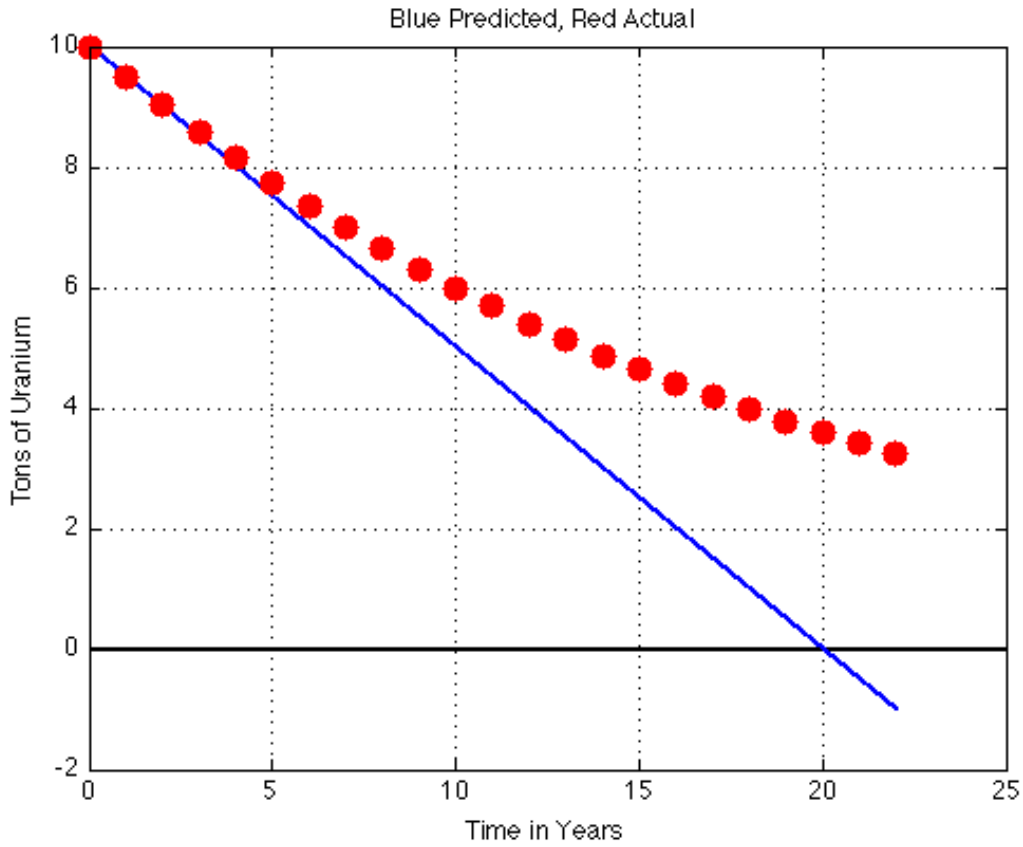
While the prospector first thought his uranium was a constant 10 tons, he recognized after the second visit that it was changing. Seeing 10 tons became 9, he estimated the rate of change:

$$\frac{\text{change in U}}{\text{change in time}} = \frac{1.0 \text{ ton}}{2 \text{ years}} = 0.5 \text{ tons per year}$$



CHANGE: The Estimate Was Way Off

However, the fact that the prospector's graph becomes negative after 20 years suggests something is wrong with the model. If the prospector had actually been able to measure the amount each year, he would have seen how his model went wrong:



CHANGE: The Actual Differential Equation

The issue here is that the prospector assumed that the rate of change could be evaluated at one time, and used forever. This is like assuming that a car's velocity will never change. An accurate model of a system must account for the fact that the rate of change might vary over time.

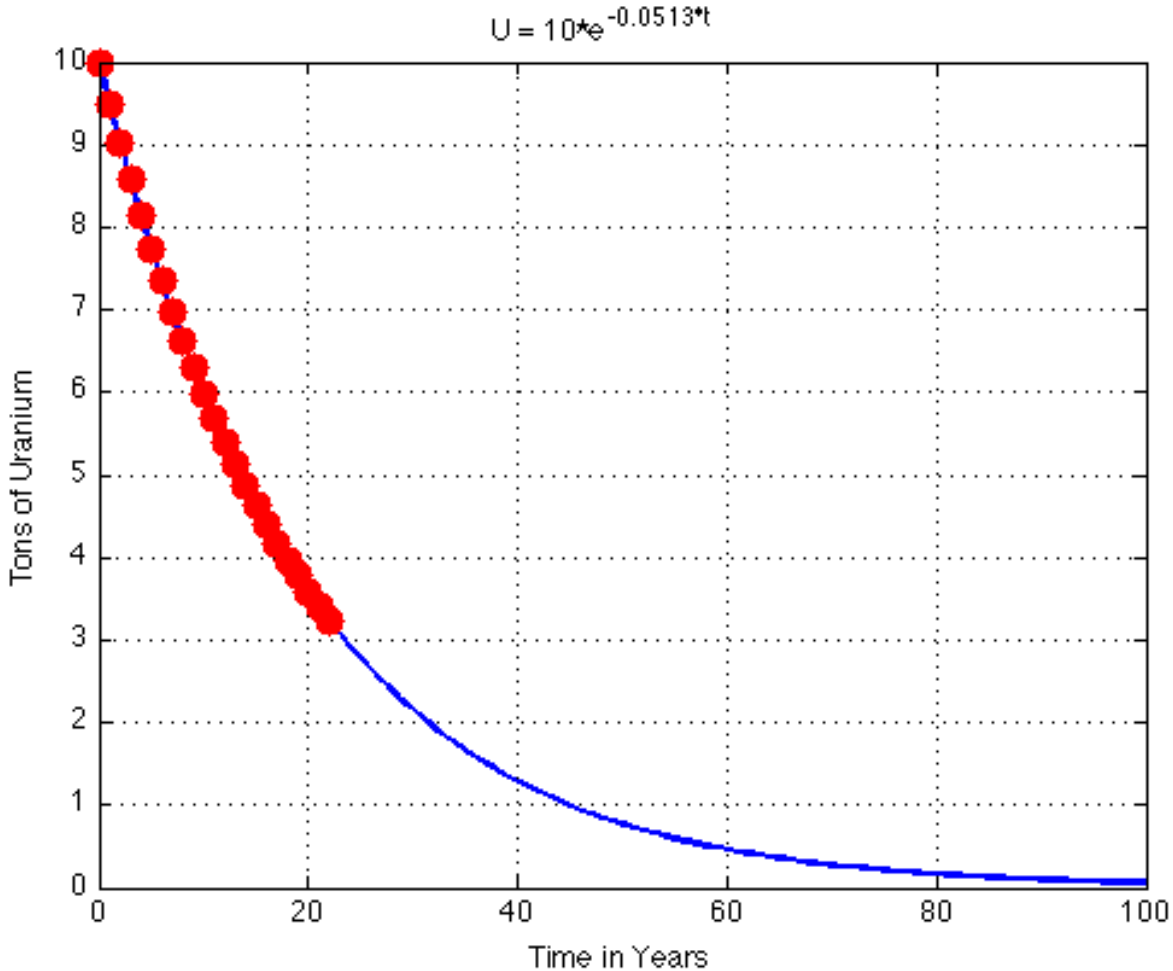
In fact, the amount of uranium that decays should be proportional to the amount of uranium we have available. If 10 tons becomes 9 in one year, then 20 tons should become 18 in a year. That means the differential equation has the form

$$\frac{du}{dt} = -0.0513 * u$$

where the coefficient -0.0513 shows the rate of decay.

CHANGE: The Change in the Rate of Change

If we plot the model over a longer span of time, you can see that the slope (the rate of change) varies a great deal. The prospector's slope of -0.5 was only a reasonable estimate initially.



CHANGE: So How Do We Do Better?

Now let's put ourselves in the prospector's place, and assume we know that u(1980) = 10, and that

$$\frac{du}{dt} = -0.0513 * u$$

How can we construct a computational solution of this problem, that is, estimates at regularly spaced times?

The prospector's guess was also a computational solution...just a bad computational solution.

Can we accurately estimate a system described by an initial value problem?

The Euler Method for the Initial Value Problem

- Introduction
- Approximating Change
- **Example Differential Equations**
- Euler's Method
- A Function for Euler's Method
- Detecting and Dealing with Error
- Lab Exercise #9

Example 1: The Baby Boom Equation

Example 1: The Baby Boom Equation

Suppose we are measuring the size of a population. In the baby boom model, we have unrestricted growth:

$$\frac{du}{dt} = c * u$$

where c measures the number of babies born per person each year. If c is positive, then in such a model the population grows with no limit.

Note that this problem is the same as the uranium decay problem, except that we assume c is positive!

Example 2: The Baby Bust Equation

Example 2: The Baby Bust Equation

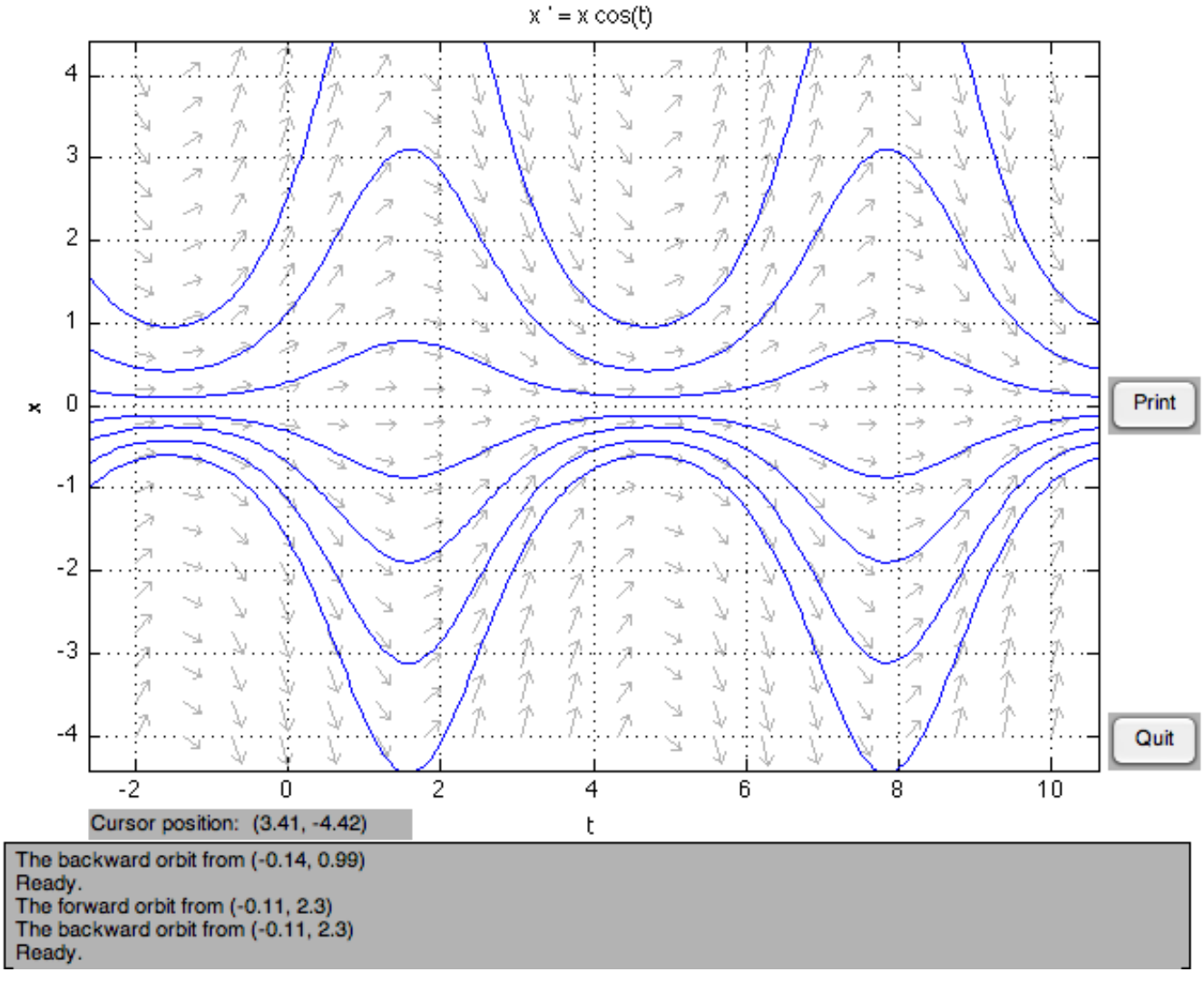
The baby boom equation is not a good model for population behavior over long times, since the population goes to infinity.

In the **baby bust** model, we add a factor that accounts for the fact that the city, island or planet we are studying can only support a maximum of *umax* people. In that case, the population can be modeled by:

$$\frac{du}{dt} = c * u * (1 - \frac{u}{umax})$$

As the population approaches *umax*, the rate slows down.

Example 3: The Wiggly Equation



Example 3: The Wiggly Equation

The derivative can depend on both u and t. We can get an idea of the behavior of the solution by plotting the derivative vector on a graph that displays t versus u. The solution is like a feather that must follow the wind whose strength and direction are indicated by the field of derivatives.

In the wiggly function, we see cosine behavior in time, but multiplication by u means the cosines get exaggerated away from the t axis.

$$\frac{du}{dt} = u * \cos(t)$$

The solution curves are hard to compute because of all the wiggling.

The Euler Method for the Initial Value Problem

- Introduction
- Approximating Change
- Example Differential Equations
- **Euler's Method**
- A Function for Euler's Method
- Detecting and Dealing with Error
- Lab Exercise #9

EULER: A Straight Line Model for a Small Step

Euler's method is the simplest approach to approximating a solution to a differential equation.

It uses the information we know to estimate unknown information that is nearby. We have the initial value u_0 at time t_0, and the derivative function $f(u, t)$ tells us how fast u_0 will change over time.

Because $f(u_0, t_0)$ is the slope of the solution curve at time t_0, so a straight line with this slope is a reasonable local approximation:

$$u(\text{nearby t}) \approx u_0 + f(u_0, t_0) * (t - t_0)$$

EULER: Step, Then Adjust the Direction

Let's decide to take a small step forward, from time t_0 to $t_1 = t_0 + \Delta t$. Then we have

$$u(t_1) \approx u_0 + f(u_0, t_0) * \Delta t = u_1$$

If our step is small enough, and the graph doesn't wiggle violently, we are still close. The direction may have changed a little, but we can check that by evaluating the derivative function at our new time. So we can try taking a second step, from t_1 to $t_2 = t_1 + \Delta t$:

$$u(t_2) \approx u_1 + f(u_1, t_1) * \Delta t = u_2$$

Continuing in this way, we can advance in time, producing a sequence of approximations to the solution at equally spaced points in time from t_0 to t_{final}.

EULER: Apply to Baby Boom Equation

To understand how this might work, let's consider the "baby boom" version of example equation #2, and specify some numbers for the initial conditions:

$$\frac{du}{dt} = 0.05 * u$$

$$t_0 = 2011$$

$$u(2011) = 10,000$$

This equation could be considered to be a model of a population that is at 10,000 in the year 2011. If it takes a couple of people to make a new child, the growth rate of 0.05 means that each year a child is born to 1 out of 10 couples (1 new baby a year for eve 0 people = 10 couples).

EULER: Use 2011 to Predict 2012

It is natural to estimate the population in 2012 as follows:

$$\begin{aligned}
\frac{u(2012)-u(2011)}{2012-2011} &= \frac{\Delta u}{\Delta t} \approx \frac{du}{dt} \\
\frac{du}{dt} &= 0.05 * u = 0.05 * 10,000 = 500 \\
u(2012) &= u(2011) + \Delta t * \frac{u(2012)-u(2011)}{2012-2011} \\
&\approx 10,000 + 1 * 500 = 10,500
\end{aligned}$$

In the last equation, everything is exact except for the 500; since the population is increasing over the year, the $\frac{du}{dt}$ is increasing as well, rather than sticking at 500.

EULER: Our Approximation Involves Choices

Keep in mind that the "magic" of being able to predict the future depends on a lot of things that are only approximately true!

If the growth of a human population can be exactly calculated by a continuous curve (which means we sometimes have fractions of people!) and that curve can be described by a formula that is differentiable, and we can work out the parameters (such as the 0.05 growth rate), then the exact solution of the differential equation gives us the future population.

In order to work computationally, we make one more approximation, that if we take a small enough step in time, the derivative won't change that much, so we can simply use its v[illegible] at the starting time as though it were constant for the whole ste[illegible]

EULER: Stepping Forward On Paper

Once we've got the population in 2012, we can take another step. However, we need to update our estimate for the derivative when we do that:

```
u(2011)                                  = 10,000
u'(2011)                                 =   +500.0

u(2012) = u(2011) + 1 * u'(2011) = 10,500
u'(2012) = 0.05 * u(2012)                =   +525

u(2013) = u(2012) + 1 * u'(2012) = 11,025
u'(2013) = 0.05 * u(2013)                =   +551.25

u(2014) = u(2013) + 1 * u'(2013) = 11,576.25
u'(2014) = 0.05 * u(2014)                =   +578.81

u(2015) = u(2014) + 1 * u'(2014) = 12,155.06
```

EULER: The Effect of the Time Step

What if we took time steps half as large? We'll write "2011.5" to mean July 1, 2011:

```
u(2011)                                          = 10,000
u'(2011)                                         =   +500.0

u(2011.5) = u(2011) + 0.5 * u'(2011)    = 10,250
u'(2011.5) = 0.05 * u(2011.5)                    =   +512.5

u(2012) = u(2011.5) + 0.5 * u'(2011.5) = 10,506.25
u'(2012) = 0.05 * u(2012)                        =   +525.31
```

So taking steps of 6 months instead of a year, we picked up 6.25 extra births. The difference between a rate of 500 and a rate of 512 isn't extreme, so in this case, our one-year time step is probably small enough, as long as we don't go too far into the future.

The Euler Method for the Initial Value Problem

- Introduction
- Approximating Change
- Example Differential Equations
- Euler's Method
- **A Function for Euler's Method**
- Detecting and Dealing with Error
- Lab Exercise #9

EULER.CPP: How Can We Compute This?

If we have an initial value problem, and we want to use a C++ implementation of Euler's method to get a solution, what things do we have to consider?

The overall design of the Euler function is the first question. It seems simplest to have it start at the initial point $t0$, and to produce an approximate solution $u1$ at the time $t1 = t0 + dt$.

If we want to take another step, we use $(t1, u1)$ as our new starting time and starting solution, set $t2 = t1 + dt$ and call the function again.

By repeated calls, we can get a sequence of approximate soluti values, which we can think of as points on the solution curve

EULER.CPP: Designing an Euler Function

If we do it this way, the input to the Euler function could be:

- **t0**, the starting time;
- **u0**, the starting value;
- **dt**, the stepsize;
- **f(t,u)**, a function that evaluates the derivative.

and what emerges as the function value is **u1**, the approximate solution at time $t1$.

So our function will be declared as

```
double euler ( double t0, double u0, double dt,
  double f ( double t, double u ) );
```

EULER.CPP: The Euler Code

It only takes a simple code to start from **u0** at **t0**, and take a step of size **dt** in the direction specified by the derivative:

```
double euler ( double t0, double u0, double dt,
  double f ( double t, double u ) )
{
  double u1;

  u1 = u0 + dt * f ( t0, u0 );

  return u1;
}
```

EULER.CPP: The Baby Boom Derivative Function

Now let's write the function for the baby boom example, which has input of the current time **t** and solution $\mathbf{u} = u(t)$, and must evaluate the derivative formula:

```
double f1 ( double t, double u )
{
  double dudt;

  dudt = 0.05 * u;

  return dudt;
}
```

EULER_F1.CPP: A Program For The Baby Boom Equation

And now we are able to write a program that tries to compute the population from 2011 to 2020:

```
# include <cstdlib>
# include <iostream>
using namespace std;

double euler ( double t0, double u0, double dt, double f ( double t, double u ) );
double f1 ( double t, double u );

int main ( )
{
  double dt = 1.0, t0 = 2011, t1, tmax = 2020, u0 = 10000, u1;  <-- Initial data

  while ( true )
  {
    cout << t0 << "  " << u0 << "\n";
    if ( tmax <= t0 )                                  <-- Did we reach our goal?
    {
      break;
    }
    t1 = t0 + dt;                                      <-- Take another step.
    u1 = euler ( t0, u0, dt, f1 );

    t0 = t1;                                           <-- Shift data for next loop.
    u0 = u1;
  }
  return 0;
}
```

. . . Text of "euler.cpp" and "f1.cpp" follows. . .

EULER.CPP: Table of Data

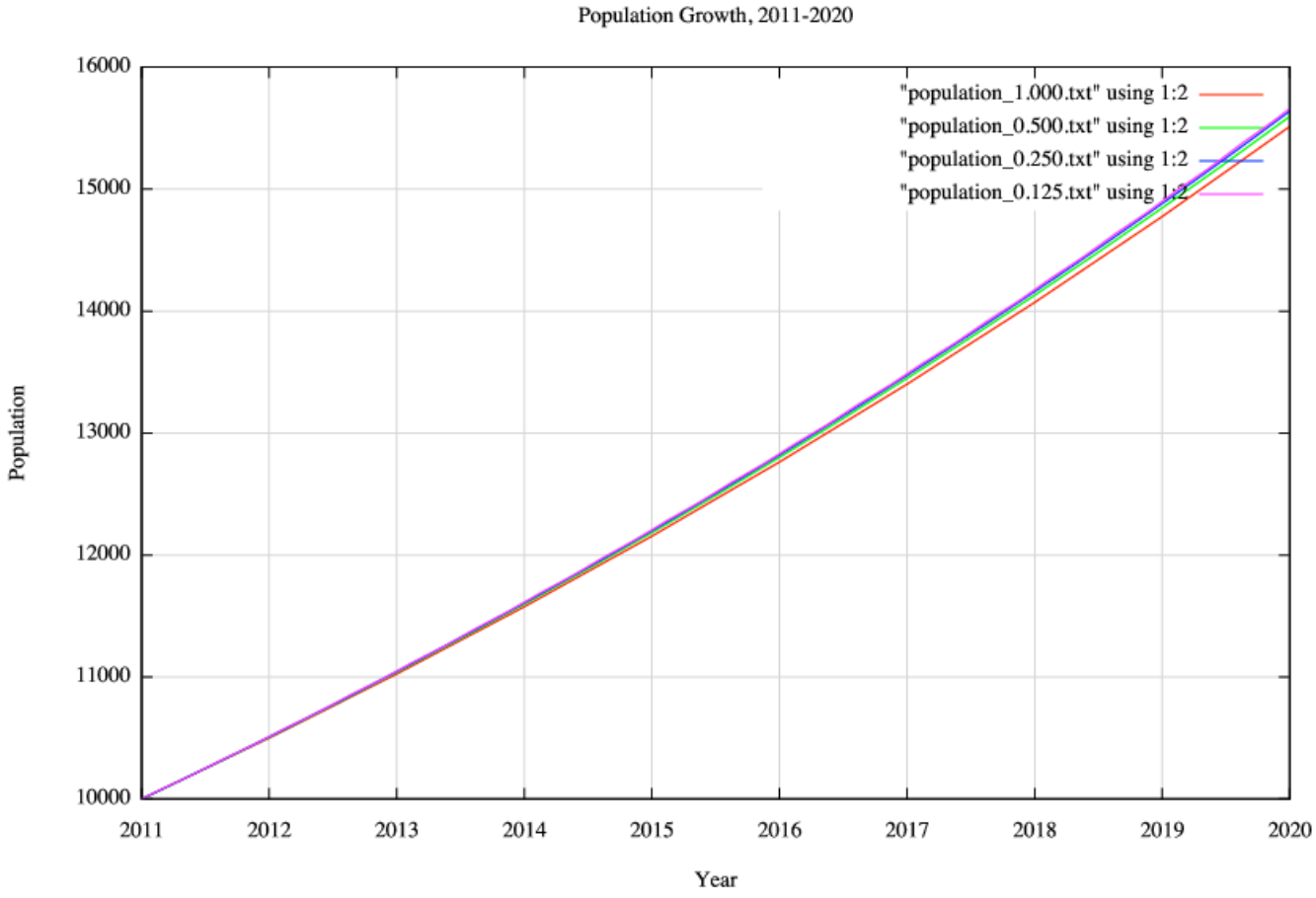
Our function returns the following computed answers:

```
   T      U(T)

2011   10000
2012   10500
2013   11025
2014   11576.2
2015   12155.1
2016   12762.8
2017   13401
2018   14071
2019   14774.6
2020   15513.3
```

EULER.CPP: Plots of Smaller Timesteps

Compare results for timesteps of $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ of a year.



Results are good, perhaps because the solution is not very "wiggly".

The Euler Method for the Initial Value Problem

- Introduction
- Approximating Change
- Example Differential Equations
- Euler's Method
- A Function for Euler's Method
- **Detecting and Dealing with Error**
- Lab Exercise #9

ERROR: How Can We Detect Errors?

How can we trust our results when we have no way of knowing the correct answer? We can't guarantee that we won't accept a bad answer, but we can certainly watch some simple warning signs, and try some easy remedies.

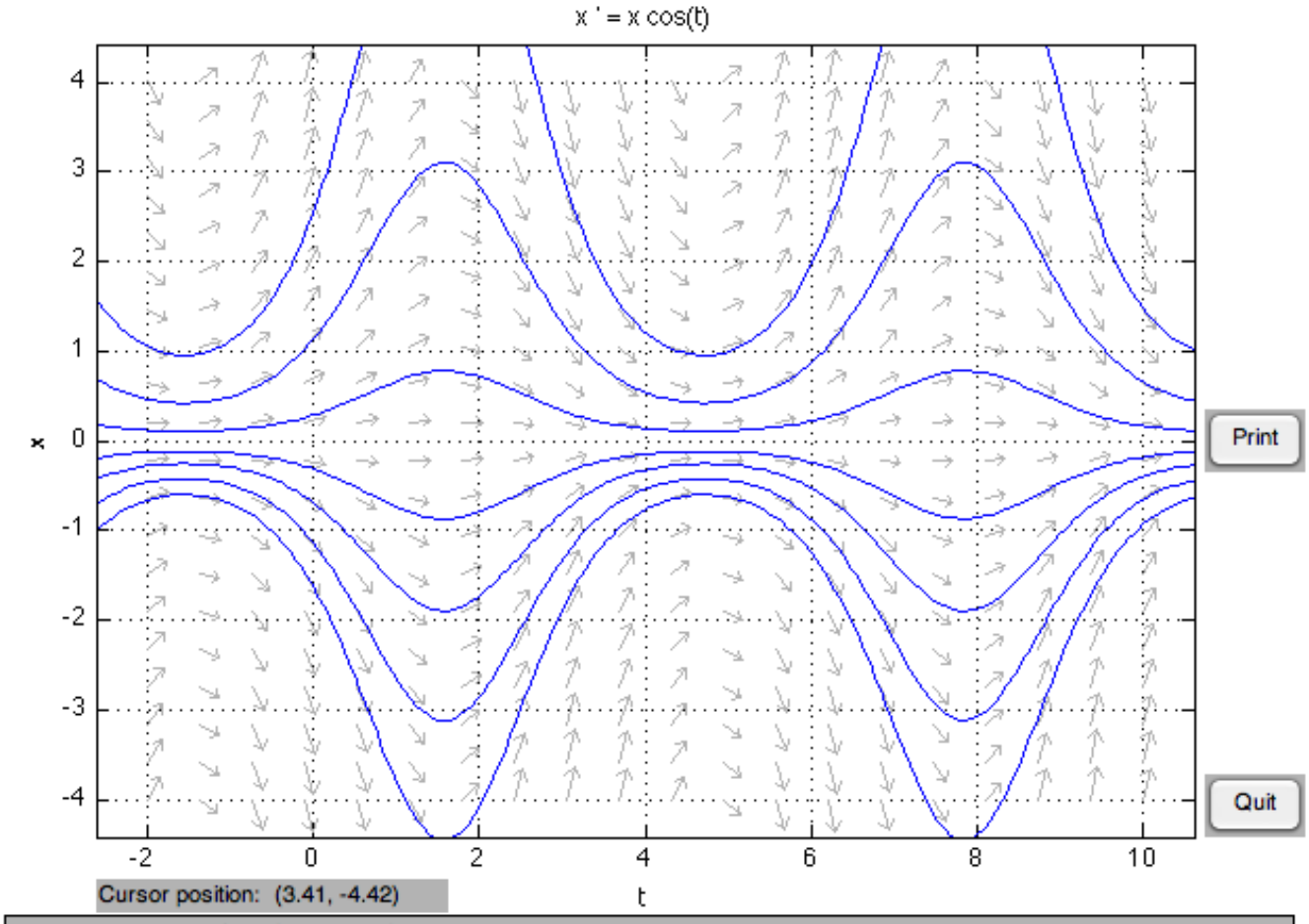
The best error detection we have is available because, generally, our solution estimate improves as we reduce the step size.

Thus, we tended to trust our baby boom calculation, because the results for timesteps of $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ of a year looked very close.

Let's try this idea out on the "wiggly function".

ERROR: The Wiggly Equation

Recall that the wiggly function has a direction field with many bends!



EULER.CPP: A Function for the Wiggly Derivative

Now let's write the function for the wiggly example:

```
double f3 ( double t, double u )
{
  double dudt;

  dudt = u * cos ( t );

  return dudt;
}
```

Our program needs **# include** <**cmath**> because we call **cos**

ERROR: The Wiggly Equation

Let's try to solve the following problem for $0 \leq t \leq 12 * \pi$:

$$\begin{aligned} \frac{du}{dt} &= u * \cos(t) \\ t_0 &= 0 \\ u(t_0) &= 0.5 \end{aligned}$$

We'll use step sizes of $dt = 0.25$, 0.1 and 0.01.

We make a copy of **euler_f1.cpp** called **euler_f3.cpp**, replace the function **f1()** by **f3()**, and modify the initial data.

ERROR: Comparing Approximate Solutions

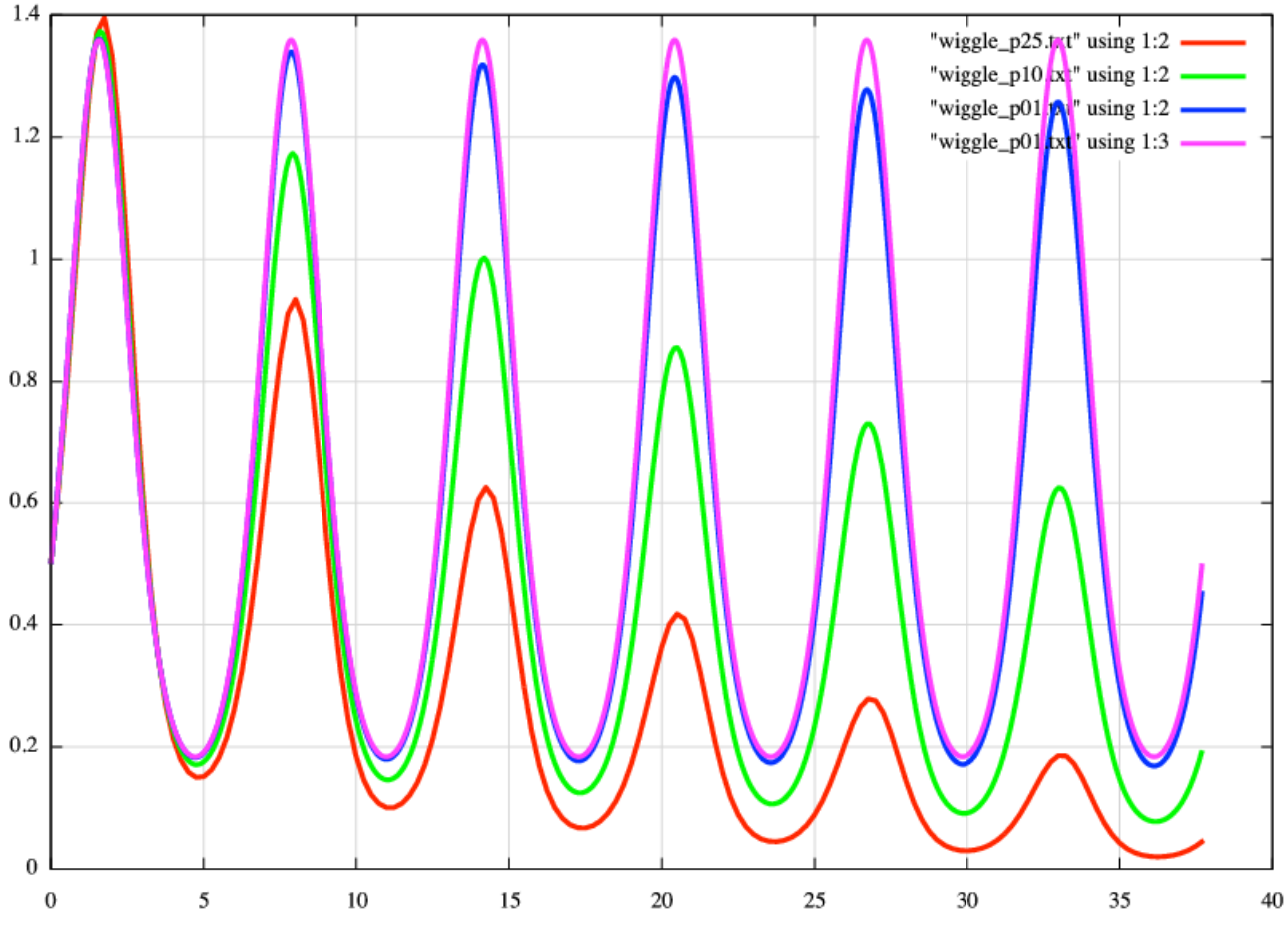
When we run our **euler_f3.cpp** program for different time steps, we concentrate on a few selected points:

T	U 0.25	U 0.10	U 0.01
0.0	0.500000	0.500000	0.500000
10.0	0.185518	0.244324	0.285323
20.0	0.363628	0.769275	1.18746
30.0	0.029936	0.091483	0.173494

The values do not agree! While we can guess that the results f
$dt = 0.01$ are the most accurate, we really can't be sure.

ERROR: Approximations versus Exact

Even our (blue) computation with $dt = 0.01$ has a noticeable error when compared to the (purple) exact solution!



ERROR: Wiggles Require Small Steps!

The good news is that comparing results for different step sizes did give us a warning that something was wrong.

The unfortunate fact is that, because the Euler method uses straight line segments to approximate a curve, it's not very good at going around bends, which the wiggly function is full of, and so the only remedy for a wiggly curve is to keep reducing the step size.

We will look, next time, at methods that are better at going around the bends without losing accuracy!

The Euler Method for the Initial Value Problem

- Introduction
- Approximating Change
- Example Differential Equations
- Euler's Method
- A Function for Euler's Method
- Detecting and Dealing with Error
- **Lab Exercise #9**

EXERCISE: The Baby Bust Equation

The "baby bust" equation is example #2.

We are going to estimate the behavior of a population that can be described by this equation. To do so, we need to know:

- **t0**, the initial time, is 1850;
- **tmax**, the final time, is 2000;
- **u0**, the initial population, is 2
- **c**, the growth rate, is 0.10;
- **umax**, the maximum population, is 750
- **dt**, the stepsize, will be 1.0

Make a copy of **euler_f1.cpp** for this problem, calling the new file **euler_f2.cpp**. Modify the function that evaluates the right hand side. Change the initial data in the main program.

EXERCISE: The Baby Bust Equation

Compile and run your program, and save your output in a file called "pop.txt".

```
g++ euler_f2.cpp
mv a.out euler_f2
euler_f2 > pop.txt
```

Plot your data using the commands:

```
gnuplot
  plot "pop.txt" using 1:2 with lines
```

Show your plot to Detelina so you can get credit for the exercise!