Introduction to Scientific Computing
Lecture 6

Professor Hanno Rein

Last updated: October 22, 2018

1 Finite Differences

There are many cases where one might not be able to write down an analytic function for function's derivate. In that case one can try to calculate the derivate numerically using what is called finite differences. We can approximate derivative $f'(x)$ by the expression

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h).$$

Mathematically, this expression is exactly equal to the derivate in the limite $h \to 0$. On a computer we need to use a finite value of h.

Which h to choose is difficult to answer in general. The smaller it is the closer the estimate to the real value of the derivative. However, the h has to remain finite and if it gets too small, we might run into floating point issues. This is an important example of why it it important to understand the limitations of floating point numbers. The term $f(x+h)$ and $f(x)$ are almost identical if h is small. Thus, calculating the difference will result in a large floating point error.

The expression above is a one sided finite difference. We can also make a central difference:

$$f'(x) = \frac{f(x+0.5h) - f(x-0.5h)}{h} + O(h^2),$$

which has often better properties.

The same procedure can be used to calculate higher order derivatives. For example, the second derivate can be written as

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2),$$

In general, one can approximate the n-th derivate with a central difference as follows:

$$\frac{d^n f}{dx^n}(x) = \frac{1}{h^n} \sum_{i=0}^{n} (-1)^i \binom{n}{i} f\left(x + \left(\frac{n}{2} - i\right) h\right) + O(h^2).$$

2 Finite Difference Method

We will now make use of the fact that we can write down approximations to derivatives to solve a differential equation. This way of solving differential equations is called the finite difference method. We will later come back to differential equations when we solve them using other methods.

Here, we will focus exclusively on the heat equation:

$$\frac{d}{dt}U(x,t) = \frac{d^2}{dx^2}U(x,t)$$

This equation describes how heat gets transported. For simplicity, we'll assume x is one dimensional, e.g. a rod.

The differential equation describes how the temperature changes in time and space. Let's us use the finite differences from above. We'll use h to describe the separation in time, and k for the separation in space. Upper indicies correspond to time, lower indicies correspond to space. We get

$$\frac{u_j^{n+1} - u_j^n}{k} = \frac{u_{j+1}^n - 2u_j^n k + u_{j-1}^n}{h^2}$$

We can slightly modify the equation to get

$$u_j^{n+1} = (1-2r)u_j^n + ru_{j-1}^n + ru_{j+1}^n$$

where $k = k/h^2$. This means, we can now calculate the new temperature at time $n+1$ using only the temperature at time n. We also need boundary conditions. Here, we use $U(0,t) = U(1,t) = 0$. Let's code this up!

3 Integration

Note that geometric integration methods of differential equations are closely related to solving standard integrals.

We start with the simplest case, an integral of a function of a single variable over a finite range. We can then generalize the concepts to higher dimensions and come up with more accurate algorithms.

3.1 Trapezoidal Rule

Suppose we have the following integral

$$I(a,b) = \int_a^b f(x)dx$$

This is the equivalent of calculating the area under the curve. Sometimes, it is possible to perform the calculation exactly, but not in all cases. If it is not possible, we need to use numerical methods to approximate the integral. One way to do that is to approximate the area by dividing it up into multiple rectangles, calculate the area for each, then adding them up. This would be a poor approximation. We can do better without any extra work. The idea is to replace the rectangles with trapezoids. The area under the trapezoids is a better approximation of the area under the curve.

Let us divide the interval from a To b into N slices with width

$$h = \frac{b-a}{N}.$$

Then, the area of the k-th slice is

$$A_k = \frac{1}{2}h\left[f(a+(k-1)\cdot h) + f(a+k\cdot h)\right].$$

We can then calculate the integral as a sum of the individual trapezoids

$$\begin{aligned} I(a,b) \approx \sum_{k=1}^{N} A_k &= \frac{1}{2}h\sum_{k=1}^{N}\left[f(a+(k-1)\cdot h) + f(a+k\cdot h)\right] \\ &= h\cdot\left[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + \frac{1}{2}f(b)\right] \\ &= h\left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N-1} f(a+k\cdot h)\right] \end{aligned}$$

Note that we've simplified the trapezoidal rule significantly in these three lines. The last line is calculating the rectangles under the curve. We argued previously that the rectangles are not a good approximation, so why do they appear here again? Note that the boundaries are different! This small change makes all the difference. Of course, for large N this difference will become smaller and smaller and the trapezoidal rule is nothing else than the rectangle rule.

As an example, let us try to integrate

$$\int_0^2 (x^4 - 2x + 1)dx$$

The true value is 4.4. Let us implement the trapezoidal rule using python.

```
def f(x):
    return x**4 - 2*x + 1

N = 10
a = 0.
b = 2.
h = (b-a)/N
s = 0.5*f(a) + 0.5*f(b)
for k in range(1,N):
    s += f(a+k*h)

print(h*s)
```

Code 1: Python code of the trapezoidal rule

The output of this code is 4.50656. Only about 2% away from the true value. Obviously, we can increase the accuracy by changing N.

3.2 Simpson's rule

The trapezoidal rule only takes a few lines of code. It uses effectively a linear interpolation between data-points. You might be able to guess what comes next. We will expand the idea to higher order. This is what Simpson's rule does. It uses quadratic curves.

To define a quadratic function, we need three points instead of two. Suppose that as before, we split the interval into N slices with $h = (b-a)/N$. And suppose, for this argument, we have the points at $x = -h$, $x = 0$, and $x = h$. Then we can try to fit the second order polynomial of the form

$$Ax^2 + Bx + C$$

to those point via

$$f(-h) = Ah^2 - Bh + C \qquad f(0) = C \qquad f(h) = Ah^2 + Bx + C$$

Solving these equations gives:

$$\begin{aligned} A &= \frac{1}{h^2}\left[\frac{1}{2}f(-h) - f(0) + \frac{1}{2}f(h)\right] \\ B &= \frac{1}{2h}\left[f(h) - f(-h)\right] \\ C &= f(0) \end{aligned}$$

We can easily integrate the second order polynomial to get

$$\int_{-h}^{h}(Ax^2+Bx+C)dx=\frac{2}{3}Ah^3+2Ch=\frac{1}{3}h\left[f(-h)+4f(0)+f(h)\right]$$

This is Simpson's Rule! Note that the final formula only includes h and f evaluated at the grid points. We do not need to perform a quadratic fit every time! It makes Simpson's rule almost as simple as the trapezoidal rule.

Applying Simpson's rule involves slicing up the interval into uniformly spaced points, as before, and then applying the above formula for each interval. At the end, we get the formula for the entire interval:

$$\begin{aligned} I(a,b)\approx \quad & \frac{1}{3}h\left[f(a)+4f(a+h)+f(a+2h)\right] \\ & +\frac{1}{3}h\left[f(a+2h)+4f(a+3h)+f(a+4h)\right]+\ldots \\ & +\frac{1}{3}h\left[f(a+(N-2)h)+4f(a+(N-1)h)+f(b)\right] \end{aligned}$$

We can collect the various term to simplify the algorithm somewhat:

$$\begin{aligned} I(a,b) &\approx \frac{1}{3}h\left[f(a)+4f(a+h)+2f(a+2h)+4f(a+3h)+\ldots+f(b)\right] \\ &= \frac{1}{3}h\left[f(a)+f(b)+4\sum_{k\text{ odd}}f(a+kh)+2\sum_{k\text{ even}}f(a+kh)\right] \end{aligned}$$

For the same example as before, with $N=10$ we get 4.00427 as a result. This is 0.01%! Significantly better. Simpson's rule is often the preferred method for evaluating integrals.