Calculus - Workshop 1 (Numerical Methods)

In this workshop we consider topics such as finding roots of a function, integration and differentiation. These topics are often introduced either in high school or in your calculus course. Our main focus here will therefore not be to study such topics analytically but instead how we can use a computer to solve problems either exactly or approximately.

Note, that these notes cover more material than we are going to cover in the lectures.

Finding Roots of a Function

The first thing we consider is a way to find roots of a function. We recall that x_0 is called a *root* of f if

$$f(x_0) = 0.$$

Thus to find a root of f, we want to solve the equation $f(x) = 0$, or at least find something that is very close to a solution (an approximate solution). To do so, we study to algorithms called *the bisection method* and *Newton's method* (also sometimes called Newton-Raphson's method).

The Bisection Method: The bisection method is based on the following result, which states that if we have a function which is continuous on some interval and the function value at the two endpoints of the interval, lies on opposite sides of the x-axis, then we can find a root (see Figur 1).

Theorem 0.1 (Intermediate Value)**.** *If f is a continuous function on the interval $[a, b]$, which satisfies that $f(a)$ and $f(b)$ have opposite signs, then f has at least one root.*

The idea of the bisection method is first to find an interval $[a, b]$ and then calculate $f(a)$ and $f(b)$. If $f(a) \cdot f(b) < 0$, then we know that they have opposite sign and therefore by Theorem 0.1 the interval must contain a root. If instead $f(a) \cdot f(b) > 0$ then we can't guarantee a root and therefore we stop. The next step is to cut the interval into two parts by taking the point $c = \frac{b+a}{2}$ and calculating $f(c)$. We then consider the product $f(a) \cdot f(c)$. If this product is negative we then consider the interval $[a, c]$ if it is positive we instead consider $[c, b]$. This can be done since the root must lie in one of the intervals and if we are not guaranteed that it lies in the first then by Theorem 0.1 it must lie in the second (see Figur 2).

We then continue to cut our interval in smaller and smaller pieces until the interval length is so small, that we say that one of the endpoints are sufficiently close to the root.

The bisection method is a relatively slow algorithm to find a root, in the sense that it often takes many iterations before we get sufficiently close to the root. But if we apply the bisection method, we are guaranteed that at some point it will get close enough to the root.

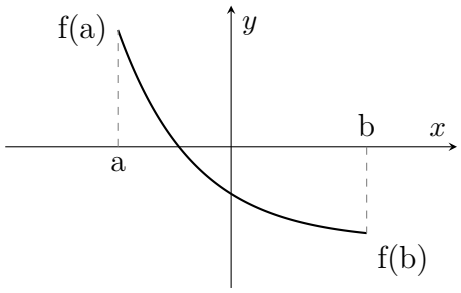


Figure 1: A function which satisfies the assumptions in Theorem 0.1.

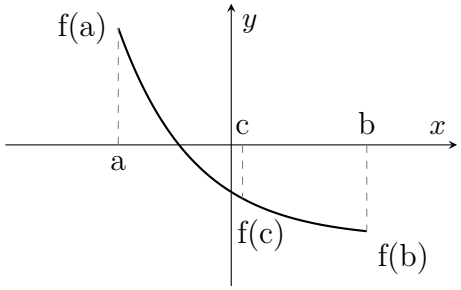


Figure 2: After the first step of the Bisection method.

Newton's Method: The next method we consider is Newton's method, which is based on Taylor's formula. We begin by recalling Taylors formula:

Theorem 0.2 (Taylor's Formula)**.** *If f is a function which is $(n+1)$-times differentiable on an interval containing the points x and x_0, then*

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + \frac{f^{(n+1)}(z)}{(n+1)!}(x-x_0)^{n+1}$$

for some z between x and x_0.

If we make a 1. order approximation of a differentiable function f, around a point x_0, we get the tangent line of f at the point x_0:

$$f(x) \approx f(x_0) + f'(x_0)(x-x_0). \tag{1}$$

To find a root of f we want to find an x such that the left-hand side of (1) is zero, but we can only work with the right hand side. Therefore we let the right hand side be equal to zero and obtain

$$f(x_0) + f'(x_0)(x-x_0) = 0 \Leftrightarrow x = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

The point x we get from this is the root of the tangent line of f at the point x_0. we denote this root by x_1 (see Figur 3). The value x_1 is not necessarily very close to the root of f, but by repeating the steps with x_0 replaced by x_1, we get a new approximate root, which we can call x_2 (see Figur 4). If our starting point x_0 is chosen sufficiently close to the actual root, then after k iterations in Newton's method (where k is some unknown integer) we get sufficiently close to the root. In Newton's method we keep updating our starting point every time we make an iteration, this makes Newton's method and iterative method with the general formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{2}$$

where x_n is the previous point and x_{n+1} is the one we are trying to find.

Note that, Newton's method are not always guaranteed to converge to the actual root, it will depend upon our starting point. But if Newton's method converge, then it will often give a good approximation of the root much faster than the bisection method.

We end this section on finding roots by giving conditions under which we can guarantee that Newton's method converges and an example where Newton's method does not converge.

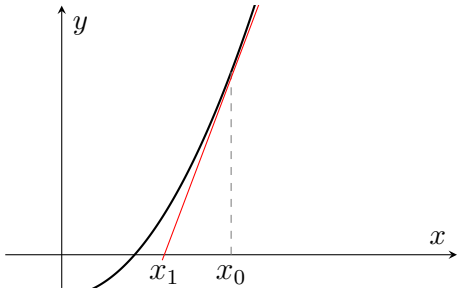


Figure 3: One iteration of Newton's method.

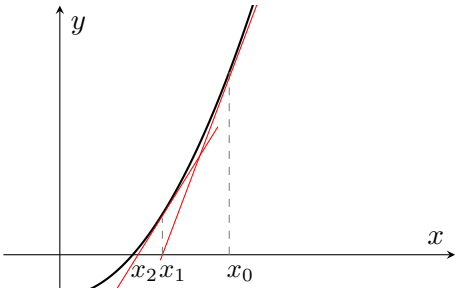


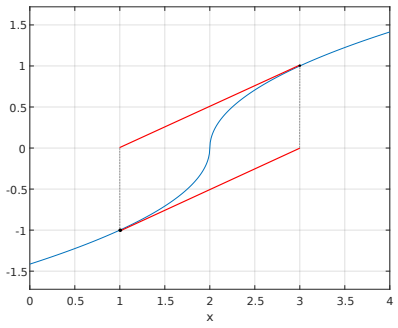
Figure 4: Two iterations of Newton's method.

Theorem 0.3. *Let $g(x) = x - \frac{f(x)}{f'(x)}$ and s be a point satisfying $s = g(s)$, i.e. s is a root of f. If g' is continuous on some interval I containing s and $|g'(s)| \leq \alpha < 1$, then Newton's method converges to a root for any starting point $x_0 \in I$.*

If we take the function

$$f(x) = \operatorname{sign}(x-2)\sqrt{|x-2|}, \qquad \text{where } \operatorname{sign}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0, \end{cases}$$

and apply Newton's method with starting point $x_0 = 1$, then it will cycle through the same two points and never get closer to the root:



Exercises:

1. Consider the function $f(x) = e^{-x} - 2$.
   (a) Show that f has a root using Theorem 0.1.
   (b) Find the root analytically by solving $f(x) = 0$.
2. Next we try to find the root of f using the bisection method. Do not implement any of the following steps before you understand why and what they do!
   (a) Choose a suitable interval $[a, b]$ in which the root is contained.
   (b) Set up a function which takes a, b, f, n as input and has the root as output
   (c) Make sure that $f(a)$ and $f(b)$ has opposite signs, else stop.
   (d) Write a for loop which runs n times.

(e) Choose c.

(f) Make an if and an else loop which switches either a or b with c.

(g) Return the root.

(h) Run the your bisection method on the function $f(x) = e^{-x} - 2$.

(i) What is a suitable n to get a good estimate of the root?

3. Try to implement the bisection method with a while loop instead of a for loop, where you keep running the bisection method until you are within some tolerance of the root (take for instance the tolerance 10^{-4}). Use it again on f.

4. Insert a counter in your bisection method with a while loop, to see how many times it run, before it gets beneath the tolerance.

5. Next we consider Newton's method. Do not implement any of the following steps before you understand why and what they do!

   (a) Choose a suitable starting value x.

   (b) Calculate f' analytically.

   (c) Set up a function which takes x, f, f', n as input and has the root as output.

   (d) define a variable $xprev$ and give it some value.

   (e) Write a for loop which runs n times.

   (f) Let $xprev$ be equal to x and update x according to (2).

   (g) Return the root.

   (h) Run Newton's method on the function $f(x) = e^{-x} - 2$.

   (i) What is a suitable n to get a good estimate of the root?

6. Try to implement Newton's method with a while loop instead of a for loop, where you keep running Newton's method until you are within some tolerance of the root (take for instance the tolerance 10^{-4}). Use it again on f.

7. Insert a counter in your Newton's method with a while loop, to see how many times it run, before it gets beneath the tolerance.

8. Is there a difference in how many times the bisection method and Newton's method runs, to get under the tolerance?

9. Try your Bisection method and your Newton's method on other functions of your choice (make sure the functions actually have a root).

Numerical Integration

The second topic we consider, is how to numerically calculate a definite integral, which is given as

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i^*)\Delta x, \tag{3}$$

where $a, b \in \mathbb{R}$ and f are given, n is the number of pieces we split $[a, b]$ into, x_i^* is a point in the subinterval $[x_{i-1}, x_i]$ and $\Delta x = \frac{b-a}{n}$ is the length of each subinterval

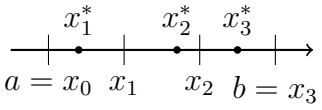


Figure 5: A partition of the interval $[a, b]$ with $n = 3$.

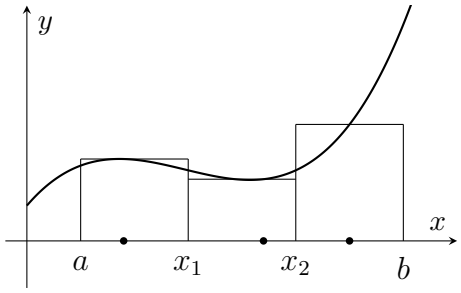


Figure 6: The approximate definite integral with $n = 3$ and x_i^* choosen as in Figur 5.

(see Figure 5 and 6). Note that (3) says that if we let the number of subintervals n go to infinity then the sum would become exactly the definite integral of f. It is not possible to make an infinite amount of intervals numerically, but choosing a n large makes the approximation better, but it does also increase the amount of time it takes for the computer to run the calculation.

We recall the Fundamental Theorem of Calculus, which relates the definite integral of a function to its antiderivative.

Theorem 0.4 (Fundamental Theorem of Calculus)**.** *If f is a continuous function with some antiderivative F, both defined on the interval $[a, b]$, then*

$$\int_a^b f(x)dx = F(b) - F(a).$$

Thus if we can find the antiderivative of a function, then we can find the definite integral. There is though functions, for which we can not easily find an antiderivative. For instance , consider the integral

$$\int_a^b e^{-x^2},$$

which we have no easy method of evaluating analytically. It is especially in such cases that numerical integration plays a central role.

Instead of using the Fundamental Theorem of Calculus to approximate the definite integral, we try approximate the sum in (3). To do so, we introduce two methods, namely the *midpoint rule* and *the trapezoid rule*.

The Midpoint Rule: If we are interested in calculating the definite integral of a function f over the interval $[a, b]$, then we partition the interval into n pieces $[x_{i-1}, x_i]$ where each subinterval has length $\Delta x = \frac{b-a}{n}$. The idea in the midpoint rule is then, that for each interval $[x_{i-1}, x_i]$ we take x_i^* to be the point exactly in the middle (see Figure 7) and then let the sum of the squares given by $\Delta x \cdot f(x_i^*)$ be our approximation of the integral (see Figure 8).

Thus if we approximate the definite integral by the midpoint rule, we get

$$\int_a^b f(x)dx \approx \Delta x[f(x_1^*) + f(x_2^*) + \cdots + f(x_n^*)].$$

The Trapezoid Rule: We again partition the interval $[a, b]$ into n subintervals $[x_{i-1}, x_i]$ of equal size $\Delta x = \frac{b-a}{n}$. The idea in the trapezoid rule is to approximate the

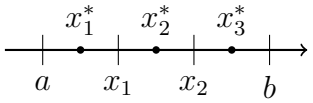


Figure 7: A partition of the interval $[a, b]$ using the midpoint rule.

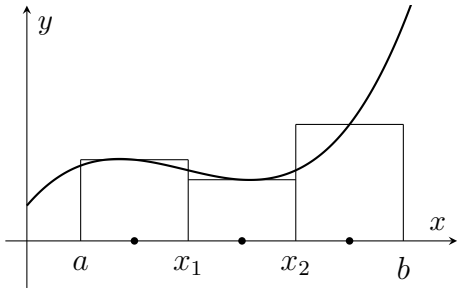


Figure 8: The approximate definite integral with $n = 3$ using the midpoint rule.

area in each subinterval by a trapezoid. To do so, note that in the interval $[x_{i-1}, x_i]$ the area of the trapezoid is given by (see Figure 9)

$$\Delta x\Big(\frac{f(x_{i-1}) + f(x_i)}{2}\Big).$$

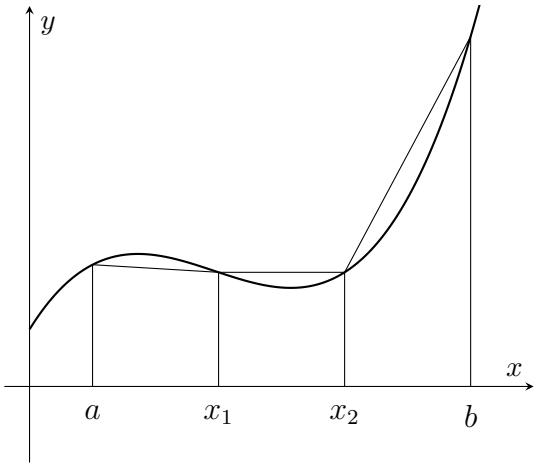


Figure 9: The approximate definite integral with $n = 3$ using the trapezoid rule.

If we approximate the definite integral by the trapezoid rule, we thus get

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}[f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]. \tag{4}$$

Exercises

1. Consider the function $g(x) = 3x^2$ on the interval $[-2, 2]$ and calculate the definite integral analytically.

2. We now use the midpoint rule to find the definite integral of g over the interval $[-2, 2]$. Do not implement any of the following steps before you understand why and what they do!

   (a) Set up a function which takes a, b, g, n as input and gives the area as output.

   (b) Define *Deltax* to be the size of each subinterval, when we partition the interval $[a, b]$ into n subintervals of equal size.

(c) Let c be half the size of each subinterval.

(d) Let N run from 1 to n.

(e) Let $xstar$ be the midpoint of each of the subintervals.

(f) Set up a for loop which runs n times.

(g) Let $fxstar$ be a vector with each entry the input function f evaluated on the i'th entry of $xstar$ (you could use $feval$ to do this, write $help\ feval$ in MATLAB to see how it works)

(h) Let the area be the sum of $fxstar$ multiplied with $Deltax$.

(i) Return the area.

(j) Apply your midpoint rule function to the function g.

(k) What is a suitable n to get a good estimate of the true definite integral?

3. Try to implement the trapez rule to find the definite integral of g over the interval $[-2, 2]$.

4. What is a suitable n to get a good estimate of the true definite integral?

5. Show that the right hand side of (4) can be written as

$$\sum_{i=1}^{n} \frac{f(x_{i-1}) + f(x_i)}{2} \Delta x.$$

6. Consider the integral $\int_{-1}^{1} 2\sqrt{1-x^2}dx$ and evaluate it analytically. (Use integration by substitution with $x = \cos(t)$).

7. Apply integration by the midpoint rule and by the trapez rule to evaluate $\int_{-1}^{1} 2\sqrt{1-x^2}dx$ numerically.

8. Give a geometric explanation for the above result (what figure does $2\sqrt{1-x^2}$ represent and how does that relate to the result you get)?