11 Numerical methods for first-order differential equations

(This will probably take 1.5 classes to cover.)

11.1 Goals

1. Be able to compute approximate solutions by hand using Euler’s method.

2. Be able to compute the concavity of a solution and say whether Euler’s method gives an over or under-estimate,

3. Know some of the ways numerical methods can fail or give misleading results

4. Know the broad outline of how other numerical methods work and understand that many of them are really fancier versions of Euler’s method.

11.2 Introduction

In this topic we will look at numerical methods for approximating solutions to differential equations. Just like numerical integration this allows us to approximate the solution to any first-order DE. It is especially valuable for those equations that we can’t solve analytically. Using the computer we can then study as many solutions as we want for a given DE.

11.3 Euler’s Method of numerical approximation

Our first method will be Euler’s method, which is like using Riemann sums over rectangles to approximate an integral. Euler’s method is very simple to compute and is the only numerical method we will compute by hand.

Just as in numerical integration there are fancier numerical methods for solving DEs. These methods require more computation than Euler’s and we will leave the computation to computers and existing software packages.

Warmup. Consider the following triangle showing the slope as the rise/run (actually showing: rise = slope \times run).

![Diagram of Euler's method](image)
In the diagram to 'step' from \((x_0, y_0)\) to \((x_1, y_1)\) we have

\[ x_1 = x_0 + h; \quad y_1 = y_0 + mh \]

Now we’ll relate this to the 18.01 tangent line approximation formula. Suppose \(y = y(x)\) is a function of \(x\). Then

\[ y(x_0 + \Delta x) \approx y(x_0) + y'(x_0)\Delta x \]

(This comes from the definition of the derivative: \(y'(x_0) \approx \frac{\Delta y}{\Delta x} = \frac{y(x_0 + \Delta x) - y(x_0)}{\Delta x}\).)

We can simplify the way this looks by letting \(y_0 = y(x_0)\) and \(h = \Delta x\). So,

\[ y(x_0 + h) \approx y_0 + y'(x_0)h. \] (1)

**Equation 1 is the key to all our numerical methods for solving differential equations.**

The next example illustrates how to use Equation 1 in Euler’s method.

**Example 11.1.** Numerically solving an IVP using Euler’s method. Consider the IVP \(y' = x^2 + y^2; \quad y(0) = 2\). Use Euler’s method to estimate \(y(1)\).

**answer:** We don’t know \(y(x)\) (and it’s hard to find), but we do know the slope at each point and we can use the formula in Equation (1).

Pick a **stepsize:** \(h = .5\)

<table>
<thead>
<tr>
<th>Step 0:</th>
<th>(x_0 = 0)</th>
<th>(y_0 = 2)</th>
<th>(m = f(x_0, y_0) = 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1:</td>
<td>(x_1 = x_0 + h = .5)</td>
<td>(y_1 = y_0 + mh = 4)</td>
<td>(m = f(x_1, y_1) = 16.25)</td>
</tr>
<tr>
<td>Step 2:</td>
<td>(x_2 = x_1 + h = 1)</td>
<td>(y_1 = y_1 + mh = 12.125)</td>
<td></td>
</tr>
</tbody>
</table>

So \(y(1) \approx 12.125\)

---

**Example 11.2.** \(y' = y; \quad y(0) = 1\). Estimate \(y(1)\)

(Note: we know the exact answer, \(y = e^x, \quad y(1) = 2.718\ldots\) )

Let \(h = .25\), so there are 4 steps from 0 to 1. We organize the calculation in a table:
Example 11.3. (Example continued.) We now continue the previous example with different step sizes. In all cases we are trying to estimate $y(1)$.

**Stepsize. $h = 1$** (this is just to be a little silly).

With $h = 1$ it takes 1 step to go from 0 to 1.0

<table>
<thead>
<tr>
<th>$n$</th>
<th>$x_n$</th>
<th>$y_n$</th>
<th>$m = f(x_n, y_n)$</th>
<th>$mh$</th>
<th>actual</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.0</td>
<td>1.0</td>
<td>.5</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>.25</td>
<td>1.25</td>
<td>1.25</td>
<td>.31</td>
<td>1.28</td>
<td>.03</td>
</tr>
<tr>
<td>2</td>
<td>.5</td>
<td>1.56</td>
<td>1.56</td>
<td>.39</td>
<td>1.65</td>
<td>.09</td>
</tr>
<tr>
<td>3</td>
<td>.75</td>
<td>1.95</td>
<td>1.95</td>
<td>.49</td>
<td>2.12</td>
<td>.17</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>2.44</td>
<td></td>
<td></td>
<td>2.7183</td>
<td>.28</td>
</tr>
</tbody>
</table>

**Notes:**
1. Organize hand calculations like this.
2. Error often accumulates.

**Stepsize. $h = .1$.**

With $h = .1$ it takes 10 steps to go from 0 to 1.0. Here is the table with some of the numbers left out.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$x_n$</th>
<th>$y_n$</th>
<th>$m = f(x_n, y_n)$</th>
<th>$mh$</th>
<th>actual</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.1</td>
<td></td>
<td></td>
<td></td>
<td>.1224</td>
<td>.011</td>
</tr>
<tr>
<td>2</td>
<td>.2</td>
<td>1.21</td>
<td></td>
<td></td>
<td>1.2214</td>
<td>.028</td>
</tr>
<tr>
<td>3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
<td>1.4918</td>
<td>.05</td>
</tr>
<tr>
<td>4</td>
<td>.4</td>
<td>1.4641</td>
<td></td>
<td></td>
<td>1.8221</td>
<td>.05</td>
</tr>
<tr>
<td>5</td>
<td>.5</td>
<td></td>
<td></td>
<td></td>
<td>2.2255</td>
<td>.082</td>
</tr>
<tr>
<td>6</td>
<td>.6</td>
<td>1.7716</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>.8</td>
<td>2.1436</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>2.5937</td>
<td></td>
<td></td>
<td>2.7183</td>
<td>.125</td>
</tr>
</tbody>
</table>

**Note.** The error is smaller when $h = .1$ than when $h = .25$

**Rules of thumb:** Using a smaller $h$ is more accurate but requires more computation.

**Mild warning.** More computation means more risk of roundoff error. In this class we never make $h$ so small that this is a problem.

### 11.4 What can go wrong

In this section we’ll see that numerical methods can sometimes give misleading results. We hasten to add that numerical methods provide an incredibly powerful tool which is used all the time with great success. But we do need to take some care to avoid certain pitfalls.

We expect that decreasing the stepsize should give a more accurate estimate. The next example shows that we shouldn’t simply accept the result, no matter how small the stepsize used.
Example 11.4. Consider the IVP \( y' = y^2; \ y(0) = 1 \). Use Euler’s method to approximate \( y(1) \).

**answer:** We know the exact solution is \( y = \frac{1}{1-x} \), so \( y(1) = \infty \). But, Euler’s method will happily estimate \( y(1) \). We do this for several different step sizes.

Take \( h = 0.2 \):

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x_n )</th>
<th>( y_n )</th>
<th>( m = f(x_n, y_n) )</th>
<th>( mh )</th>
<th>actual</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>...</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>.2</td>
<td>1.2</td>
<td>...</td>
<td>...</td>
<td>1.25</td>
<td>.05</td>
</tr>
<tr>
<td>2</td>
<td>.4</td>
<td>1.49</td>
<td>...</td>
<td>...</td>
<td>1.67</td>
<td>.18</td>
</tr>
<tr>
<td>3</td>
<td>.6</td>
<td>1.93</td>
<td>...</td>
<td>...</td>
<td>2.5</td>
<td>.57</td>
</tr>
<tr>
<td>4</td>
<td>.8</td>
<td>2.68</td>
<td>...</td>
<td>...</td>
<td>5</td>
<td>2.32</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>4.11</td>
<td>...</td>
<td>...</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
</tbody>
</table>

So \( y(1) \approx y_5 = 4.11 \).

For decreasing values of \( h \) we get the following:

For \( h = 0.1 \), \( y(1) \approx y_{10} = 37.6 \)

For \( h = 0.05 \), \( y(1) \approx y_{20} = 91.25 \)

For \( h = 0.025 \), \( y(1) \approx y_{40} = 238.21 \)

Instead of settling down to a limiting value, as we decrease \( h \) the estimate grows. This is a sign that something is wrong with our estimates.

11.4.1 Lesson

You should try smaller and smaller \( h \) until the answer settles down. That is run the estimate with stepsize \( h \). The rerun it with stepsize \( h/2 \). If the estimates are very close then we have one good bit of evidence to accept the estimate as a good approximation. Otherwise try \( h/4 \) etc. If the estimate never settles down then we will have to reject the estimates and use other methods.

The computer doesn’t eliminate the need to think!

**Note.** We could make the previous example even more extreme by asking to estimate \( y(2) \). The problem is that with the vertical asymptote at \( x = 1 \) the solution is not even defined at \( x = 2 \). Nonetheless, for any stepsize \( h \) Euler’s method will produce an estimate of \( y(2) \).

Example 11.5. Stepping across region boundaries. The following shows another risk in using numerical methods. Consider the IVP \( y' = y^2; \ y(-2.5) = -2.5 \).

The blue curve is the exact solution to the IVP. It goes asymptotically to \( y = 0 \)
The orange curve is the Euler approximation using stepsize \( h = 0.5 \). It goes off to infinity.

The problem is that the first step in the approximation went past the separatrix \( y = 0 \). After that instead of going asymptotically to 0 the approximation continued to grow.
11.5 Other numerical techniques

All the techniques that we’ll look at take steps of the form

\[ x_{n+1} = x_n + h; \quad y_{n+1} = y_n + mh. \]

where \( m \) is some sort of average slope near \((x_n, y_n)\). The differences between the various methods are in how \( m \) and possibly \( h \) is chosen at each step. We’ll only touch on this briefly.

**Improved Euler (also called RK2).** This is a fixed stepsize algorithm, that is we fix the value of \( h \) before using it. Here is the algorithm:

1. Start at \((x_n, y_n)\)
2. Compute the slope \( k_1 = f(x_n, y_n) \) and take a regular Euler step to a temporary point \((x_a, y_a)\).
   \[ x_a = x_n + h; \quad y_a = y_n + k_1h. \]
3. Compute the slope at \((x_a, y_a)\): \( k_2 = f(x_a, y_a) \).
4. Average the two slopes: \( m = (k_1 + k_2)/2 \).
5. Use \( m \) as the slope to take the Improved Euler step.
   \[ x_{n+1} = x_n + h; \quad y_{n+1} = y_n + mh. \]

**Runga-Kutta 4 (RK4).** This is also a fixed stepsize algorithm. You can read about it in Edwards and Penney §6.3 or do a web search to get the details. In brief, the algorithm computes 4 different slopes \( k_1, k_2, k_3, k_4 \) and then takes a weighted average of these slopes to get \( m \). There are different ways to choose the \( ks \) and the weights, one common scheme (used in E&P) is

\[
\begin{align*}
  k_1 &= f(x_n, y_n); & k_2 &= f(x_n + h/2, y_n + k_1h/2); \\
  k_3 &= f(x_n + h/2, y_n + k_2h/2); & k_4 &= f(x_n + h, y_n + k_3h) \\
  m &= \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}.
\end{align*}
\]
Then as usual, 
\[ x_{n+1} = x_n + h; \ y_{n+1} = y_n + mh. \]

**Variable step size methods.** There is no reason we have to have a fixed stepsize. It is possible to adjust \( h \) at each step. One way to do this is the following: Suppose we get to \((x_n, y_n)\) with current stepsize \( h \).

1. Take one RK4 step with stepsize \( h \).
2. Repeat with stepsize \( h/2 \) and \( 2h \).
3. If the 3 results are very close then change the current stepsize to \( 2h \) and take the step.
   If they are not close then change the current stepsize to \( h/2 \) and take the step.

Thus sometimes the stepsize will get bigger and save computation. When needed to maintain accuracy it will get smaller.

### 11.6 More technical discussion on error size

(This section is for enrichment only. You will not be asked about this on exams.)

For this discussion we fix a first order IVP: \( y' = f(x, y); \ y(x_0) = y_0 \). We also fix the value \( x_f \) and ask to approximate \( y(x_f) \).

**Euler’s method is linear in the error.** This means that the error is roughly proportional to \( h \). So, if you halve the stepsize then you approximately halve the error. Of course you also double the amount of computation.

**Improved Euler is quadratic in the error.** This means that the error is roughly proportional to \( h^2 \). So, if you halve the stepsize then the error is approximately quartered.

**RK4 is a fourth order method.** This means that the error is roughly proportional to \( h^4 \). So, if you halve the stepsize then the error is approximately multiplied by \( 1/16 \).

### 11.7 Second derivative and concavity

If we know \( y' = f(x, y) \) then we can find \( y'' \). This can be used to determine the concavity of the integral curve and thus whether the Euler estimate is an over- or under- estimate.

**Example 11.6.** Assume \( y' = 3xy \) and \( y(1) = 2 \). Use Euler’s method to estimate \( y(1.1) \). Is the estimate too high or too low?

**answer:** First: \( y'(1) = 6 \).

Now fix the stepsize \( h = .1 \).

The Euler estimate is \( y(1.1) \approx 2 + .1 \cdot 6 = 2.6 \).

To find the concavity we compute the second derivative. (Note well that \( y \) is a function of \( x \) so, for example, \((y^2)' = 2yy'\).)

\[
y'' = (xy)' = y + xy', \ \text{so} \ y''(1) = y(1) + 1 \cdot y'(1) = 2 + 6 = 8 > 0.
\]

We see that \( y \) is concave up at \( x = 1 \) and therefore the Euler estimate is too low.
11.8 Relation to numerical integration

(This section is also just for your enjoyment and enrichment. We won’t discuss it in class or on psets or exams.)

Even in 18.01 you were solving (simple) differential equations. A typical 18.01 integration question is to compute $\int_a^b f(x) \, dx$. We can rephrase this as the following initial value problem:

Let $y(x)$ be the solution to the IVP $y' = f(x); y(a) = 0$. What is $y(b)$?

It is clear that this has solution $y(b) = \int_a^b f(x) \, dx$.

Thus for this IVP estimating $y(b)$ with numerical methods amounts to estimating the definite integral using numerical methods. More precisely

- Euler’s method = numerical integration using left Riemann sums with rectangles.
- Improved Euler = numerical integration using the trapezoidal rule.
- RK4 = numerical integration using Simpson’s rule.

**Example 11.7.** (Euler’s method = left Riemann sum.) For $y' = f(x), y(a) = 0$ estimate $y(b)$ using Euler’s method and $N$ steps.

**answer:** $N$ steps implies the stepsize is $h = \frac{b-a}{N}$. Thus Euler’s method gives

$$y_{n+1} = y_n + f(x_n) \, h.$$ 

This leads to the following table:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$x_n$</th>
<th>$y_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$a$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$a+h$</td>
<td>$f(x_0) , h$</td>
</tr>
<tr>
<td>2</td>
<td>$a+2h$</td>
<td>$f(x_0) , h + f(x_1) , h$</td>
</tr>
<tr>
<td>3</td>
<td>$a+3h$</td>
<td>$f(x_0) , h + f(x_1) , h + f(x_2) , h$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N$</td>
<td>$a+Nh = b$</td>
<td>$f(x_0) , h + (f(x_1) + f(x_2) + \ldots + f(x_{N-1})) , h$</td>
</tr>
</tbody>
</table>

Thus our approximation is $y(b) = \sum_{j=0}^{N-1} f(x_j) \, h$. In 18.01 you might have learned to use $\Delta x$ instead of $h$. In either case the approximation is the left Riemann sum approximating $\int_a^b f(x) \, dx$. 

---