Università Commerciale
Luigi Bocconi

Bachelor of Science in
Mathematical and Computing Sciences for Artificial Intelligence

An introductory dive into numerical methods for computing integrals and solving differential equations

Thesis by: **BARBIERE Mattia Nino**
Advisor: Prof. LAVENANT Hugo

Academic Year 2023-2024

I dedicate my work to my parents and grandparents whose sacrifices and support have granted me the opportunity to pursue this bachelor's degree. A special thanks to professor Hugo Lavenant for this guidance in writing this thesis.

Abstract

Integrals and differential equations are a vital tool used to model our world with applications in every scientific discipline. Ever more complicated models and the lack of analytical solutions gave rise to *numerical analysis*. This branch of mathematics provides numerical methods to approximate the correct value without needing an explicit solution to the problem. As an added benefit these numerical methods are easily translated into computer code, allowing us to apply modern computational power to compute and simulate our mathematical models.

Contents

1 Introduction **3**

2 Polynomial interpolation **3**

2.1 Error analysis of Lagrange polynomial interpolation 5

2.2 Convergence of Lagrangian interpolation 6

2.2.1 Runge's phenomenon 7

3 Numerical methods for solving integrals **8**

3.1 Trapezoidal method for integration 9

3.1.1 Trapezoid method with uniform grid 11

3.1.2 Trapezoid method with non-uniform grid 12

3.1.3 Stability of the trapezoid method 12

3.1.4 Trapezoid integration of differentiable vs non-differentiable functions . 13

3.2 Simpson's rule for integration 16

3.2.1 Specific case of Simpson's rule 16

3.2.2 Simpson's rule 17

3.2.3 Composite Simpson's rule 17

3.2.4 The power of Simpson's rule 18

3.2.5 Error analysis of Simpson's rule 19

3.2.6 Simpson's rule example 22

3.2.7 Simpson's rule on differentiable vs non-differentiable functions 23

4 Numerical methods for solving ordinary differential equations **23**

4.1 Ordinary differential equations 23

4.1.1 Initial value problems 23

4.2 Euler methods 26

4.3 Trapezoidal method 26

4.4 Adams-Bashforth method 28

4.5 Runge-Kutta of fourth order (RK4) 29

5 Stability analysis of linear multistep methods **31**
5.1 Global and local truncation error 32
5.2 Characteristic polynomials 34
5.3 Zero-stability of a linear multistep method 34
5.4 Absolute stability for linear multistep methods 37
5.5 Particular case: Euler methods 38
5.5.1 Discussion of the Euler methods 39
5.6 Particular case: Trapezoidal method for IVPs 40
5.6.1 Discussion of the trapezoidal method 40
5.7 Particular case: Adams-Bashforth 42
5.7.1 Discussion of the Adams-Bathforth method 43

6 Stability analysis of Runge-Kutta of fourth order **43**
6.1 Discussion of Runge-Kutta of fourth order 45

7 Concluding remarks **45**

A Appendix: Supplementary Theorems **46**

Bibliography **48**

1 Introduction

The need for numerical methods for computing integrals and solving differential equations has been around for many years. Over time better methods and stronger theorems have been developed and nowadays, numerical analysis has become a vast and complex branch of mathematics.
With the advent of modern computers the most fundamental application of numerical analysis has been in computer science. For this reason, this paper is pared up with a Jupyter Notebook available at [1] where I implement all the numerical methods covered, together with graphical illustrations, some of which are already presented below.
Before starting with the methods I will devote some time to polynomial interpolation which is at the heart of many derivations in numerical analysis.

2 Polynomial interpolation

Throughout numerical analysis we encounter problems that have to deal with very intricate and complicated functions which often cause trouble when we try applying our mathematical tools on them. To try and overcome these issues, we must find clever ways to approximate the complicated functions with simpler and more mathematically tractable ones. Polynomials seem to check all the boxes. They are very easy to manipulate and are well-behaved when we apply operations on them. Furthermore, they are extremely versatile as you can essentially approximate any function by choosing the correct degree and playing around with the coefficients. Actually, the Weierstrass Approximation Theorem (see Theorem A.1) states that any continuous function on a closed interval $[a, b]$ can be approximated to an arbitrary level of precision by a polynomial. This theorem, however, does not tell us how to find such a polynomial. A classic why of finding a possible polynomial is using a Taylor expansion. While this is very powerful, in this chapter we will explore another why of finding a polynomial approximation of a function: *Lagrange polynomial interpolation*.
Given a function f and $n + 1$ points $\{x_0, \ldots, x_n\}$ (sampling points for example) define

$$y_i := f(x_i) \quad \text{for} \quad i \in \{0, \ldots, n\}.$$

We say a function $p(x)$ is an *interpolant* for $f(x)$ if $p(x)$ matches $f(x)$ at the $n+1$ points,

$$p(x_i) = y_i \quad \text{for} \quad i \in \{0, \dots, n\}. \tag{2.1}$$

We will focus on interpolations with polynomials throughout this paper, but this idea can be generalized to other classes of functions. On a side note, I often use interpolation and approximation interchangeably, however they are not the same thing. Interpolation is a method that may or may not be a useful approximation. All we say is that it must match the function at a finite amount of points. A priori, there is no reason to believe that this may work as a good approximation. Luckily, in numerical analysis, with a small modification presented in section 2.2.1, interpolation works very well! Its main usefulness is when we have a finite number of data points and a finite amount of values that the function takes. In this case, methods like Taylor series will fail as we have no knowledge about derivatives.
Our objective is to construct a polynomial that satisfies (2.1). The construction of $p(x)$ with the Lagrange method comes down to understanding the Lagrange basis. Define

$$l_i(x) := \prod_{j:j\neq i} \frac{x - x_j}{x_i - x_j}.$$

By simple inspection, we notice that for all $k \neq i$,

$$l_i(x_k) = \prod_{j:j\neq i} \frac{x_k - x_j}{x_i - x_j} = 0 \quad \text{and} \quad l_i(x_i) = \prod_{j:j\neq i} \frac{x_i - x_j}{x_i - x_j} = \prod_{j:j\neq i} 1 = 1.$$

This is basically the Kronecker delta constructed with polynomials. Using these n-th degree polynomials we can easily construct our interpolant polynomial as

$$p(x) = \sum_{i=0}^{n} y_i \, l_i(x). \tag{2.2}$$

This polynomial turns out to be unique among all the polynomials of degree n.

Proposition 2.1. *Given a function $f : \mathbb{R} \longrightarrow \mathbb{R}$ and $n+1$ points,*

$$x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n \quad \textit{with} \quad x_i \in [a, b] \quad \forall i \in \{0, \dots, n\}.$$

Then there is exactly one interpolating polynomial $p_n(x)$ of degree at most n that satisfies

(2.1) *with respect to* f.

Proof. **Existence**: The construction of the Lagrange basis as shown above is enough to prove existence.
Uniqueness: Assume, by contradiction, that there exist two polynomials p_n and q_n such that

$$deg(p_n), deg(q_n) \leq n \quad \text{and} \quad \exists x \in [a,b] : p_n(x) \neq q_n(x).$$

Define $r(x) = p_n(x) - q_n(x)$ which implies that $deg(r) \leq n$. Moreover, notice that

$$r(x_i) = p_n(x_i) - q_n(x_i) = 0 \quad \forall i \in \{0, \dots, n\}.$$

By the Fundamental Theorem of Algebra a non-constant polynomial of degree at most n, must have at most n roots, however $r(x)$ has $n+1$ roots and $deg(r) \leq n$ which implies that $r(x) = 0 \quad \forall x \in [a,b]$. This mean that $p_n(x) = q_n(x) \quad \forall x \in [a,b]$ which gives a contradiction. ■

2.1 Error analysis of Lagrange polynomial interpolation

Many of the numerical method I will cover start from a polynomial interpolation of the function in question. To provide a detailed error analysis of such methods, we must first understand the error that comes from polynomial interpolation. To this end we have the following theorem.

Theorem 2.2 (Lagrange error formula)**.** *Assume* $f \in C^{n+1}([a,b])$ *and* $n+1$ *points*

$$x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n \quad \textit{with} \quad x_i \in [a,b] \quad \forall i \in \{0, \dots, n\}.$$

Let $p_n(x)$ *be the interpolating polynomial of degree at most n. Then for all* $x \in [a,b]$

$$f(x) = p_n(x) + E_n(x)$$

with $E_n(x)$ *the interpolation error which takes value*

$$E_n(x) = \frac{f^{(n+1)}(\eta_x)}{(n+1)!} \prod_{j=0}^{n} (x - x_j) \tag{2.3}$$

for some $\eta_x \in [a, b]$.

Proof. A proof of this theorem can be found in [4] ■

Let's take a better look at (2.3). The first term we see is the $(n+1)$-th derivative. As we are working with a compact interval, we can easily bound this by taking the $\max_{x\in[a,b]} |f^{(n+1)}(x)|$. We have little to no control of this value as it entirely depends on the function in question.
The second term is $\frac{1}{(n+1)!}$ which, lucky for us, goes to 0 very quickly as $n \to +\infty$.
The last term is what we call $\omega(x)$:

$$\omega(x) := \prod_{j=0}^{n}(x - x_j).$$

We can see that $\omega(x)$ will be small if x is close to one of the points and will be large otherwise.
The error analysis essentially boils down to which factor will have a bigger effect on the upper bound as we increase the number of points at our disposal. The upper bound on an interval I is of the form

$$|E_n(x)| \leq \frac{\max_{x\in I} |f^{(n+1)}(x)|}{(n+1)!} \max_{x\in I} |\omega(x)|.$$

2.2 Convergence of Lagrangian interpolation

The idea behind the discussion above is that we can sample n values of our arbitrarily complicated function and construct a polynomial approximation that coincides with the function at those points. The next question that comes to mind is whether the interpolation error goes to 0 as we increase the number of samples. The answer to this depends on our choice of the interpolation points. All the methods I will present will assume that the data points are evenly spaced with size h. In doing so, however, we risk running into a problem of the interpolation polynomial diverging from the function. The next example illustrates an instance in which this happens.

2.2.1 Runge's phenomenon

Consider the interval $[-5, 5]$ and the following $C^\infty([-5, 5])$ function over that interval,

$$G(x) := \frac{1}{1 + x^2}. \tag{2.4}$$

If we try and interpolate this function using a uniform grid we see that towards the edges the polynomial starts to rapidly oscillate.

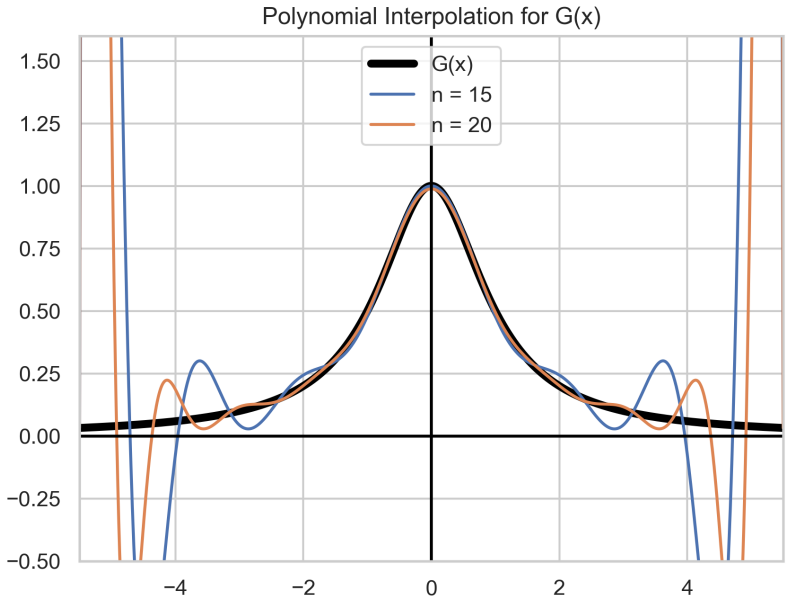


Figure 1: Divergence of interpolant polynomial with uniform grid when applied to the Runge function $G(x) = \frac{1}{1+x^2}$.

For larger values of n this behaviour only gets worse. The problem of this function is that the n-th derivatives blows up in the interval $[-5, 5]$. Some empirical evidence of this is shown in the Jupyter Notebook [1].

It might seem that we have little hope of working with the polynomial interpolation with evenly spaced points. Lucky for us there are a few workarounds that we can exploit. The main solution that most derivations will use is *spline interpolation*. What this does is to break up the interval into many subintervals. The advantages of this are twofold. The first and foremost is that we avoid the bad behaviour seen above. Secondly, by reducing the

size of the subintervals, if our function is smooth enough, we can approximate very well the function with a low degree interpolating polynomial, which reduces complexity. Below we can see that spline interpolation works very well even with the problematic Runge function $G(x)$.

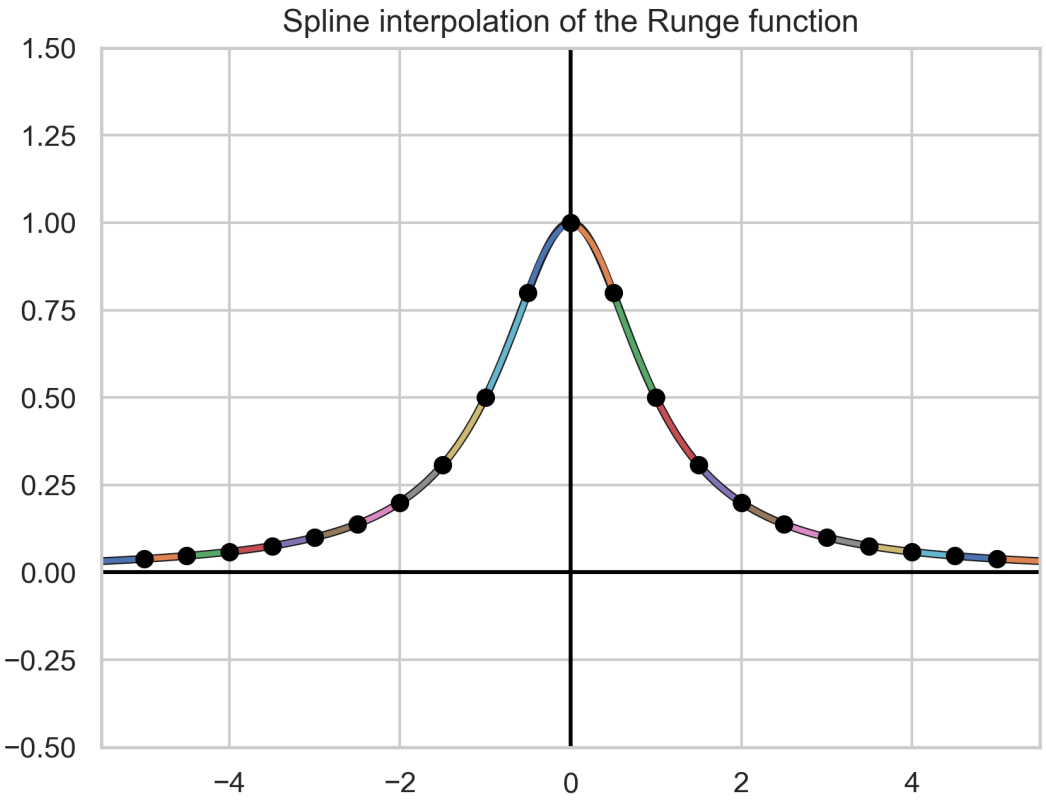


Figure 2: Spline interpolation applied to the Runge function $G(x) = \frac{1}{1+x^2}$.

With this solution in mind, numerical methods boil down to working over each subinterval individually and then combining all the answers at the end.

3 Numerical methods for solving integrals

We start with exploring some methods for numerically solving integrals. This is motivated by the fact that often differential equations and integrals are two sides of the same coin. Take

a differential equation as defined in (4.1). The assumptions in the Picard–Lindelöf Theorem A.3 needed to guarantee the existence and uniqueness of a solution are also enough for us to conclude that the function f is Riemann integrable over the first variable. Taking the integral of the ODE yields

$$\int_{t_0}^{t} f(s, y(s))ds = \int_{t_0}^{t} y'(s)ds = y(t) - y(t_0) = y(t) - y_0$$

giving us the equation

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds. \tag{3.1}$$

By building methods for computing integrals, we are often able to make good progress in numerically solving differential equations. I will discuss the Trapezoidal Rule and Simpson's rule for integration, and in later sections we will see how these can be transformed into methods for solving ODEs.

3.1 Trapezoidal method for integration

Given any function $f : \mathbb{R} \longrightarrow \mathbb{R}$ our objective is to compute

$$I(f) = \int_{a}^{b} f(x)\, dx. \tag{3.2}$$

In light of what we know of polynomial interpolation, a first step would be to use the interpolation of f of degree 1. From (2.2) we have

$$p(x) = \frac{x-a}{b-a} f(b) + \frac{b-x}{b-a} f(a).$$

This polynomial provides a linear approximation of the function f in the interval $[a, b]$. By integrating the polynomial, we could approximate the integral we are after. Geometrically, by integrating $p(x)$ we are calculating the area of a trapezoid with vertices $(a, 0)$, $(a, f(a)), (b, f(b))$ and $(b, 0)$. Using the formula for the area of a trapezoid with height equal to $(b - a)$ we get

$$T_1(f) := \int_{a}^{b} p(x)\, dx = \frac{1}{2}(b-a)(f(a) + f(b)). \tag{3.3}$$

If is very evident that if $b-a$ is very large, this could turn out to be a very crude approximation. The error denoted as $E_T(f)$ in fact is given by the following theorem.

Theorem 3.1 (Trapezoid method error with a single interval)**.** *Let $f \in C^2([a,b])$ and let $T_1(f)$ be the trapezoid approximation to $I(f)$ as defined in* (3.3)*. Then there exists $\xi \in [a,b]$ such that*

$$E_T(f) := I(f) - T_1(f) = -\frac{1}{12} f''(\xi)(b-a)^3.$$

Proof.

$$\begin{aligned} E_T(f) = I(f) - T_1(f) &= \int_a^b f(x)\,dx - \int_a^b p(x)\,dx = \\ &= \int_a^b (f(x) - p(x))\,dx = \frac{1}{2}\int_a^b ((x-a)(x-b)f''(\eta_x))\,dx. \end{aligned}$$

Where the last step comes from Theorem 2.2.
As $(x-a)(x-b) \leq 0$ on $[a,b]$ we use Theorem A.2 and get that, for some $\xi \in [a,b]$:

$$\begin{aligned} E_T(f) &= \frac{1}{2}\int_a^b (x-a)(x-b)f''(\eta_x)\,dx = \\ &= \frac{1}{2} f''(\xi)\int_a^b (x-a)(x-b)\,dx = \\ &= \frac{-1}{12} f''(\xi)(b-a)^3 \end{aligned}$$

as requested. ■

As we expected, the error is highly dependent on $b-a$. This means that if we have a large interval we may be very far away from the true value of the integral. Taking inspiration from spline interpolation, the trick is to divide the interval into many smaller intervals and then summing up the area of all the smaller trapezoids.
Take n subintervals $[x_{i-1}, x_i]$ for $i \in \{1, \ldots, n\}$ with $x_0 = a$ and $x_n = b$. Our integral then becomes

$$I(f) = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx.$$

For each subinterval we follow the approximation of the previous part and we are left with

$$\sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \sum_{i=1}^{n} \frac{1}{2}(x_i - x_{i-1})(f(x_i) + f(x_{i-1})) =: T_n(f). \tag{3.4}$$

How we divide the interval $[a, b]$ will have an effect on the overall error of the approximation.

3.1.1 Trapezoid method with uniform grid

Under the assumption that the grid is uniform we can find an exact value of the error directly from the calculation of $E_T(f)$ in the previous section. This leads us to the following theorem.

Theorem 3.2 (Trapezoid method error with a uniform grid)**.** *Let $f \in C^2([a, b])$ and let $T_n(f)$ be the n-subinterval trapezoid approximation of $I(f)$, using a uniform grid with subinterval size $h = \frac{b-a}{n}$. Then there exists $\xi \in [a, b]$ such that*

$$E_T(f) := I(f) - T_n(f) = -\frac{b-a}{12} h^2 f''(\xi).$$

Proof. The proof consists of summing the result from Theorem 3.1 over all the subintervals. More details can be found in [3]. ■

From this theorem we can clearly see that the trapezoidal rule with uniform grid has an error of second order $O(h^2)$.
This gives us to the following important corollary.

Corollary 3.2.1. *Let $f \in C^2([a, b])$ and let $T_n(f)$ be the n-subinterval trapezoid approximation of $I(f)$, using a uniform grid with subinterval size $h = \frac{b-a}{n}$. By letting the number of subintervals go to infinity or equivalently sending the size h of the uniform grid to zero, we get that*

$$\lim_{n\to\infty} T_n(f) = I(f).$$

Proof. We clearly have that $\max_{x\in[a,b]} |f''(x)|$ is finite by the continuity of f'' over $[a, b]$. Taking the absolute value of Theorem 3.2 and taking the limit gives the desired result.

$$\lim_{n\to\infty} |E_T(f)| = \lim_{h\to 0} \frac{b-a}{12} h^2 |f''(\xi)| \leq \left(\lim_{h\to 0} \frac{b-a}{12} h^2 \right) \max_{x\in[a,b]} |f''(x)| = 0.$$

■

So we are able to get the error sufficiently small by choosing h small enough. In other words, to lower the error, all we have to do is increase the number of subintervals.

3.1.2 Trapezoid method with non-uniform grid

If we have an arbitrary grid we are able to find an upper bound for the error depending on the size of the largest subinterval we have.

Theorem 3.3 (Trapezoid method error with a non-uniform grid)**.** *Let $f \in C^2([a,b])$ and let $T_n(f)$ be the n-subinterval trapezoid approximation of $I(f)$, using a non-uniform grid defined by*

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

with $h_i = x_{i+1} - x_i$ and $h = \max_i h_i$. Then

$$|E_n(f)| := |I(f) - T_n(f)| \leq \frac{b-a}{12} h^2 \max_{x \in [a,b]} |f''(x)|.$$

Proof. Similar to before but working with the new definition of h. ■

It is possible to construct a corollary similar to Corollary 3.2.1 however in this case increasing the number of subintervals will not guarantee that the error converges to 0. To see why, take a subinterval $\left[a, \frac{a+b}{2}\right]$. Now we can divide $\left[\frac{a+b}{2}, b\right]$ infinitely many times, however this will not lower the upper bound of our error, as the upper bound depends on $h = \max_i h_i$. So, to make sure that our error goes to 0, we must impose that all grid sizes go to 0 which is equivalent to imposing that the maximum grid size goes to 0. Similarly to the uniform case we also see that in this case the error has second order $O(h^2)$ paying careful attention to the definition of h in this context.

3.1.3 Stability of the trapezoid method

Other than the convergence to the correct value, another important aspect of approximations is stability. That is, if we slightly perturb the input function how does the value of our integral approximation change? We would like for the approximation to change very little and moreover, if we send the size of the perturbation to zero, the approximation should converge to the original value. This gives us a sense of *stability* for integration methods.

Often these methods are stable, however we will see that when we do a similar analysis for numerical methods for ODEs this is not always the case.
Define $g(x) := f(x) + \epsilon(x)$ with $\epsilon(x)$ the slight perturbation of $f(x)$. What can we say about $|T_n(f) - T_n(g)|$ given that $|f(x) - g(x)| = |\epsilon(x)|$? Recall from equation (3.4) that

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

Taking the difference gives

$$\begin{aligned} T_n(f) - T_n(g) &= \sum_{i=1}^{n} \frac{1}{2}(x_i - x_{i-1})(f(x_i) + f(x_{i-1}) - g(x_i) - g(x_{i-1})) \\ &= \sum_{i=1}^{n} \frac{1}{2}(x_i - x_{i-1})(\epsilon(x_i) + \epsilon(x_{i-1})). \end{aligned}$$

Taking the absolute value and bounding it from above gives,

$$|T_n(f) - T_n(g)| \leq 2 \max_{x\in[a,b]} |\epsilon(x)| \sum_{i=1}^{n} \frac{1}{2}(x_i - x_{i-1}) = (b-a) \max_{x\in[a,b]} |\epsilon(x)|.$$

Where the last step comes from the fact that we have a telescoping sum and that $x_0 = a$ and $x_n = b$.
This tells us that if the maximum perturbation is sufficiently small also $|T_n(f) - T_n(g)|$ will be small. Actually if we let $\max_{x\in[a,b]} |\epsilon(x)| \to 0$ we get that $|T_n(f) - T_n(g)| \to 0$ proving that the trapezoid method is numerically stable.

3.1.4 Trapezoid integration of differentiable vs non-differentiable functions

I will now present a comparison of the trapezoid method applied to two similar function one of which will be differentiable everywhere and the other not. By plotting the error, we can compare the convergence of the trapezoid method applied to these two functions. When I say convergence here I mean that the error between the approximated value and the correct value goes to 0.

Example 3.1. *Consider the following two functions*

$$g(x) = \frac{\pi}{2}\cos\left(e^x + \pi\right) + \frac{\pi}{2} \quad \textit{and} \quad f(x) = \arccos\left(\cos\left(e^x\right)\right) \quad \textit{for} \quad x \in [ln(2\pi), ln(20\pi)].$$

By plotting them over the interval we see that they are very similar. The main difference is that f is not differentiable everywhere.

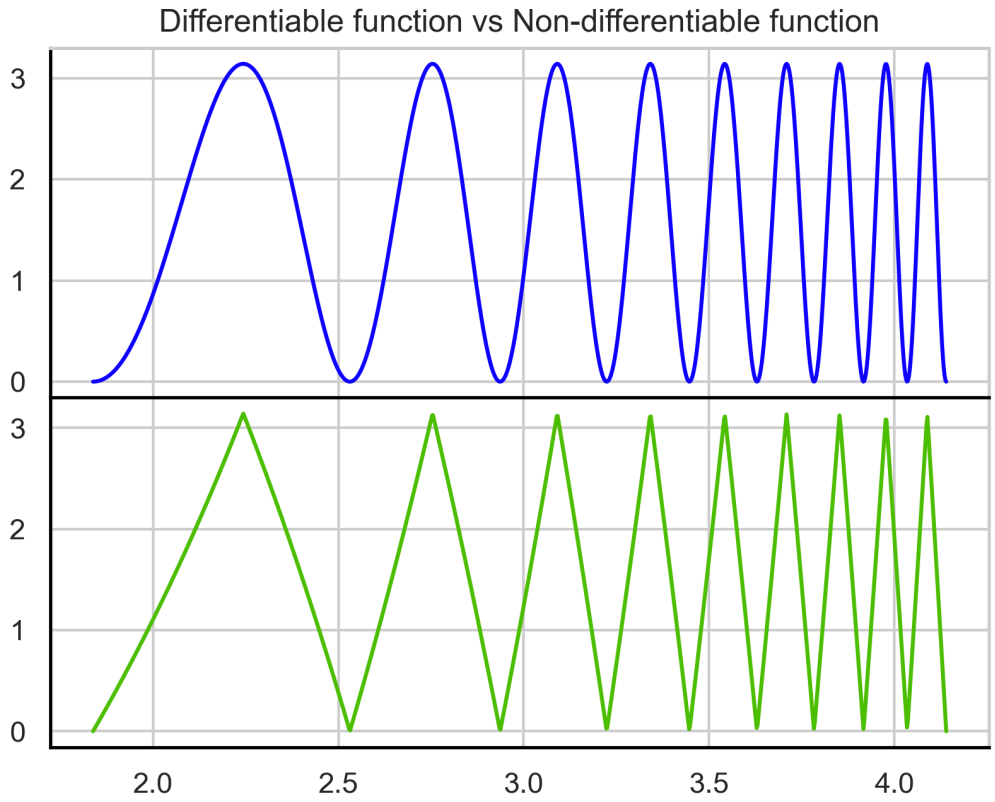
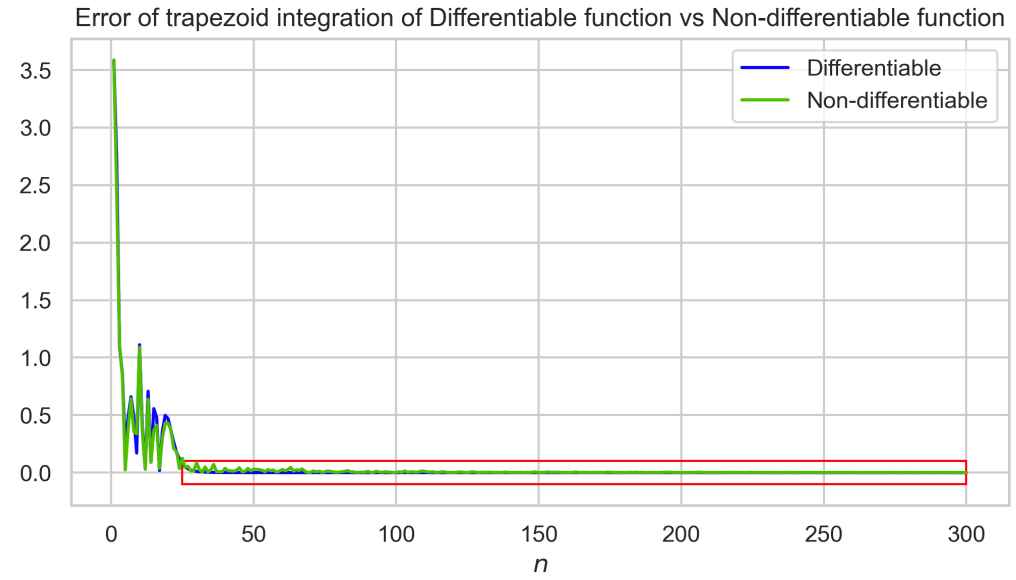


Figure 3: Plotting the functions $g(x) = \frac{\pi}{2}\cos\left(e^x + \pi\right) + \frac{\pi}{2}$ (blue) and $f(x) = \arccos\left(\cos\left(e^x\right)\right)$ (green) over the interval $[\ln(2\pi), \ln(20\pi)]$.

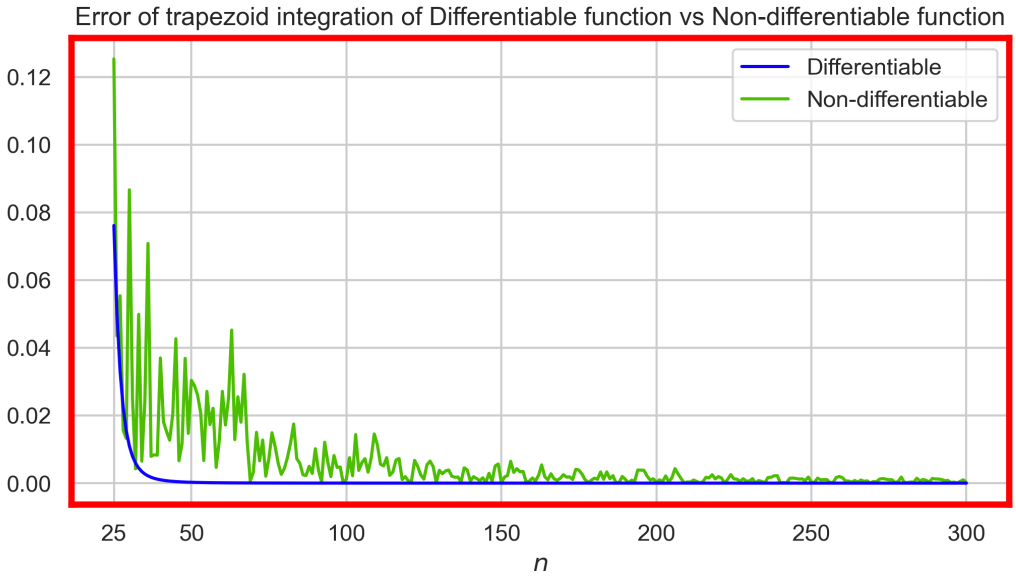
By applying the trapezoid method to both functions and plotting the error as a function of n, we get the following graphs.

We notice that the trapezoid method struggles with both functions at the beginning. This is due to the oscillatory nature of the functions which means the approximation cannot follow the functions well enough for small values of n. Once we get to $n \approx 25$ we see that for the

Figure 4: Error given by the trapezoid method (3.4) when integrating a differentiable function $g(x) = \frac{\pi}{2}\cos\left(e^x + \pi\right) + \frac{\pi}{2}$ vs a non-differentiable function $f(x) = \arccos\left(\cos\left(e^x\right)\right)$ for various values of n.



(a) Error values plotted with n ranging from 1 to 300.



(b) Zoomed in portion of the graph above with n ranging from 25 to 300.

differentiable function the error dramatically goes to 0, while for the non-differentiable function the method takes longer to reduce the error. Eventually, the error does decrease even for the non-differentiable function, but even at high values of n we see some fluctuations. This example illustrates the fact that numerical integration methods have an easier time approximating differentiable functions compared to non-differentiable. This is why many theorems that talk about approximation error assume some differentiability of the function.

3.2 Simpson's rule for integration

In wake of the derivation of the trapezoid method, we can take the polynomial interpolation one step further. That is, for each (sub)interval $[a, b]$ we interpolate the function f at $a, b, \frac{a+b}{2}$ with a second order polynomial. Recall that in the trapezoid method we only interpolate the end points of the intervals with a first degree polynomial. By increasing the degree of the polynomial we get Simpson's rule for integration.
Definite $m := \frac{a+b}{2}$, our polynomial interpolation of f at $a, b,$ and m is

$$q_{a,b}(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)}. \tag{3.5}$$

To derive Simpson's method we could follow the steps as we did with the trapezoid method, however there is a more elegant derivation.

3.2.1 Specific case of Simpson's rule

Our aim is interpolating a function f with a second degree polynomial in order to approximate the integral of f. Let's solve for the case when $a = -1, m = 0$ and $b = 1$:

$$q_{-1,1}(x) = f(-1)\frac{x(x-1)}{2} - f(0)(x+1)(x-1) + f(1)\frac{x(x+1)}{2}.$$

The idea is that the integral of $q_{-1,1}(x)$ over $[-1, 1]$ should approximate the integral of f over the same interval,

$$\int_{-1}^{1} q_{-1,1}(x)\,dx \approx \int_{-1}^{1} f(x)\,dx.$$

Integrating $q_{-1,1}(x)$ on $[-1,1]$ gives

$$\int_{-1}^{1} q_{-1,1}(x)\,dx = f(-1)\int_{-1}^{1}\frac{x(x-1)}{2}\,dx - f(0)\int_{-1}^{1}(x^2-1)\,dx + f(1)\int_{-1}^{1}\frac{x(x+1)}{2}\,dx =$$

$$= \frac{1}{3}(f(-1)+4f(0)+f(1)) \approx \int_{-1}^{1} f(x)\,dx. \tag{3.6}$$

3.2.2 Simpson's rule

Now notice the following function

$$y(x) = \frac{2(x-a)}{b-a} - 1. \tag{3.7}$$

This function linearly maps $x \in [a,b]$ into $y(x) \in [-1,1]$.
We can now move to the general case. Using the inverse of (3.7) we can perform a substitution and we get

$$x = \frac{b-a}{2}(y+1)+a \implies dx = \frac{b-a}{2}\,dy$$

$$I(f) = \int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{b-a}{2}(y+1)+a)\,dy.$$

For clarity let's define $F(y) := f(\frac{b-a}{2}(y+1)+a)$, and using (3.6) with $F(y)$ we get

$$\frac{b-a}{2}\int_{-1}^{1} F(y)\,dy \approx \frac{b-a}{2}\int_{-1}^{1} q_{-1,1}(x)\,dx = \frac{b-a}{6}(F(-1)+4F(0)+F(1)).$$

Recalling that $F(-1)=f(a), F(0)=f(m), F(1)=f(b)$ and letting $h=\frac{b-a}{2}$ we have the Simpson's method for integration defined as

$$S_2(f) := \frac{h}{3}(f(a)+4f(m)+f(b)) \approx I(f). \tag{3.8}$$

3.2.3 Composite Simpson's rule

Recalling the motivation behind spline interpolation, we divide the interval $[a,b]$ in many subintervals and perform what we did in Section 3.2.2 on each subinterval. Since we need a middle point to perform our interpolation, each subinterval must contain three of our sample points.

Given $n+1$ uniformly distributed points with n even and $n \geq 2$,

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b \quad \text{and} \quad h = \frac{x_{2k} - x_{2k-2}}{2} = x_{2k} - x_{2k-1}$$

we apply Simpson's method on each subinterval of the form $[x_{2k-2}, x_{2k}]$ for $k \in \{1, \ldots, \frac{n}{2}\}$ using x_{2k-1} as the midpoint.
For a single subinterval we have

$$\int_{x_{2k-2}}^{x_{2k}} f(x)\,dx \approx \frac{h}{3}(f(x_{2k-2}) + 4f(x_{2k-1}) + f(x_{2k})).$$

By summing over all the subintervals we get the composite Simpson's rule as

$$\int_a^b f(x)\,dx = \sum_{k=1}^{n/2} \int_{x_{2k-2}}^{x_{2k}} f(x)\,dx \approx \frac{h}{3} \sum_{k=1}^{n/2} (f(x_{2k-2}) + 4f(x_{2k-1}) + f(x_{2k})) =: S_n(f). \quad (3.9)$$

3.2.4 The power of Simpson's rule

Recall from Section 3.2 that Simpson's rule starts from a second degree polynomial interpolation of the function f. Consider the special case when f is itself a polynomial of at most second degree. Then by interpolating f as we did in (3.5) we would get that $q_{a,b}$ is exactly equal to f. If this were not the case we could view f as a polynomial interpolation of itself and we would contradict Proposition 2.1. In this case, Simpson's rule exactly calculates the integral. Now this seems like a reasonable thing to happen, however what is special about this rule is that the same is true for third degree polynomials!

Lemma 3.4. *Let $f(x)$ be a polynomial with $deg(f) \leq 3$. Then Simpson's rule comes out to be an equality*

$$S_2(f) = \int_a^b f(x)\,dx.$$

Proof. I have already talked about the case when $deg(f) \leq 2$. Let's focus on $deg(f) = 3$.

$$f(x) = Cx^3 + g(x) \quad \text{with} \quad deg(g) \leq 2.$$

Notice that both I and S_2 are linear thus

$$I(f) - S_2(f) = C(I(x^3) - S_2(x^3)) + (I(g) - S_2(g)).$$

By our discussion above $I(g) - S_2(g) = 0$. Let's evaluate $S_2(x^3)$

$$\begin{aligned} S_2(x^3) &= \frac{b-a}{6}\left(a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3\right) = \\ &= \frac{b-a}{6}\left(a^3 + \frac{1}{2}a^3 + \frac{3}{2}a^2b + \frac{3}{2}ab^2 + \frac{1}{2}b^3 + b^3\right) = \\ &= \frac{b-a}{4}(a^3 + a^2b + ab^2 + b^3) = \\ &= \frac{b^4 - a^4}{4} = \\ &= I(x^3). \end{aligned}$$

Which gives us that $I(f) - S_2(f) = 0$ concluding the proof. ■

3.2.5 Error analysis of Simpson's rule

One might be tempted to follow a similar reasoning as we did with the trapezoid method; we have three points $a, m,$ and b, and starting from (2.3) we integrate and get the resulting error. The error term comes out to be

$$E_2(x) = \frac{f^{(3)}(\eta_x)}{3!}(x-a)(x-m)(x-b) \quad \text{for some} \quad \eta_x \in [a,b].$$

In this case the approach we took in the case of the trapezoid method breaks down as $(x-a)(x-m)(x-b)$ changes sign over the interval $[a,b]$ and thus Theorem A.2 cannot be applied. With this direct approach the best we can do is to provide an upper bound by splitting up the integral.

$$\begin{aligned} |E_2(x)| &= \left| \int_a^b \frac{f^{(3)}(\eta_x)}{6}(x-a)(x-m)(x-b)\,dx \right| \leq \\ &\leq \frac{C}{3}\int_a^m |(x-a)(x-m)(x-b)|\,dx \end{aligned} \tag{3.10}$$

where the last step comes from the symmetry of $|(x-a)(x-m)(x-b)|$ around m and $C := \max_{x\in[a,b]} |f^{(3)}(x)|$.

Even if we could somehow directly take $f^{(3)}(\eta_x)$ outside the integral, this would be of little

help as

$$\int_a^b \left((x-a)\left(x - \frac{a+b}{2}\right)(x-b) \right) dx = 0.$$

After having seen Lemma 3.4 this comes as no surprise, however this means that we have to come up with a better method for deriving the error of Simpson's rule.

Theorem 3.5 (Error of Simpson's rule)**.** *Suppose $f \in C^4([a,b])$, then there exists a point $\xi \in [a,b]$ such that*

$$I(f) - S_2(f) = -\frac{(b-a)^5}{2880} f^{(4)}(\xi).$$

In Simpson's rule we are only given 3 points which means that we can interpolate with a second degree polynomial. As we have already discussed directly computing the error from $E_2(x)$ does not work. We would like to go up by one degree, that is, calculating the error using $E_3(x)$. We only have 3 points at our disposal and ultimately to derive Simpson's rule we only need these. To overcome this we start with 4 points and by taking a limit we end up with only 3.

Proof. Let p_3 interpolate f at the points $x_0 = a, x_1 = m - \epsilon, x_2 = m + \epsilon$, and $x_3 = b$ with $\epsilon > 0$ a small deviation. By (2.3), for $x \in [a,b]$

$$f(x) - p_3(x) = E_3(x) = \frac{1}{4!}(x-x_0)(x-x_1)(x-x_2)(x-x_3) f^{(4)}(\eta_x)$$

for some $\eta_x \in [a,b]$. By Lemma 3.4 we have that

$$I(f) - S_2(f) = I(E_3) - S_2(E_3) + I(p_3) - S_2(p_3) = I(E_3) - S_2(E_3).$$

As p_3 is an interpolating polynomial of f we know that at $x = a$ and $x = b$ the error $E_3(x)$ will vanish and so we get

$$\begin{aligned} S_2(E_3) &= \frac{b-a}{6}(E_3(a) + 4E_3(m) + E_3(b)) = \\ &= \frac{b-a}{6}(4E_3(m)) = \\ &= \frac{4(b-a)}{6 \cdot 4!}(m-x_0)(m-x_1)(m-x_2)(m-x_3) f^{(4)}(\eta_m). \end{aligned}$$

Now by taking the limit as $\epsilon \to 0$ we get that $x_1 \to m$ and $x_2 \to m$. In turn this gives

$E_3(m) \to 0$ and so also $S_2(E_3)$ goes to 0. Lastly, $f \in C^4([a,b])$ implies that

$$\frac{1}{4!}(x-a)(x-x_1)(x-x_2)(x-b)f^{(4)}(\eta_x)$$

is continuous on $[a,b]$ and hence can be bounded by a constant. We can now invoke the Dominated Convergence Theorem A.4 allowing us to interchange limit and integral. Thus,

$$\lim_{\epsilon \to 0} I(E_3) = \frac{1}{4!}\int_a^b (x-a)(x-m)^2(x-b)f^{(4)}(\eta_x)\,dx.$$

So in the limit we are left with

$$I(f) - S_2(f) = \frac{1}{4!}\int_a^b (x-a)(x-m)^2(x-b)f^{(4)}(\eta_x)\,dx.$$

As $(x-a)(x-m)^2(x-b) \leq 0 \quad \forall x \in [a,b]$ we can use Theorem A.2 and for some $\xi \in [a,b]$ we get

$$I(f) - S_2(f) = \frac{f^{(4)}(\xi)}{4!}\int_a^b (x-a)(x-m)^2(x-b)\,dx.$$

Computing the integral gives

$$\int_a^b (x-a)(x-m)^2(x-b)\,dx = -\frac{(b-a)^5}{120}.$$

So

$$I(f) - S_2(f) = -\frac{(b-a)^5}{2880}f^{(4)}(\xi).$$

■

By applying this theorem to each subinterval one can prove the following theorem.

Theorem 3.6 (Error of composite Simpson's rule)**.** *Assume we have a uniform grid of n points with grid size $h = \frac{b-a}{n}$ and that $f \in C^4([a,b])$, then for some $\xi \in [a,b]$*

$$I(f) - S_n(f) = -\frac{(b-a)h^4}{180}f^{(4)}(\xi).$$

Here we clearly see that in fact Simpson's method has error order 4, $O(h^4)$. The remarkable thing about this rule is that we only increased the interpolation degree by one, compared to the trapezoid method, but the error increase by 2 orders!

3.2.6 Simpson's rule example

Does this theoretical order really hold? With the following simple example we will see that this does in fact occur. We would like to calculate the following integral:

$$\int_0^{\pi} e^x \cos(x)\, dx.$$

Which we know has value $-(e^{\pi}+1)/2 \approx -12.070346316$. By implementing (3.9) we get the following table.

n	h	S_n	Errors	Error ratio
2	1.570796	-11.592840	4.775068e-01	N.A
4	0.785398	-11.984944	8.540230e-02	5.591264
8	0.392699	-12.064209	6.137359e-03	13.915154
16	0.196350	-12.069951	3.949931e-04	15.537889
32	0.098175	-12.070321	2.486034e-05	15.888486
64	0.049087	-12.070345	1.556458e-06	15.972377
128	0.024544	-12.070346	9.732054e-08	15.99311
256	0.012272	-12.070346	6.083193e-09	15.998268
512	0.006136	-12.070346	3.802132e-10	15.999425
1024	0.003068	-12.070346	2.376233e-11	16.000673

In the table above we compute the integral with Simpson's rule doubling n every time. Consequently, this means that h gets halved each time. The third column is the approximate value given by Simpson's rule. We see that it gets remarkably close to the correct value. In fact, the error becomes very small for large n. Now, to compute the order we calculate the ratio using Theorem 3.6

$$\frac{I(f)-S_n(f)}{I(f)-S_{2n}(f)} = \frac{(\frac{b-a}{n})^4}{(\frac{b-a}{2n})^4} = 16.$$

By looking at the last column of the table we see that this theoretical convergence does hold.

It should be noted that we can easily perform a similar analysis as we did with the trapezoid method in Section 3.1.3, allowing us to conclude that also Simpson's rule for integration is numerically stable.

3.2.7 Simpson's rule on differentiable vs non-differentiable functions

Similarly to what we did with the trapezoid method, we can apply Simpson's rule to the functions suggested in Example 3.1. Plotting the errors for different values of n we get the plots in Figure 5.
By inspection we notice that, for the differentiable function, Simpson's rule needs larger n to stabilize and converge compared to the trapezoid method. Furthermore, Simpson's rule has a very hard time approximating the non-differentiable function. Other than the rapid fluctuations we also see that the overall decrease in the error is very slow and still fluctuates for large values of n.

4 Numerical methods for solving ordinary differential equations

4.1 Ordinary differential equations

To fix some notation, I will denote $x : [a, b] \longrightarrow \mathbb{R}$ the unknown function which we wish to solve for in the differential equation. From now on I will be assuming that x is a function of one variable, commonly denoted t, but many concepts can be extended to more general functions. An *ordinary differential equation* is an equation of the form

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = x'(t) = f(t, x(t)). \tag{4.1}$$

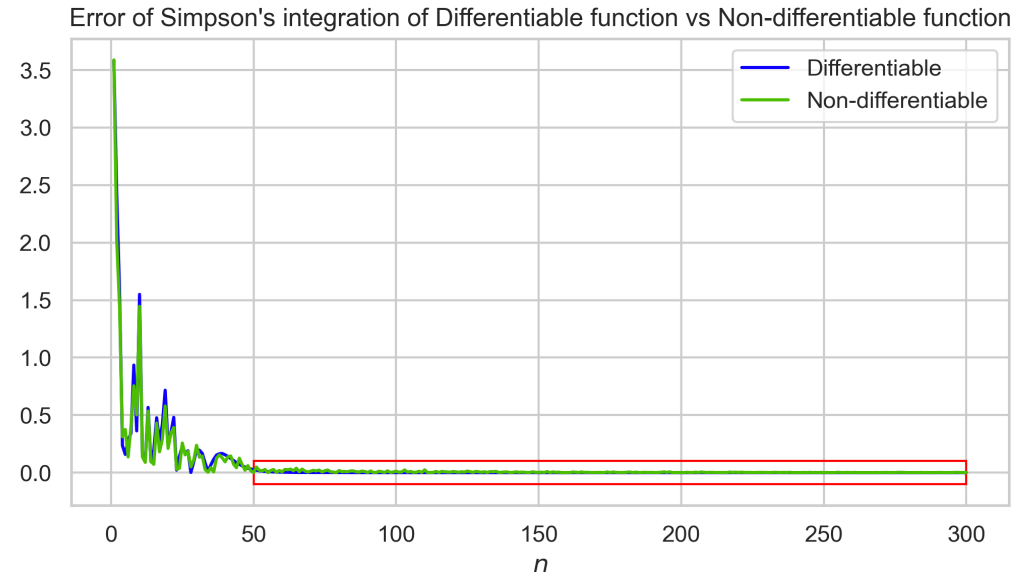
When solving for $x(t)$ we find a whole family of functions that satisfy (4.1). To have a unique solution we need to pair our ODE with an initial condition.

4.1.1 Initial value problems

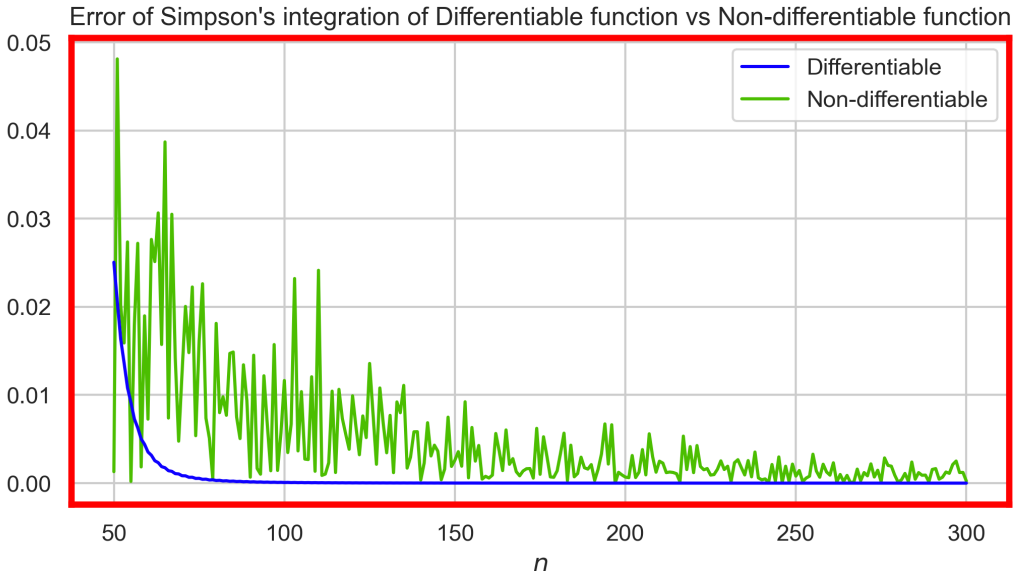
As the name suggests, an *initial value problem* combines an ODE with a known value x_0 that our function $x(t)$ takes at some initial time denoted t_0. Often in examples I will take $t_0 = 0$ for simplicity.

Definition 4.1 (Initial Value Problem)**.** *Assume that $\Omega \subseteq \mathbb{R}^2$ is an open set and let $f : \Omega \longrightarrow \mathbb{R}$ be a known function. Define a function $x : [a, b] \longrightarrow \mathbb{R}$ such that $(t, x(t)) \in \Omega$ for*

Figure 5: Error given by Simpson's rule method when integrating differentiable function $g(x) = \frac{\pi}{2}\cos\left(e^x + \pi\right) + \frac{\pi}{2}$ vs non-differentiable function $f(x) = \arccos\left(\cos\left(e^x\right)\right)$ for various values of n.



(a) Error values plotted with n ranging from 1 to 300.



(b) Zoomed in portion of the graph above with n ranging from 50 to 300.

all $t \in [a, b]$*. An Initial Value Problem is the following:*

$$\begin{cases} x'(t) = f(t, x(t)) \\ x(t_0) = y_0 \end{cases}$$

for some known $t_0 \in (a, b), x_0 \in \mathbb{R}$*, and* $(t_0, x_0) \in \Omega$*.*

The Picard–Lindelöf Theorem guarantees the existence and uniqueness of a solution x to an IVP under the assumptions that f is continuous in t and Lipschitz continuous in x. For more details see Theorem A.3 and [11]. I will be implicitly assuming that these assumptions hold for the rest of the paper so that all the calculations are well-defined.
An interesting example of initial value problem is the following. Its importance will be understood once the notion of absolute stability is presented in Section 5.4.

Example 4.1. *Consider the following IVP:*

$$\begin{cases} x'(t) = -2x \\ x(0) = 3 \end{cases}$$

Which has an exact solution $x(t) = 3e^{-2t}$*.*

Often in numerical analysis it is common practice to discretize time. From now on I will assume that we discretize time uniformly with step size h.

$$t_n := a + nh \quad \text{for} \quad n \in \left\{0, \ldots, \left\lfloor \frac{b-a}{h} \right\rfloor \right\}.$$

For clarity, I will set $x_n := x(t_n) = x(a + nh)$ and I will denote y_n as the approximation of x_n given by a numerical method.
We are finally ready to talk about numerical methods for ordinary differential equations. These become more complex compared to integration since an error in the beginning may lead to "snowball effect" and so a much bigger error further down the line. For this reason, the stability analysis becomes more important and complicated than before. We start with the Euler methods which are the easiest and simplest methods for solving IVPs.

4.2 Euler methods

The simplest methods for solving IVPs are the Euler methods. These are derived directly from approximating the derivative with the difference quotient.
Explicit Euler: The relation of the explicit Euler method is as follows

$$y_{n+1} = y_n + hf(t_n, y_n), \quad y_0 = x_0.$$

Implicit Euler: The implicit Euler method is similar to the previous with a slight variation

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad y_0 = x_0.$$

I will not present the derivation and error analysis of these methods as the computations are fairly straight forward.

4.3 Trapezoidal method

Recalling equation (3.1), by discretizing time we have that at time t_{n+1}

$$x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} f(s, x(s))\, ds.$$

Now applying $T_1(f)$ from equation (3.3) to the integral

$$y_{n+1} = y_n + \frac{1}{2}h(f(t_n, y_n) + f(t_{n+1}, y_{n+1})) \quad \text{and} \quad y_0 = x_0. \tag{4.2}$$

Now we have a formula for computing the next step of our method. When applying this method one usually computes the value for many steps to get an idea of the shape of the unknown function.
To begin our error analysis, we start with the notion of *local truncation error*. To put it simply the local truncation error of a numerical method is the error we accumulate by interpolating the function f. In section 5.1 I provide an alternative but equivalent definition of this quantity which is easier to generalize. Since our derivations start from the polynomial interpolation of the function, it is easier to work with the first definition. Lucky for us, we have already done all the calculations needed to calculate the local truncation error of this method in

Theorem 3.1 and we get that, for all $n > 0$, there exists some $\xi_n \in [a, b]$

$$E_{t_n}^{Trap} = -\frac{1}{12}h^3 f''(\xi_n, x(\xi_n)) = -\frac{1}{12}h^3 x'''(\xi_n).$$

This local truncation error is the error of a single step and it turns out to be of order $O(h^3)$. When a method has local truncation error of order $p+1$ (i.e. $O(h^{p+1})$) then we say that the method is of order p. Theorem 5.1 will provide justification of this definition. In this case the trapezoid method is of order 2. As we can see from (4.2) this is an implicit method, that is, to find the value of y_{n+1} we must solve an equation. Depending on f this is easier said than done, and the method we use to solve the equation might have an impact on the accuracy of our numerical solution. This will be discussed further in Section 5.6.1.

To illustrate the trapezoid method at work we can apply it to Example 4.1 and we get the plot in Figure 6.

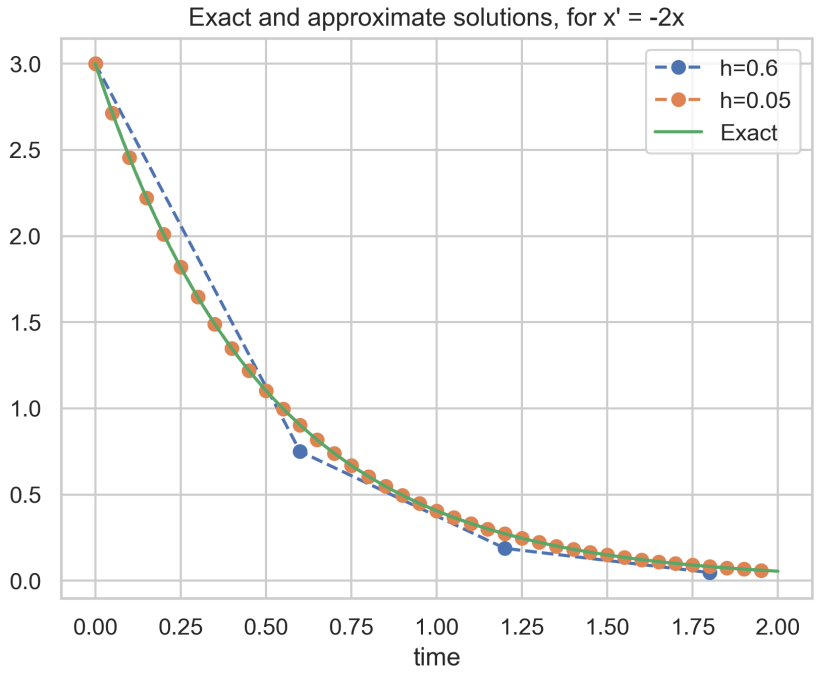


Figure 6: Trapezoidal method when solving $x'(t) = -2x$, $x(0) = 3$.

4.4 Adams-Bashforth method

It has become common practice in this paper to start deriving numerical methods starting from a polynomial interpolation. This method is no exception. The derivation of the Adams-Bashforth method is reminiscent of what we did for the trapezoidal method and in fact there is a derivation starting from the trapezoidal method. Nevertheless, I will again start from polynomial interpolation.

We first discretize time into many steps of size h. Then, for ease of notation set $g(t) = f(t, x(t))$ and take the interpolating polynomial of g at the points t_{n-1} and t_n

$$p(x) = \frac{x - t_{n-1}}{t_n - t_{n-1}} g(t_n) + \frac{t_n - x}{t_n - t_{n-1}} g(t_{n-1}).$$

Where this derivation differs compared to that of the trapezoid method is that we now integrate over $[t_n, t_{n+1}]$ as opposed to $[t_{n-1}, t_n]$. As $p(x)$ interpolates g we get the following approximation

$$\int_{t_n}^{t_{n+1}} g(s)\, ds \approx \int_{t_n}^{t_{n+1}} p(s)\, ds.$$

By direct computations we get that the integral of the interpolating polynomial is

$$AB(g) := \int_{t_n}^{t_{n+1}} p(s)\, ds = \frac{h}{2}(3g(t_n) - g(t_{n-1})).$$

To find the truncation error we again start from Theorem 2.2.

$$E_{t_n}^{AB} := \int_{t_n}^{t_{n+1}} g(s)\, ds - \int_{t_n}^{t_{n+1}} p(s)\, ds = \int_{t_n}^{t_{n+1}} \frac{g''(\eta_s)}{2}(s - t_{n-1})(s - t_n)\, ds.$$

Then using Theorem A.2 and computing the integral, for some $\xi_n \in [t_n, t_{n+1}]$ we have that

$$\int_{t_n}^{t_{n+1}} \frac{g''(\eta_s)}{2}(s - t_{n-1})(s - t_n)\, ds = \frac{g''(\xi_n)}{12}(t_{n+1} - t_n)^2(t_n + 2t_{n+1} - 3t_{n-1}) = \\ = \frac{5}{12} h^3 g''(\xi_n).$$

This indicates that the Adams-Bashforth method has truncation error of order 3, $O(h^3)$ and so is of second order. Starting from (3.1) and recalling that $g(t) = f(t, y)$ we get this plot.

$$y_{n+1} = y_n + \frac{1}{2} h(3f(t_n, y_n) - f(t_{n-1}, y_{n-1})) \quad \text{and} \quad y(t_0) = y_0. \tag{4.3}$$

To start the recursion we also need y_1. The simplest way to do this is to use Euler's method to give the first value. If we try and apply this method to Example 4.1 we get

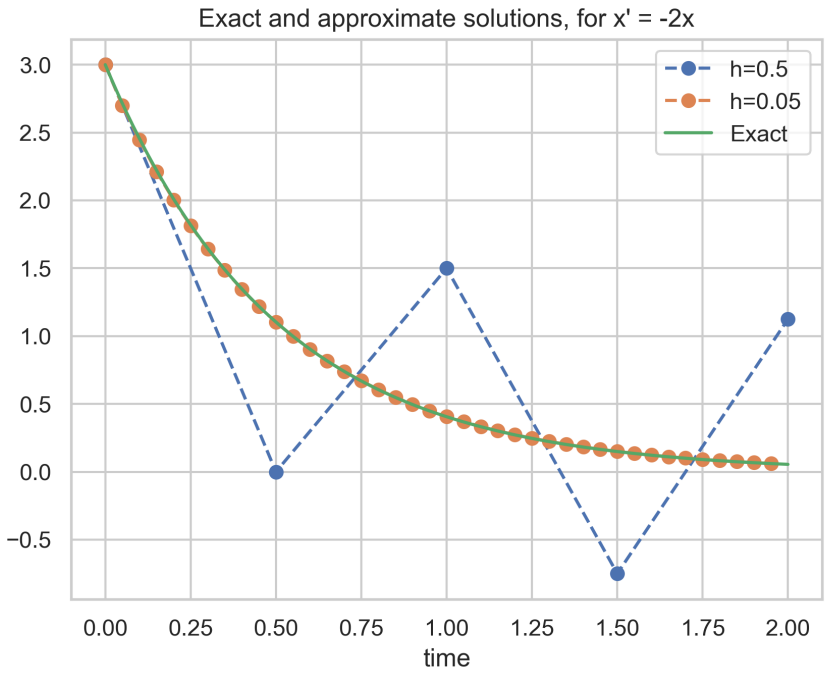


Figure 7: Adams-Bashforth method applied to $x'(t) = -2x$, $x(0) = 3$.

4.5 Runge-Kutta of fourth order (RK4)

The Runge-Kutta methods are of a whole family of methods, nonetheless I will only present the Runge-Kutta method of fourth degree which is the most popular. I will not present the derivation and jump immediately to the analysis. The method is best written as

$$
\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f(t_n + \frac{1}{2}h, y_n + k_1\frac{h}{2}), \\
k_3 &= f(t_n + \frac{1}{2}h, y_n + k_2\frac{h}{2}), \\
k_4 &= f(t_n + h, y_n + k_3 h), \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

For a more intuitive description, let's understand what the various k's mean.

- k_1 is the slope at the start of time step $n+1$ (equal to the slope at the end of time step n).
- Following the slope given by k_1 but only taking half a step, we then recalculate the slope at this midpoint and we get k_2.
- Now if we go back to the start of the time step and take half a step following the slope given by k_2 and recalculate again the slope at this point we get k_3.
- Lastly we take a complete step following the slope given by k_3 and the slope at that final point is exactly k_4.
- Taking inspiration from Simpson's rule for approximating integrals we get the last equation.

The last point justifies the fact that this method, like Simpson's rule on a single interval, has a local truncation error of fifth order $O(h^5)$, and so is considered of fourth order.
The last equation can also be seen as a weighted average of the slopes that we calculated in the previous steps. With this perspective in mind, we notice that the slope at the middle points, denoted k_2 and k_3, have a higher weight compared to the other two slopes. If we go back to the derivation of Simpson's method we can see why this would be the case. When we interpolate the function f, by construction the interpolant agrees exactly with the function at the midpoint and the extrema of the subinterval. If the subinterval is small enough and the function is sufficiently smooth it wouldn't be so far-fetched to say that the slopes of f and the polynomial are similar at the midpoint. This comes from the fact that the function cannot vary too drastically inside the subinterval we are considering because of regularity. Furthermore, at the extrema even though the function and the polynomial agree, they may have very different slopes because in the interpolation over a single subinterval we do not consider any information we have about the function outside the subinterval. Actually, using spline interpolation there is no guarantee that we can even differentiate the interpolating polynomial at the extrema of the subinterval. The Jupyter Notebook in [1] illustrates an example of this.
When we apply Example 4.1 to RK4 we are left with the following plot.

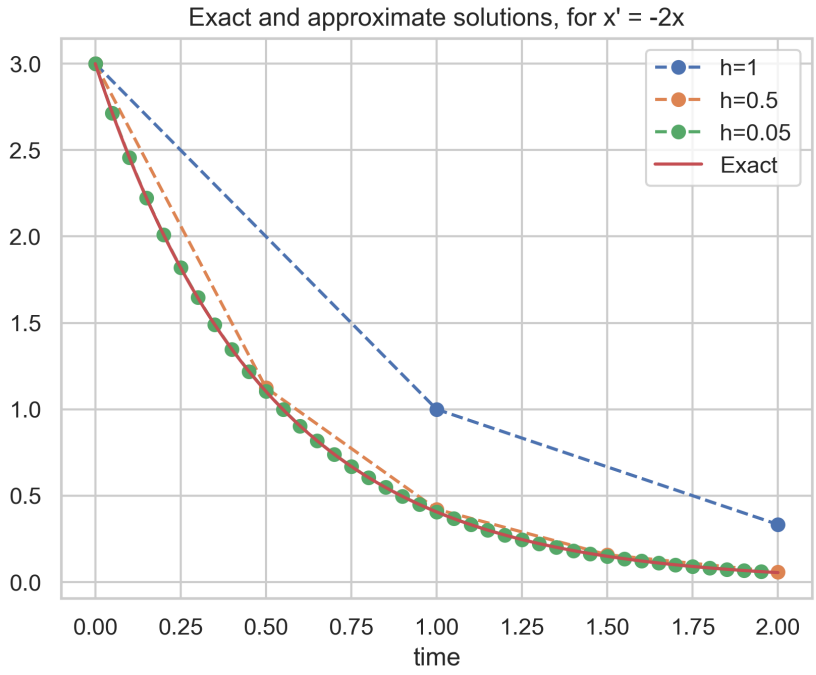


Figure 8: Runge-Kutta method of fourth order applied to $x'(t) = -2x$, $x(0) = 3$.

5 Stability analysis of linear multistep methods

Most of the methods we have seen thus far are all examples of linear multistep methods with the only exception being the Runge-Kutta method. This is an important distinction to make as it affects how we perform the stability analysis. In this section I will focus on linear multistep methods, therefore all the definition will be tailored to these kinds of methods. Nevertheless, all the derivations and explanations behind the definitions are presented in such a way that they can easily be adapted to other types of methods. We will see this in actions when we analyse the Runge-Kutta method in Section 6.

A general *r-step linear multistep method* is a method of the form

$$\sum_{j=0}^{r} \alpha_j y_{n+j} = h \sum_{j=0}^{r} \beta_j f(t_{n+j}, y_{n+j}). \tag{5.1}$$

Notice that the trapezoidal method and the second order Adams-Bashforth method are both particular cases of this general equation. To get the trapezoidal method in equation (4.2)

we choose

$$r = 1, \quad \alpha_0 = -1, \quad \alpha_1 = 1 \quad \text{and} \quad \beta_0 = \beta_1 = \frac{1}{2}. \tag{5.2}$$

To get Adams-Bashforth seen in equation (4.3), with a slight change of indexing, we must set

$$r = 2, \quad \alpha_0 = 0, \quad \alpha_1 = -1, \quad \alpha_2 = 1, \quad \beta_0 = -\frac{1}{2}, \quad \beta_1 = \frac{3}{2} \quad \text{and} \quad \beta_2 = 0. \tag{5.3}$$

5.1 Global and local truncation error

To start analysing the stability of linear multistep methods we first want to quantify the error our methods make. Errors are inevitable when we try to approximate an exact solution, however we would like to make sure that these errors remain under control and go to zero as our step size goes to zero. The first type of error we are interested in is the *global (truncation) error* which is defined as the exact solution minus the numerical approximation

$$e_n := x(t_n) - y_n$$

This notion is important as it leads to the definition of *convergence*.

Definition 5.1 (Convergence)**.** *We say a numerical method is convergent if, for every initial value problem* (4.1)*,*

$$e_n = x(t_n) - y_n \to 0 \quad \textit{as} \quad n \to \infty \quad \textit{for all} \quad t_n \in [a, b]. \tag{5.4}$$

Notice that this definition is not restricted to linear multistep methods and is valid for general numerical methods.

Next the *local truncation error* for linear multistep methods is defined[1] as the following

$$\tau(t_{n+r}) := \sum_{j=0}^{r} \alpha_j x(t_{n+j}) - h \sum_{j=0}^{r} \beta_j x'(t_{n+j}).$$

We will see the relation between local and global truncation error in Theorem 5.1. Assuming

[1] The definition of local truncation error may vary between texts. We use this definition to remain consistent with the local truncation error used thus far.

that $x(t)$ is smooth enough we can take the Taylor series of $x(t)$ and $x'(t)$

$$x(t_{n+j}) = x(t_n) + jhx'(t_n) + \frac{1}{2}(jh)^2 x''(t_n) + o(h^2),$$
$$x'(t_{n+j}) = x'(t_n) + jhx''(t_n) + \frac{1}{2}(jh)^2 x'''(t_n) + o(h^2).$$

Substituting this back into the local truncation error gives

$$\tau(t_{n+r}) = \left(\sum_{j=0}^{r} \alpha_j\right) x(t_n) + h\left(\sum_{j=0}^{r} (j\alpha_j - \beta_j)\right) x'(t_n) + \tag{5.5}$$
$$+ h^2\left(\sum_{j=0}^{r}\left(\frac{1}{2}j^2\alpha_j - j\beta_j\right)\right) x''(t_n) + o(h^2).$$

Definition 5.2 (Consistent method)**.** *We call a numerical method consistent if* $\tau(t) = o(h)$ *for all* $t \in [a, b]$*. That is,*

$$\frac{\tau(t)}{h} \to 0 \quad \textit{as} \quad h \to 0 \quad \forall t \in [a, b].$$

From (5.5) we see that for consistency to hold we must have that

$$\sum_{j=0}^{r} \alpha_j = 0 \quad \text{and} \quad \sum_{j=0}^{r} j\alpha_j = \sum_{j=0}^{r} \beta_j. \tag{5.6}$$

By further expanding (5.5) we can set the coefficients of higher powers of h to 0, and in so doing, we will be left with a system of equations which we can use to solve for the coefficients α_i and β_i for $i \in \{0, \ldots, r\}$. As a consequence the local truncation error will be of higher order and by Theorem 5.1 also the global error will be of higher order. A derivation of higher order Adams-Bashforth methods using this approach can be found in [2].
An important observation is that the consistency conditions (5.6) only depend on the method and not the actual IVP we are solving. Hence, they hold for any initial value problem we may tackle.

5.2 Characteristic polynomials

For linear multistep methods it is useful to define the two following polynomials called *characteristic polynomials*.

$$\rho(\zeta) := \sum_{j=0}^{r} \alpha_j \zeta^j \quad \text{and} \quad \sigma(\zeta) := \sum_{j=0}^{r} \beta_j \zeta^j.$$

With these definitions in mind we can rewrite the consistency conditions as

$$\rho(1) = 0 \quad \text{and} \quad \rho'(1) = \sigma(1). \tag{5.7}$$

5.3 Zero-stability of a linear multistep method

The next step is introducing the notion of *zero-stability*. This is one of many different notions of stability for numerical methods one can look at. It is very simple but nonetheless very impactful thanks to Theorem 5.1.
To get some intuition behind zero-stability, let's look at the simplest example of IVP.

Example 5.1. *Consider this IVP:* $x'(t) = 0$ *with* $x(0) = 0$.

The obvious solution of this is $x(t) = 0$, but there are many methods that fail to reach this solution given a small initialization error. The next examples will illustrate this very well.
Take the following linear multistep method trying to solve Example 5.1,

$$y_{n+2} - 3y_{n+1} + 2y_n = 0, \qquad y_0 = 0. \tag{5.8}$$

While there is so real justification why someone would use this method, it seems like a perfectly reasonable 2-step linear multistep method. To start the recurrence we need to estimate y_1. Assume that in doing so we get that $y_1 = \varepsilon \neq 0$. With methods we will explore soon, one can solve (5.8) for y_n starting from y_1 and y_0 giving

$$y_n = 2y_0 - y_1 + 2^n(y_1 - y_0).$$

Under our assumption of $y_1 = \varepsilon$ we are left with

$$y_n = (2^n - 1)\varepsilon$$

which obviously diverges as $n \to \infty$.
Why did this happen? Well, in general it comes down to solving a *linear difference equation*

$$\sum_{j=0}^{r} \alpha_j y_{n+j} = 0. \tag{5.9}$$

As y_{n+j} depends on the previous y_i for $i \in \{n, \dots, n+j-1\}$ an educated guess for a solution would be

$$y_{n+j} = k^{n+j}.$$

For some value of $k \in \mathbb{R}$. Some insight as to where this guess comes from can be found in [2]. Plugging back in we get that k must satisfy

$$\sum_{j=0}^{r} \alpha_j k^{n+j} = 0$$

which is exactly like saying that k must be a root of $\rho(\zeta)$. What's more, by the Fundamental Theorem of Algebra $\rho(\zeta)$ must have r roots and as (5.9) is linear, the general solution is

$$y_n = \sum_{j=1}^{r} c_j \zeta_j^n,$$

where ζ_j for $j = 1, \dots, r$ are the roots of $\rho(\zeta)$. To solve for the coefficients we must use the initial conditions. To start the recurrence we need r initial conditions so we can construct a $r \times r$ system of equations. A first idea of stability could be imposing that y_n should not diverge as $n \to \infty$. For this to be to fulfilled we must have that $|\zeta_j| \leq 1$ for all the roots. There is a slight catch if the roots repeat as the system to solve becomes singular and so we must impose that $|\zeta_j| < 1$ if ζ_j is a repeated root to mend this. A more precise explanation of this may be found in [7].
All of this reasoning is the motivation for defining zero-stability.

Definition 5.3 (zero-stability for linear multistep methods)**.** *A r-step linear multistep method is said to be zero-stable if the roots of the characteristic polynomial $\rho(\zeta)$ denoted $\zeta_1, \dots, \zeta_r$*

satisfy the following conditions:

$$|\zeta_j| \leq 1 \quad \textit{for} \quad j = 1, \ldots, r \tag{5.10}$$

$$\textit{If } \zeta_j \textit{ is a repeated root, then } |\zeta_j| < 1. \tag{5.11}$$

The discussion above conveys the intuition behind zero-stability: we say that a numerical method is zero-stable if, given a slight perturbation of the initial condition of Example 5.1, y_n does not diverge as $n \to \infty$ (equivalently $h \to 0$). This important definition as it leads to the following theorem.

Theorem 5.1 (Dahlquist's Equivalence Theorem)**.** *Given a linear multistep method used to solve an initial value problem as in Definition 4.1, under the assumptions of the Picard–Lindelöf Theorem* (A.3) *we have that:*

$$\textit{consistency} \quad + \quad \textit{zero stability} \quad \Longleftrightarrow \quad \textit{convergence}.$$

Moreover, if the solution $x \in C^{p+1}([a, b])$ and local truncation error is of order $O(h^{p+1})$ then the global error if of order $O(h^p)$.

Proof. The proof of this theorem is very intricate and can be found in [4]. ■

This theorem provides theoretical guarantees that for h small enough, if the method is consistent and zero-stable we will converge to the correct result, which is as good as we could hope for.
Zero-stability however is not the only kind of stability we care about. In fact often this stability is not useful. These condition basically tell us that "eventually" for h small enough we will be very close to the exact answer. How small does h have to be for us to be arbitrarily close to the correct value? Actually, it can also happen that for values of h not small enough the error is extremely large. Essentially there are no guarantees that a given h will give a small error. To dive deeper into stability analysis we will introduce *absolute stability*.

5.4 Absolute stability for linear multistep methods

This notion of stability focuses on a specific linear example of ODE. We will call this the *test equation*.

$$x'(t) = \lambda x(t). \tag{5.12}$$

While this equation seems extremely simple, by varying λ we can really put to the test many of the methods we have seen. If we take one step further and take λ to be complex we can visualize regions of absolute stability in the complex plane. Why do we care so much about equation (5.12) which has a very simple solution $Ce^{\lambda t}$? Firstly, it can be generalized to include linear system of differential equations. In this case all we have to do is calculate the eigenvalues of the matrix representing the system and, assuming we have distinct eigenvalues, we can apply absolute stability to each eigenvalue in order to conclude absolute stability over the whole system. Actually it can be even generalized to non-linear systems, by locally linearising the system and applying the same idea as for linear system of differential equations. The second reason for its importance is that this is the simplest example of what's called a stiff differential equation. If, for example, we take λ to be very large, then the step the numerical method makes is multiplied by λ and might lead to an overshooting of the real value. Hence, the test equation causes many numerical methods to behave unstably if h is not chosen small enough impacting their convergence to the correct value. This is clearly seen in Example 4.1 applied to AB2 when $h = 0.5$. We notice that this can be mitigated by choosing h small enough to counteract the impact λ has on the method. For example applying the explicit Euler method to (5.12) gives

$$y_{n+1} = (1 + h\lambda)y_n. \tag{5.13}$$

For the explicit Euler method to be absolutely stable we must have that $|1 + h\lambda| \leq 1$. Note that while λ is complex we still take h to be real. It is evident that the important aspect here is the product of h and λ. This is perfectly inline with what we discussed above since taking h small can reduce the effect of λ. So the important aspect to monitor is the product λh and it is precisely this that we are going to plot to find what is called the *absolute stability regions* of the numerical methods.

In the case of explicit Euler the stability region comes out to be a circle of radius 1 centred

at $(-1, 0)$ as we can see in Section 5.5.

Using (5.1) to solve our test problem and rearranging we obtain

$$\sum_{j=0}^{r} (\alpha_j - h\lambda\beta_j)\, y_{n+j} = 0. \tag{5.14}$$

Similar to the previous example, the important parameter to consider is the product of h and λ. Setting $z = h\lambda$ we define the stability polynomial defined as

$$\pi(\zeta; z) = \rho(\zeta) - z\sigma(\zeta),$$

where $\pi(\zeta; z)$ is a polynomial in ζ and whose coefficients depend on z. The essential point in this argument is that equation (5.14) is identical to equation (5.9) with slightly different coefficients. Using the exact argument we did in Section 5.3 with $\pi(\zeta; z)$ instead of $\rho(\zeta)$ we get another kind of stability: *Absolute stability*.

Definition 5.4 (Absolute stability)**.** *A r-step linear multistep method is said to be absolutely stable if the roots of the stability polynomial $\pi(\zeta; z)$ denoted $\zeta_1, \ldots, \zeta_r$ satisfy the following conditions:*

$$|\zeta_j| \leq 1 \quad \textit{for} \quad j = 1, \ldots, r \tag{5.15}$$

$$\textit{If } \zeta_j \textit{ is a repeated root, then } |\zeta_j| < 1. \tag{5.16}$$

We are now ready to plot some stability regions of some numerical methods.

5.5 Particular case: Euler methods

Explicit Euler: We are left with the following characteristic polynomials

$$\rho_{EE}(\zeta) = -1 + \zeta \quad \text{and} \quad \sigma_{EE}(\zeta) = 1 \tag{5.17}$$

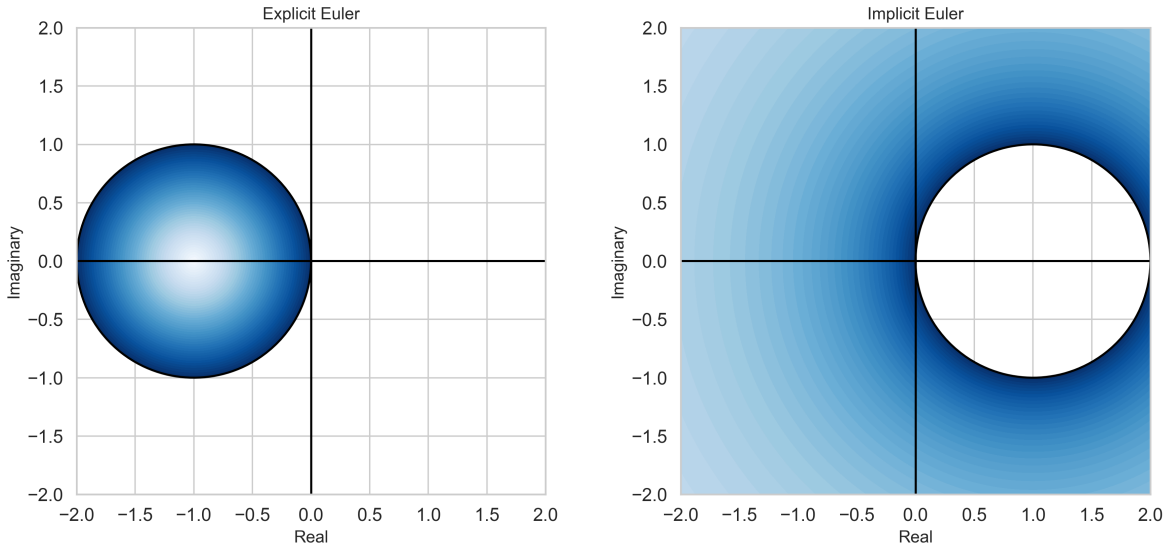
where $\rho_{EE}(\zeta)$ has a single root namely $\zeta_1 = 1$. As the conditions (5.7) are satisfied we conclude that the explicit Euler method is zero-stable and consistent which, by the previous theorem, is equivalent to being convergent.

Implicit Euler: This method has the following characteristic polynomials

$$\rho_{IE}(\zeta) = -1 + \zeta \quad \text{and} \quad \sigma_{IE}(\zeta) = \zeta \tag{5.18}$$

Where $\rho_{IE}(\zeta)$ has a single root $\zeta_1 = 1$. Similarly to the explicit Euler method also the implicit Euler is zero-stable and consistent, which is to say convergent.
We have already seen the derivation for the stability region of the explicit Euler method. Deriving the implicit Euler method stability region is very similar.

Figure 9: Absolute stability regions for the Euler methods.



(a) Explicit Euler method.

(b) Implicit Euler method.

5.5.1 Discussion of the Euler methods

Euler methods come out naturally from approximating the derivative using the difference quotient. Their simplicity means that it takes very little computational power to run as we are only calling the function once (which might become be expensive for complicated functions) and performing basic operations. This simplicity however comes at a cost of instability. Just by looking at the stability region of the explicit Euler method. This means that h must be

chosen very well to avoid instability and often h might have to be so small that the number of computations needed may become too large. Often computers have trouble dealing with very small numbers as this will likely lead to floating point number approximations.
Backwards Euler on the other hand does not suffer from a small stability region. Nevertheless, slow convergence and the high chance of low accuracy remain very prominent issues. Furthermore, it is an implicit method which comes with some serious drawback as we shall see in the discussion of the trapezoidal method, Section 5.6.1.

5.6 Particular case: Trapezoidal method for IVPs

To check if the method is zero-stable let's first find its characteristic polynomials.

$$\rho_T(\zeta) = -1 + \zeta \quad \text{and} \quad \sigma_T(\zeta) = \frac{1}{2} + \frac{1}{2}\zeta \tag{5.19}$$

where $\rho_T(\zeta)$ and has root $\zeta_1 = 1$. With these characteristic polynomials in mind we can easily state that the trapezoidal method is both zero-stable and consistent, and so convergent.
From the characteristic polynomials found in (5.19) we have that the stability polynomial

$$\pi_T(\zeta; z) = -\left(1 + \frac{1}{2}z\right) + \left(1 - \frac{1}{2}z\right)\zeta$$

has a single root,

$$\zeta_1 = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}$$

and taking the absolute value we get

$$|\zeta_1| \leq 1 \iff \left|1 + \frac{1}{2}z\right| \leq \left|1 - \frac{1}{2}z\right| \iff |2 + z| \leq |2 - z|$$

which is like saying all the complex numbers z that are closer to -2 than to 2, giving us the left half plane as below.

5.6.1 Discussion of the trapezoidal method

The trapezoidal method came directly from the trapezoid method for numerical integration. The remarkable property of this method is the fact that its stability region is the half plane.

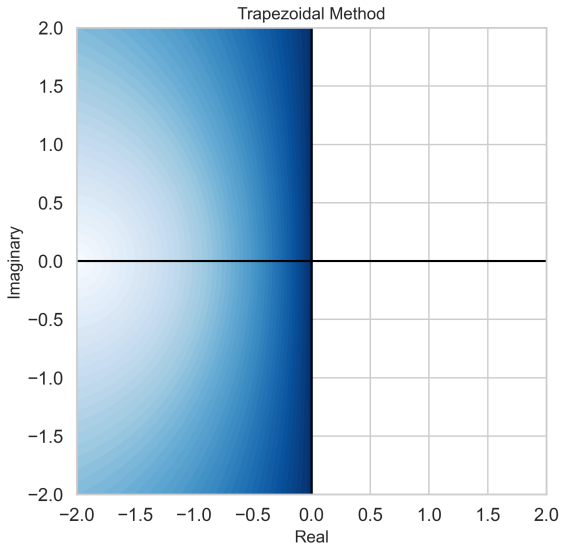


Figure 10: Absolute stability region for the trapezoidal method.

It turns out that it is impossible to have an explicit method with this property and, what's more, the only linear multistep method that achieves this must have order at most 2. The trapezoidal method turns out to be the most accurate among all of these. The most evident problem of this method however is the fact that we need to solve an equation (that depends on f) to implement the step. Solving equations numerically is a whole other branch of numerical analysis, however all we really care about it that solving equations numerically may bring with them errors that were not accounted for in the error analysis. Furthermore, implicit methods also need more initial points to start the calculation. Often this is done by using very simple methods to get the first few points such as the Euler methods. The problem is that this may lead to possible errors in the starting conditions, which is obviously not ideal. Luckily we introduced zero-stability to make sure this error does not explode, but still errors in the starting conditions may propagate through the steps and end up hurting our final accuracy.

5.7 Particular case: Adams-Bashforth

Similarly to the trapezoidal method, also Adams-Bashforth is consistent. Furthermore, its polynomial is

$$\rho_{AB}(\zeta) = -\zeta + \zeta^2 \quad \text{and} \quad \sigma_{AB}(\zeta) = -\frac{1}{2} + \frac{3}{2}\zeta.$$

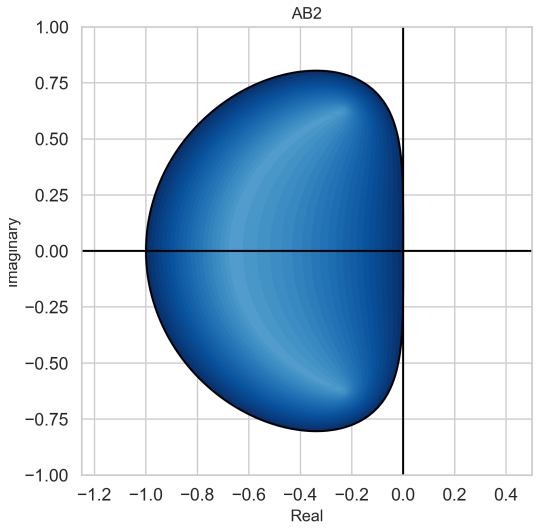
Since $\rho_{AB}(\zeta)$ has two roots $\zeta_1 = 0$ and $\zeta_2 = 1$ we conclude that also this method is convergent.

The stability polynomial of the Adams-Bashforth method of order 2 is

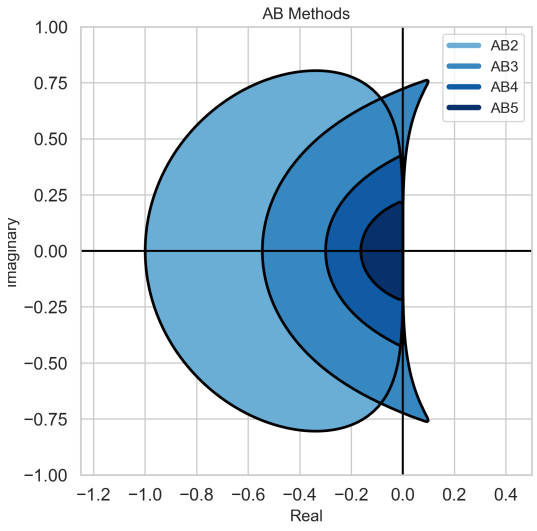
$$\pi_{AB}(\zeta; z) = \frac{1}{2}z + \left(-\frac{3}{2}z - 1\right)\zeta + \zeta^2.$$

In this case checking the root condition may become complicated, and more so for higher orders. In this case we can resort a computer to display the region of stability. Thanks to the construction of the stability polynomial it is very easy to generalize the program to Adams-Bashforth of any order. In Figure 11 I illustrate the stability regions for the Adams-Bashforth methods up to order 5.

Figure 11: Absolute stability regions for the Adams-Bashforth methods.



(a) Adams-Bashforth of order 2.



(b) Adams-Bashforth of order 2 through to order 5.

5.7.1 Discussion of the Adams-Bathforth method

After having slightly varied the derivation of the trapezoidal method we discovered another method: Adams-Bashforth (which belongs to a whole family of methods). This method is particularly interesting as it only uses one extra function call compared to the explicit Euler method, but the improvement is quite substantial. Despite this if we try to use higher order AB methods we see that the stability region becomes extremely small leading to a very high chance of instability.

6 Stability analysis of Runge-Kutta of fourth order

We now move on to the stability of the Runge-Kutta of fourth order. Here I present the derivation only for the fourth order method, but the reasoning can be generalized. The ideas behind zero-stability and absolute stability are carried over also to this method but we need a slightly different approach. For example, it is clear from Section 4.5 that if we apply the RK4 method to the differential equation behind the definition of zero-stability (Example 5.1) we observe that we get the correct solution with no chance of diverging, so the RK4 method is zero-stable.
In the footsteps of absolute stability for multistep methods, we will also apply RK4 to the differential equation defined as

$$x'(t) = \lambda x(t). \tag{6.1}$$

By direct computation we get the following values for k_1, k_2, k_3, and k_4.

$$\begin{aligned} k_1 &= \lambda y_n, \\ k_2 &= y_n \left(\lambda + \frac{\lambda^2 h}{2} \right), \\ k_3 &= y_n \left(\lambda + \frac{\lambda^2 h}{2} + \frac{\lambda^3 h^2}{4} \right), \\ k_4 &= y_n \left(\lambda + \lambda^2 h + \frac{\lambda^3 h^2}{2} + \frac{\lambda^4 h^3}{4} \right). \end{aligned} \tag{6.2}$$

Putting everything together we get the value of y_{n+1}.

$$y_{n+1} = y_n \left[1 + \lambda h + \frac{(\lambda h)^2}{2} + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{24} \right].$$

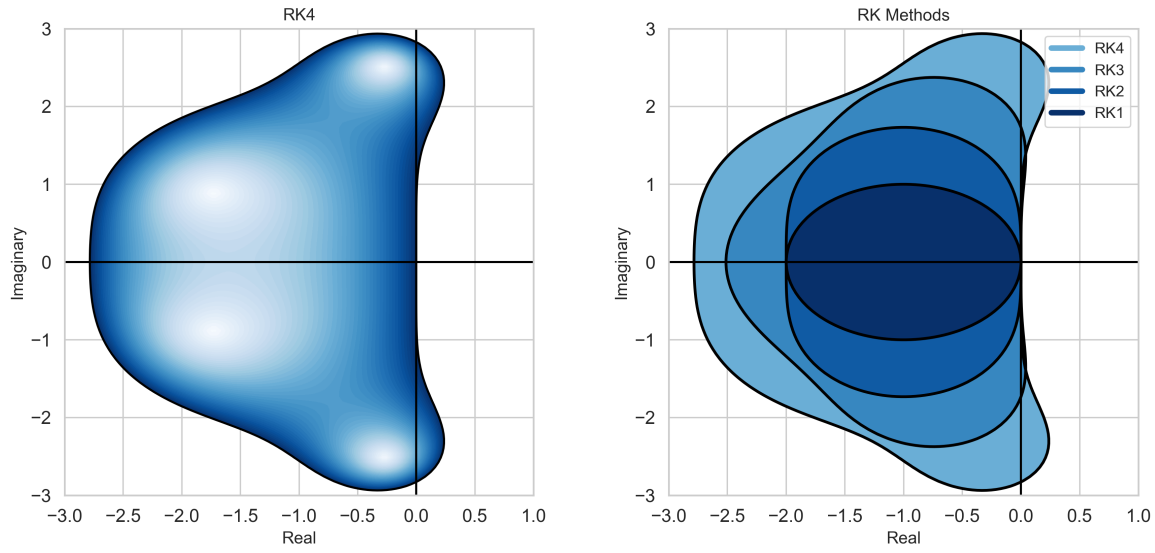
The spectacular part about this is that the multiplicative coefficient of y_n is exactly the fourth order Taylor expansion of $e^{\lambda h}$ which, up to a multiplicative constant, is the correct solution to (6.1).

Using $z = \lambda h$ as above, let us define

$$R(z) := 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!}.$$

Similarly to how we analysed the absolute stability of the explicit Euler method in equation (5.13), for the RK4 method to be absolutely stable we need that the coefficient of y_n does not blow up to infinity, which is to say $|R(z)| \leq 1$. Plotting this inequality similarly to what we did with the other methods results in the following graph.

Figure 12: Absolute stability region for Runge-Kutta methods



(a) Runge-Kutta method of fourth order

(b) Runge-Kutta of order 1 through to order 4

6.1 Discussion of Runge-Kutta of fourth order

The RK4 method is a very good compromise between complexity and stability. We can see that this method performs remarkably well in many examples and, taking special care in noticing the scale of the axes we observe that the RK4 method has a very large stability region. Moreover, we notice that in fact the stability regions gets bigger as we increase the order of the method. By not being a multistep method there is no need to estimate the first few points which means it is "self-starting". Ultimately what makes it powerful is the fact that it takes the weighted average of various slopes before taking the step. This makes it more robust and helps to lower any errors that may come up from calculating a single slope. All of this obviously comes at a cost. Usually RK4 takes more computational time than multistep methods of comparable accuracy and this becomes more evident when we try to solve systems of ODEs. Lastly, the error estimation of the RK methods is quite messy and hard to analysis.

7 Concluding remarks

Throughout this paper I have presented some of the simplest and most common numerical methods for solving ODEs, each with its advantages and disadvantages. The reason why there are so many methods is that no method is perfect and ultimately choosing the best method may depend on the problem at hand.

After having done of all this analysis, a natural question comes up: why don't we use a method of higher order? This seems like a very reasonable question, in fact the stability region of the Runge-Kutta method seems to keep getting bigger, so why don't we just keep increasing the order? Well the first problem is that it is even very complicated just to come with the higher order methods! Take the Runge-Kutta family for example, even though I did not present the derivation, the formula comes from solving a system of equations. To derive higher order methods the number of equations in the system blows up. For example fifth order RK would require sixteen equations. Actually in [6] the authors point out that this approach to deriving higher order Runge-Kutta methods can only be done efficiently up to order eight. Even if it were possible to find an other higher order numerical method they usually become very complicated and harder to implement, which in turn brings with it

more computation and so longer run times, not to mention the extremely complicated error analysis that would come with it. All of this effort would only bring a slight improvement to the best numerical methods we have so far. In most applications the accuracy of the methods we already have is sufficiently high and increasing the order of methods does not necessarily mean that we would necessarily always get better results. Take for example the Adams-Bashforth methods. So in summary, the effort of discovering and implementation makes using higher order methods hard to justify.

A Appendix: Supplementary Theorems

Theorem A.1 (Weierstrass Approximation Theorem)**.** *Suppose $f \in C([a,b])$ is given. For every $\varepsilon > 0$, there exists a polynomial function $p(x)$ such that*

$$\sup_{x\in[a,b]} |f(x) - p(x)| < \varepsilon.$$

Proof. [10] presents a proof of this theorem. ■

Theorem A.2 (Generalized Integral Mean Value Theorem)**.** *Suppose $f, g : \mathbb{R} \longrightarrow \mathbb{R}$ with $f, g \in C([a,b])$. Furthermore, assume g does not change sign on $[a,b]$. Then there exists a point $\xi \in [a,b]$ such that*

$$\int_a^b g(x)f(x)\,dx = f(\xi)\int_a^b g(x)\,dx.$$

Proof. This is a generalization of the Integral Mean Value theorem. More details and a proof are discussed in [3]. ■

Theorem A.3 (Picard–Lindelöf)**.** *Let $f : \Omega \longrightarrow \mathbb{R}^n$ and suppose $f \in C(\Omega)$ where Ω is an open subset of $\mathbb{R}^{n+1}$, and $(t_0, x_0) \in \Omega$. If f is locally Lipschitz continuous in the second argument and uniformly continuous with respect to the first, then there exists a unique local solution $x(t) \in C(I)$ of the IVP* (4.1)*, where I is some interval around t_0.*

Proof. The proof can be found in [11]. ■

Theorem A.4 (Dominated convergence theorem for Riemann integrals)**.** *Let* $f_1, f_2, \ldots : [a,b] \longrightarrow \mathbb{R}$ *be a sequence of Riemann-integrable functions, which converges on* $[a,b]$ *to a Riemann-integrable function* f*. If there exists a constant* $M > 0$ *satisfying* $|f_n(x)| < M$ *for all* $x \in [a,b]$ *and for all* $n \in \mathbb{N}$*, then*

$$\lim_{n\to\infty} \int_a^b |f_n(x) - f(x)|\, dx = 0.$$

In particular,

$$\lim_{n\to\infty} \int_a^b f_n(x)\, dx = \int_a^b \lim_{n\to\infty} f_n(x)\, dx = \int_a^b f(x)\, dx.$$

Proof. The proof is presented in [8]. ■

Bibliography

[1] Mattia Nino Barbiere. *Numerical methods for computing integrals and solving ODEs*. URL: https://github.com/MattiaBarbiere/Numerical-methods-for-computing-integrals-and-solving-ODEs.

[2] J.C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2016. ISBN: 9781119121503. URL: https://books.google.it/books?id=JlSvDAAAQBAJ.

[3] J.F. Epperson. *An Introduction to Numerical Methods and Analysis*. Wiley, 2013. ISBN: 9781118367599. URL: https://books.google.it/books?id=310lAgAAQBAJ.

[4] W. Gautschi. *Numerical Analysis*. SpringerLink : Bücher. Birkhäuser Boston, 2011. ISBN: 9780817682590. URL: https://books.google.it/books?id=-fgjJF9yAIwC.

[5] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2008. ISBN: 9783540566700. URL: https://books.google.it/books?id=F93u7VcSRyYC.

[6] David Ketcheson and Umair bin Waheed. "A comparison of high-order explicit Runge–Kutta, extrapolation, and deferred correction methods in serial and parallel". In: *Communications in Applied Mathematics and Computational Science* 9.2 (June 2014), pp. 175–200. ISSN: 1559-3940. DOI: 10.2140/camcos.2014.9.175. URL: http://dx.doi.org/10.2140/camcos.2014.9.175.

[7] R.J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, 2007. ISBN: 9780898717839. URL: https://epubs.siam.org/doi/book/10.1137/1.9780898717839.

[8] W. A. J. Luxemburg. "Arzela's Dominated Convergence Theorem for the Riemann Integral". In: *The American Mathematical Monthly* 78.9 (1971), pp. 970–979. ISSN: 00029890, 19300972. URL: http://www.jstor.org/stable/2317801 (visited on 06/07/2024).

[9] D.E. Stewart. *Numerical Analysis: A Graduate Course*. CMS/CAIMS Books in Mathematics. Springer International Publishing, 2022. ISBN: 9783031081217. URL: https://books.google.it/books?id=twafEAAAQBAJ.

[10] E. Süli and D.F. Mayers. *An Introduction to Numerical Analysis*. An Introduction to Numerical Analysis. Cambridge University Press, 2003. ISBN: 9780521007948. URL: https://books.google.it/books?id=hj9weaqJTbQC.

[11] G. Teschl. *Ordinary Differential Equations and Dynamical Systems*. Graduate studies in mathematics. American Mathematical Soc. ISBN: 9780821891056. URL: https://www.mat.univie.ac.at/~gerald/ftp/book-ode/ode.pdf.

[12] C. Vuik et al. *Numerical Methods for Ordinary Differential Equations*. DAP, Delft Academic Press, 2015. ISBN: 9789065623737. URL: https://books.google.it/books?id=-g2TrgEACAAJ.