SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Derivation of Runge-Kutta order conditions

av

Niklas Hedberg

2014 - No 1

Derivation of Runge-Kutta order conditions

Niklas Hedberg

Självständigt arbete i matematik 15 högskolepoäng, Grundnivå

Handledare: Ivan Martino

2014

Derivation of Runge-Kutta order conditions

Niklas Hedberg

January 23, 2014

Abstract

In the field of numerical analysis to solve Ordinary Differential Equations (ODEs), Runge-Kutta (RK) methods take a sequence of first order approximations of the ODE and weights them in a linear combination for each time step. Given existence and uniqueness criteria, the numerical solution can therefore approximate the theoretical solution to a great deal of accuracy. The point of interest when constructing these methods is thus to ensure convergence. In order to do this, one compares the Taylor expansions of the "true" solution with that of the numerical. One matches the two up to and including the order of a particular derivative, we gain a RK method of that order. The strive for higher orders makes this matching difficult, and this paper concerns the derivation of the conditions required to construct a method of a certain order. This is done by connecting the Taylor expansions with rooted trees. From this, we also get a partial result on the compact relation:

Theorem. *A Runge-Kutta method is of order p if and only if*

$$\sum_j b_j \Phi_j(\boldsymbol{t}) = \frac{1}{\gamma(\boldsymbol{t})}$$

for all trees $\boldsymbol{t}$ *of order* $\leq p$.

Acknowledgements

I wish to express my deepest gratitude to my advisor Ivan Martino for his constant support and guidance. With his directions I have come to appreciate the vast field of numerical analysis, its interconnections and applications. For that, I am much obliged. I would also like to thank Yishao Zhou for her valuable comments and insights.

Contents

Introduction

In the theory of Ordinary Differential Equations (ODE), early emphasis of study was finding solutions in the form of elementary functions. To do this, methods were pioneered by the likes of Newton and Leibeniz. However, it was soon to be found that most ODEs were if not difficult, impossible to solve by the means at hand. This should come as no surprise, as taking the derivatives of elementary functions produce elementary functions. However taking integrals of elementary functions do not guarantee an elementary primitive function. Resultantly it was in the early 19th century that solutions as the focal point of study was abandoned. The work of Cauchy, who in his strive for making rigorous error estimates in series solutions and Euler's polygon method allowed the statements "Does a solution exist?" and "Is the solution unique?" to be answered. To a mathematician, mere knowledge of the existence and uniqueness of a solution might provide a great deal of insight, without knowing the actual solution. However, due to the immense importance of ODEs in the applied sphere of mathematics, this would be unacceptable to any physicist, chemist or yet even Wall Street analysts - they need their solutions! Given the existence and uniqueness criteria and this urge for finding a solution, numerical methods have a been a solid bridge between the two.

In this paper we examine the popular Runge-Kutta (RK) schemes to provide a numerical solution. These have proved their effectiveness for little over over a hundred years and are still a source of mathematical exploration. The structure of the paper hints at the historical progress in the development of the Runge-Kutta method. We begin with a treatment of ODEs, with an emphasis on transformations into systems. We then treat existence theory in brief, acquainting ourselves with the general conditions and proof of the Picard and Lindelöf theorem. The first numerical method we study in detail is the Euler method, examining its derivation and convergence. The Runge-Kutta method is then introduced. By a careful analysis of its derivation and connection to Taylor series we define the order conditions - the set of equations that the coefficients of the RK method have to satisfy in order to guarantee convergence to the "true" solution. We thereby proceed to lend the rest of the paper to the task of proving the order conditions using a beautiful connection between series solutions and trees.

As most mathematical exploration is based on accumulated knowledge and communal effort, the results and proofs of the first sections involve many well known results regarding ODEs, one step methods and tensors. The proofs presented are written from my own knowledge and synthesis of ideas, but are by no means original or

contrasting to traditional treatments. The section on the derivation follows mainly from Harier's [7] treatment of the topic, which is not completely self contained. Resultantly, the addition of Lemma 4 (due to Butcher [2]) has allowed the independent proof of theorem 7, which is my own.

We assume the reader knows some basic notions of topology (including definitions of open sets, metric spaces, norms etc.) and some elementary notions of multilinear maps and tensors (we provide a brief summary in section 6). Topics from analysis such as convergence and the Weierstrass M-test are also assumed. The emphasis on theoretical results of convergence has diminished the important treatments of implementation and stability analysis. For the novice reader, some elementary notions have been presented in the appendix to showcase the strengths and weaknesses of the methods when implemented.

1 Ordinary Differential Equations

In general terms, a differential equation is an equation relating some function $y(t)$ to one or more of its derivatives. These are related by binary and unitary operations. They depend on one or more independent variables, coefficients from different fields such as that of the real or complex numbers, raised to a power and the list goes on. A familiar example is a function which is equal to its derivative:

$$y(t) = \frac{dy(t)}{dt}. \tag{1}$$

Notation. The convention in this paper will be to write the n'th derivative of $y(t)$ with respect to t as $y^{(n)}$. Furthermore when $y(t) \in \mathbb{R}$ and $y'(t) = f(t, y(t))$ we will often omit t and write $y' = f(t, y)$.

For what concerns this thesis we are primarily going to work over the real numbers, $\mathbb{R}$, and we move to the following definition of an ODE.

Definition 1. *(Ordinary Differential Equation)*
Let Ω be a open set in $\mathbb{R} \times \mathbb{R}^n$ and let F be a function $F : \Omega \to \mathbb{R}^n$. Then, an Ordinary Differential Equation (ODE) is the equation $F(t, y, y^{(1)}, \ldots, y^{(n)}) = 0$. If we can write the previous equation as $y^{(n)} = F(t, y, y^{(1)}, \ldots, y^{(n-1)})$ this is called explicit of order n, otherwise it is called implicit of order n.

Example. $F(t, y) = y'$.
This equation models the velocity of an object in classical kinematics. It is worth to note that this seemingly simple encapsulation of the rate of change of position y with respect to time would be the springboard for the understanding of the universe. Momentum and energy are two pillars which rests on the understanding of this equation. It is therefore interesting that it turned out to model velocity incompletely after Einstein's special theory of relativity.

Example. $y'' = -(\frac{k}{m})y$ *or* $F(t, y, y', y'') = y'' + (\frac{k}{m})y$.
This equation is often referred to as Hooke's Law, or the spring law. As it models particles under the action of springs. The constant $(\frac{k}{m})$ *is the ratio of the spring constant (stiffness) to the mass of the particle under its action. Its application ranges from macroscopic objects to the fundamental structures of atomic interactions. Understanding this equation provides models of heat transfer in solids, to the underpinnings of space itself in Quantum Field Theory.*

Example. $y'' - 2ty' + 2py = 0$ *or* $F(t, y, y', y'') = y'' - 2ty' + 2py$.
The famous Schrödinger equation of Quantum Mechanics can be written in this form, which is called the Hermite equation. The solutions of this equation gives rise to the quantised nature of the Harmonic oscillator (a solution to Hookes Law above). It gives deep insights into the structure of nature on small scales.

When we speak of solutions to the differential equation we mean the function $y(t) : \mathbb{R} \to \mathbb{R}^n$ such that if this function is substituted into the given equation, equality occurs. Of course, we can ask ourselves, is $y(t)$ "the" solution or does it solve the equation only up to an arbitrary function of t? If it does, is it unique? In order to be sure about the answer to these questions one must usually introduce an initial condition $(t_0, y_0) \in \Omega$ together with the ODE. Here, y_0 is interpreted as a vector when we work in higher dimensions. This is known as an Initial Value Problem (IVP) and its implications will be discussed at a later stage.

Example. *Let* $y' = 2t + e^{2t}$ *be the ODE with initial condition* $t = 0$ *and* $y(0) = 1$
By method of separating variables we write in Leibniz notation

$$\begin{aligned} \frac{dy}{dt} &= 2t + e^{2t} \\ dy &= (2t + e^{2t})dt. \end{aligned}$$

Where dy *and* dt *are the differentials of* $y(t)$ *and* t *respectively. Taking an integral we get*

$$y(t) = \tfrac{1}{2}(t^2 + e^{2t}) + C$$

where C *is a real valued constant. Using the initial values* $y(0) = 1$*, we have* $1 = \frac{1}{2} + C$ *and so we get* $C = \frac{1}{2}$*. Therefore,*

$$y(t) = \tfrac{1}{2}(t^2 + e^{2t} + 1).$$

Example. *(Hooke's Law) We have already in passing emphasised the fundamental nature of this law and its solution . Let* $(\frac{k}{m}) \in \mathbb{R}$ *and we study*

$$y'' + (\tfrac{k}{m})y = 0.$$

If we call $\frac{k}{m} = a^2$ *and use the characteristic polynomial we get the equation:*

$$r^2 + a^2 = 0.$$

This has the following solutions $r = ai$ *and* $r = -ai$. *Then* $y(t) = c_1 e^{iat} + c_2 e^{-iat}$ *with* $c_1, c_2 \in \mathbb{R}$*, which is a Simple Harmonic Oscillator*[1] *with natural frequency* a.

If a given ODE is of order $n > 1$ we can transform the equation to a system of first order equations. For instance, consider a second order equation of the form:

$$y'' = f(t, y, y') \tag{2}$$

together with the initial conditions $y(t_0) = y_0$ and $y'(t_0) = y'_0$.
We can indeed transform this into a first order linear system through a series of substitutions. If we set $y = y_1$ and $y_2 = y'$ then $y'_2 = y''$. Equation (4) therefore becomes:

$$y'_2 = f(t, y_1, y_2).$$

This defines a linear system as the following lemma explicates.
Lemma 1. *Any explicit ODE of order* n *can be transformed to a first order system of differential equations.*

Proof. Suppose that an n-th order ODE is solved for the n-th derivative: we write this in the form $y^{(n)} = F(t, y, y^{(1)}, \ldots, y^{(n-1)})$. Then we convert it into a system by this proposed scheme of variables $y_i = y^{(i-1)}$.

$$\text{For } 1 \le i \le n, \begin{cases} y_1 = y \\ y_2 = y^{(1)} \\ \vdots \\ y_n = y^{(n-1)}. \end{cases} \qquad \text{This gives the system } \begin{cases} y'_1 = y_2 \\ y'_2 = y_3 \\ \vdots \\ y'_n = F(t, y_1, y_2, \ldots, y_n). \end{cases}$$

If we think of $y' = F(t, y)$ as a first order system with i'th component $F_i(t, y) = y_{i+1}$ for $i < n$ and $F_n(t, y) = F_n(t, y_1, y_2, \ldots, y_n)$. □

[1]For a treatment of a wide range of applications of the simple harmonic oscillator we refer to[6].

To be more explicit, let Ω be an open set in $\mathbb{R} \times \mathbb{R}^n$ and let F be a *nice*[2] function $F : \Omega \to \mathbb{R}^n$. If $y'(t) = F(t, y(t))$ This means that

$$y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix} \text{ and } \quad y'(t) = \begin{bmatrix} y_1'(t) \\ \vdots \\ y_n'(t) \end{bmatrix},$$

and so one has:

$$\begin{bmatrix} y_1'(t) \\ \vdots \\ y_n'(t) \end{bmatrix} = \begin{bmatrix} f_1(t, y(t), \ldots, y_n(t)) \\ \vdots \\ f_n(t, y(t), \ldots, y_n(t)) \end{bmatrix}. \tag{3}$$

Example. *We consider the ODE* $y^{(3)} + 2y^{(2)} - y^{(1)} - 2y = 0$*, where* $F : \mathbb{R} \times \mathbb{R}^4 \to \mathbb{R}$*. We convert this into an ODE system by our scheme of changing variables.*

$$\begin{aligned} y_1' &= y^{(1)} = y_2 \\ y_2' &= y^{(2)} = y_3 \\ y_3' = y^{(3)} &= 2y + y^{(1)} - 2y^{(2)} = 2y_1 + y_2 - 2y_3. \end{aligned}$$

So $y' = Ay$ *with the matrix*

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 1 & -2 \end{bmatrix}.$$

When we consider the domain $\Omega \subset \mathbb{R} \times \mathbb{R}^n$, we consider a variable t and n other variables that depend on t i.e. $y_1(t), ..., y_n(t)$. We can get rid of this separation and consider all variables with equal importance. In this way F will not depend on the variable t.

Definition 2. *A first order system of ODEs is called autonomous if each component of the system doesn't explicitly depend on the variable* t*:*

$$y'(t) = f(y_1(t), \ldots, y_n(t)).$$

[2]We consider "nice" a function F such that $y'(t) = F(t, y(t))$ has a solution. We will specify what we need in section 2. But for this example one can consider that F has sufficiently many derivatives to write our system.

Lemma 2. *Any first order ODE system can be written as an autonomous system.*

Proof. We consider an n dimensional, non-autonomous ODE system of the following form:

$$\begin{bmatrix} y_1'(t) \\ \vdots \\ y_n'(t) \end{bmatrix} = \begin{bmatrix} f_1(t, y(t), \ldots, y_n(t)) \\ \vdots \\ f_n(t, y(t), \ldots, y_n(t)) \end{bmatrix}.$$

We augment the dimension to $n+1$ by introducing the variable $y_0(t) = t$. This allows us to consider the equivalent system, which has no explicit dependence on t:

$$\begin{bmatrix} y_0'(t) \\ y_1'(t) \\ \vdots \\ y_n(t)' \end{bmatrix} = \begin{bmatrix} 1 \\ f_1(y_0(t), y(t), \ldots, y_n(t)) \\ \vdots \\ f_n(y_0(t), y(t), \ldots, y_n(t)) \end{bmatrix}.$$

□

We have now learnt that it is possible to regard an arbitrary ODE of any order as a system of first order autonomous ODEs. This generalisation allows us to construct general solution methods for first order equations that work for a vast number of problems of even higher order. Before any such methods are constructed, we must be familiar with the existence and uniqueness of the solutions of an ODE.

2 Existence and Uniqueness Theorems

As we said in passing, a solution (if it exists) together with initial conditions of an ODE, ensures that the solution is unique. If there are no initial conditions we must regard the solution as a family of solutions, up to arbitrary constants. In this section we determine the existence and uniqueness theorems for ODE's defined on $F : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. When we continue our study of systems in later sections, these theorems and their assumptions have to be formulated to apply for higher dimensions. This poses no issue as the ideas presented here naturally generalise to higher dimensions. For instance, by introducing norms in place of absolute values[3].

Definition 3. *Lipschitz condition Let (X, d_x) and (Y, d_y) be two metric spaces. A function $f : X \to Y$ is called Lipschitz continuous if there exists a positive real constant, such that for all $x_1, x_2 \in X$, $d_y(f(x_1), f(x_2)) \leq K d_x(x_1, x_2)$. This is called a Lipschitz condition and K is called the Lipschitz constant.*

What is deduced from the Lipschitz condition is that if we map two points in X to Y the difference in the values does not get arbitrarily large. In the subsequent theorems we deal with the normal euclidean metric which we denote by absolute values.

Theorem 1. *(Picard and Lindelöf's Existence and Uniqueness theorem)*
Let $f(x, y)$ be a bounded continuous function of x, y in a region $\mathcal{R} \subset \mathbb{R}^2$ and let (x_0, y_0) be an interior point of $\mathcal{R}$. Furthermore, assume that for all pairs of points (x, s) and (x, t) in $\mathcal{R}$, $f(x, y)$ satisfies a Lipschitz condition:

$$|f(x, s) - f(x, t)| \leq K|s - t|.$$

Then there exists an open interval $I = (x_0 - h, x_0 + h)$ with h a positive number on which there is a unique continuous function $y(x)$ which solves the IVP

$$\frac{dy}{dx} = f(x, y) \text{ and } y(x_0) = y_0.$$

The rest of this section is devoted to the proof of this theorem, via a series of propositions. We begin with introducing the *Picard iterates*.

[3]For a treatment of Existence and Uniqueness for systems we refer to [7].

Since we are dealing with an IVP of the form:

$$\frac{dy}{dx} = f(x, y) \text{ and } y(x_0) = y_0.$$

We write a possible solution $y(x)$ from the fundamental theorem of calculus:

$$y(x) = y_0 + \int_{x_0}^{x} f(s, y(s))\ ds. \tag{4}$$

Conversely, if $y(x)$ is a continuous function which satisfies this integral equation, then $y(x)$ is a solution of the IVP. As a result, $y(x)$ is a solution of the IVP if and only if it is a continuous solution of (4). Since the function $y(x)$ occurs on the LHS and inside the integrand, we can think of the right hand side of equation (4) as an operator on functions. That is, it takes in a function $y(x)$ as an input, and produces another function as an output. If we call this operator Π, then we can write this idea down as follows:

$$\Pi[y(x)] = y_0 + \int_{x_0}^{x} f(s, y(s))\ ds.$$

The operator takes $y(x)$, forms the new function $f(s, y(s))$ in the "dummy" variable s and returns the integral from x_0 to x, producing a new function of x, to which it adds the constant y_0. It becomes clear then, that the input functions $y(x)$ which solve the IVP are left unchanged by the operator:

$$\Pi[y(x)] = y(x).$$

This motivates our definition of the *Picard iterates*.

Definition 4. *Given the aforementioned IVP, we define a sequence of functions recursively with n a positive integer, according to the Picard Iteration scheme:*

$$\begin{aligned} y_0(x) &= y_0 \\ y_{n+1}(x) &= y_0 + \int_{x_0}^{x} f(s, y_n(s))\ ds. \end{aligned}$$

Alternatively, in the operator notation: $y_{n+1}(x) = \Pi[y_n(x)]$.

Example. *Consider the first ODE introduced in this paper.* $\frac{dy}{dt} = y$ *with the initial condition* $y(0) = 1$. *Then the sequence of Picard iterates can be computed to,*

$$\begin{aligned} y_0(x) &= 1 \\ y_1(x) &= 1 + \int_0^x ds = 1 + x \\ y_2(x) &= 1 + \int_0^x 1 + s \; ds = 1 + x + \frac{x^2}{2} \\ &\vdots \\ y_n(x) &= 1 + x + \frac{x^2}{2} + \cdots + \frac{x^n}{n!}. \end{aligned}$$

Since $\sum_0^\infty \frac{x^n}{n!} = e^x$, *the sequence of Picard iterates converge to a solution* $y(x) = e^x$ *when* $\lim_{n\to\infty} y_n(x)$.

Turning to the proof of the Picard and Lindelöf theorem, we construct a suitable interval, on which the Picard iterates converge to the unique solution of the IVP, given the conditions of the theorem. Since we assume that the function $f(x, y)$ is bounded in the region $\mathcal{R}$, there exists a positive real number M for which we can write:

$$|f(x, y)| < M.$$

It was also assumed that (x_0, y_0) was an interior point of $\mathcal{R}$, as such we construct a rectangle Ω with the dimensions $|x - x_0| \leq h$ and $|y - y_0| \leq Mh$ for some positive number h, such that Ω lies within $\mathcal{R}$. It is worth to note that the interval I is one side of this rectangle without the endpoints.

Proposition 1. *For every* x *in* $[x_0 - h, x_0 + h]$ *with* h *real and positive, the function* $y_n(x)$ *remains in* Ω *for all* n.

Proof. Knowing that $(x, y_0) \in \Omega$, for all x in $[x_0 - h, x_0 + h]$, together with the first Picard iterate $y_0(x) = y_0$ being a constant function - a straight line parallel with the x-axis, we use this as our basis of induction. Then, assuming that $(x, y_{n-1}(x)) \in \Omega$, we conclude that:

$$\begin{aligned} y_n(x) &= y_0 + \int_{x_0}^x f(s, y_{n-1}(s)) \; ds \\ y_n(x) - y_0 &= \int_{x_0}^x f(s, y_{n-1}(s)) \; ds. \end{aligned}$$

If we impose the condition of the function $f(x, y)$ being bounded, the inequality;

$$|y_n(x) - y_0| = |\int_{x_0}^{x} f(s, y_{n-1}(s))\ ds| < M|x - x_0| \leq Mh,$$

ensures that $(x, y_n) \in \Omega$. The theorem holds for every $n \in \mathbb{N}$ by induction. □

Lemma 3. *The difference between two successive Picard iterations in the rectangle satisfy:*

$$|y_n(x) - y_{n-1}(x)| < \tfrac{MK^{n-1}}{n!}|x - x_0|^n.$$

Proof. From the previous proposition, the base case of our induction will be:

$$|y_1(x) - y_0| = |\int_{x_0}^{x} f(s, y_0(s))\ ds| < M|x - x_0|.$$

Assume the lemma holds up to $n - 1$, then we write:

$$y_n(x) - y_{n-1}(x) = \int_{x_0}^{x} f(s, y_{n-1}(s)) - f(s, y_{n-2}(s))\ ds.$$

We know from the Lipschitz condition that:

$$|\int_{x_0}^{x} f(s, y_{n-1}(s)) - f(s, y_{n-2}(s))\ ds| \leq K\int_{x_0}^{x} |y_{n-1}(s) - y_{n-2}(s)|\ ds,$$

but by the inductive hypothesis the RHS will satisfy the lemma and so:

$$|y_{n(x)} - y_{n-1}(x)| \leq \tfrac{MK^{n-1}}{(n-1)!}\int_{x_0}^{x} |s - x_0|^{n-1}\ ds = \tfrac{MK^{n-1}}{n!}|x - x_0|^n.$$

□

Proposition 2. *For each x in $[x_0 - h, x_0 + h]$ the sequence $y_n(x)$ converges uniformly.*

Proof. We have a sequence of functions each of which remain in the rectangle, furthermore the difference between each iteration satisfies lemma (3). We use the Weirstrass M-test to conclude uniform convergence on the interval. □

We have therefore proved, that on the interval $[x_0 - h, x_0 + h]$, the sequence of Picard iterates converge to a continuous function $y(x)$ on this interval. In addition, this function lies in Ω.

Proposition 3. *The limit of the sequence satisfies the initial value problem.*

Proof. We have proved uniform convergence of $y_n(x)$ on our interval of interest, furthermore the Lipschitz condition is satisfied in Ω:

$$|f(x, y_n(x)) - f(x, y(x))| \leq K|y_n(x) - y(x)|$$

Because of uniform convergence of the RHS, there exists an index m such that for all x in $[x_0 - h, x_0 + h]$,

$$|y_n(x) - y(x)| < \frac{\epsilon}{K}$$

when $n > m$. Therefore, when $n > m$:

$$|f(x, y_n(x)) - f(x, y(x))| < \epsilon.$$

Resultantly $f(x, y_n(x))$ converges uniformly to the continuous function $f(x, y(x))$ and we can conclude that:

$$\begin{aligned} y(x) = \lim_{n\to\infty} y_n(x) &= y_0 + \lim_{n\to\infty} \int_{x_0}^{x} f(s, y_{n-1}(s))\ ds \\ &= y_0 + \int_{x_0}^{x} \lim_{n\to\infty} f(s, y_{n-1}(s))\ ds \\ &= y_0 + \int_{x_0}^{x} f(s, y(s))\ ds. \end{aligned}$$

Since $f(x, y(x)$ is a continuous function on $[x_0 - h, x_0 + h]$, this together with the above expression implies that $\frac{dy}{dx} = f(x, y)$ on the open interval I in the theorem conditions. It also follows that:

$$y(x_0) = y_0 + \int_{x_0}^{x_0} f(s, y(s))\ ds = y_0.$$

Which is exactly the nature of the solution to the IVP. □

So far, this constitutes a proof of the existence of a solution to the IVP. This solution is also unique, as the last proposition guarantees.

Proposition 4. *The function $y(x)$ is the unique function satisfying the IVP.*

Proof. Assume that there exists another particular solution of the IVP, $g(x)$ such that:

$$\begin{aligned} \frac{dg}{dx} &= f(x, g(x)) \\ g(x_0) &= y_0 \\ g(x) &= y_0 + \int_{x_0}^{x} f(s, g(s))\ ds. \end{aligned}$$

Then we can express the difference between the two particular solutions as:

$$|y(x) - g(x)| \leq \int_{x_0}^{x} |y(s) - g(s)|\ ds.$$

Since the two functions stem from two respective rectangles Ω and Ω' both which are contained within the region $\mathcal{R}$. Since one side of the two respective rectangles is the compact set of points $[x_0 - h, x_0 + h]$, it is ensured that $|y(x) - g(x)|$ attains a maximum μ on this interval. Resultantly:

$$\begin{aligned} |y(x) - g(x)| &\leq \int_{x_0}^{x} f(s, y(s)) - f(s, g(s))\ ds \\ |y(x) - g(x)| &\leq K \int_{x_0}^{x} |y(s) - g(s))|\ ds \\ |y(x) - g(x)| &\leq K\mu|x - x_0|. \end{aligned}$$

If we continue integrating this expression, we can better the approximations of the upper bound of the difference in the two functions over the interval. Each new approximation yields a sequence:

$$K\mu|x - x_0|, \tfrac{K^2}{2!}\mu|x - x_0|^2, \tfrac{K^3}{3!}\mu|x - x_0|^3, \ldots, \tfrac{K^n}{n!}\mu|x - x_0|^n, \ldots$$

which tends to zero. It follows that

$$y(x) - g(x) = 0,\ y(x) = g(x).$$

□

With this understanding of the conditions of existence and uniqueness, one can speak of solutions to a ODE in a confident manner. As this does not necessarily mean that it is possible find the solution by means of elementary functions, we proceed to introduce the concept of a numerical method. In the next section we propose some general definitions of methods and clarify what it means for a method to be convergent, as we wish to assure that the numerical method "approximates" the true solution to desired accuracy. Since the existence of a unique solution can only be guaranteed on the interval I, all numerical methods must therefore be used within such an interval, as its the only region in which we posses a "hunting licence".

3 Taylor series and numerical methods.

One particular result which stems from the fruits of calculus is the *Taylor series*. The implications of the work of Taylor (although he was not alone) can arguably be compared to that of the revolution brought about by Newton, albeit not as immediate. In a most general sense, given a complicated problem satisfying a set of assumptions, the Taylor series allows us to consider instead a sequence of "easier" problems. Resultantly topics in analysis, perturbation theory and particularly numerical analysis makes frequent use of Taylor series.

In perturbation theory one considers for example a "difficult" problem such as finding the real roots of the function $f(x) = x^5 + x - 1$. Where the difficulty lies in the famous theorem of Abel. By introducing a perturbation ϵ to the problem and compute a perturbed solution $x(\epsilon)$ as a formal power series (by assumption), we can often reduce the difficult problem to matching the unperturbed solution as a power-series to the perturbed solution.

$$f(x) = x^5 + \epsilon x - 1 \tag{5}$$

$$x(\epsilon) = \sum_{n=0}^{\infty} a_n \epsilon^n. \tag{6}$$

When $\epsilon = 0$ the only real root is 1, and the first coefficient is therefore $a_0 = 1$. The problem we seek the solution for corresponds to $\epsilon = 1$, if this epsilon is within some radius of convergence of this series in a nontechnical sense, then we can approximate the solution to our desired accuracy by truncating the series. In many of these problems, the power series in question is often a Taylor series.
Viewing the problem of solving equation (5) from a numerical point of view we can consider, for example the *Newton-Raphson* method for finding roots of equations. By taking a single variable function $f(x) : \mathbb{R} \to \mathbb{R}$ with an educated guess of a root x_0, we can compute a sequence of approximations of the root $\{x_0, x_1, \ldots, x_n\}$ recursively with n a positive integer.

$$x_0 = Guess \tag{7}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad f'(x_n) \neq 0. \tag{8}$$

It is in our interest for this sequence of numbers to converge to the root of the equation. However the main point of address is the question of accuracy, i.e. under what assumptions can we quantify the difference between the true and numerical solution? and furthermore, will this allow the sequence of approximated roots to converge? In answer of this question,we begin with our guess of the solution x_0 and denote by $x^\star$ the unknown true solution. Then the perturbation $h = x^\star - x_0$ together with the assumption of the function being sufficiently smooth (C^2 to be precise) and $f'(x_0) \neq 0$ allows us to take the Taylor expansion of the first order, to derive:

$$\begin{aligned} f(x^\star) &= 0 \\ f(x^\star) &= f(x_0 + h) \\ f(x_0 + h) &\approx f(x_0) + hf'(x_0) \\ h &\approx -\frac{f(x_0)}{f'(x_0)} \\ x^\star &\approx x_0 - \frac{f(x_0)}{f'(x_0)}. \end{aligned}$$

This scheme is a special case of a general class of root-finding methods called *Householder methods.* In order to ensure that these methods work, one considers the true root $x^\star$ and computes a Taylor expansion in a neighbourhood of this root for some small deviation, which is our guess x_0. This allows us to compute a new, more accurate guess for the solution depending on the order of terms of the expansion. The subtle point here, is that by assuming $f(x) \in C^2$ we can use the properties of Taylor expansions to make a quantified statement about the difference between the true and numerical solution[4]. The property of interest comes from Taylor's theorem itself.

[4]A theorem of *Kantorovich*[12] provides a bound on the difference between the root $f(x^\star) = 0$ and approximated root x_n after n iterations, leading to a sequence which converges to zero in the limit as n gets large.

Theorem 2. *Suppose $f(x) : \mathbb{R} \to \mathbb{R}$ is a real valued function on $[a, b]$ which is n-times continuously differentiable such that $f^{(n)}(x)$ exists for every x in (a, b). If α and β are distinct points in the interval we define the Taylor polynomial as:*

$$P(x) = \sum_{k=0}^{n-1} \frac{f^{(k)}(\alpha)}{k!}(x - \alpha)^k.$$

Then there exists a point χ between α and β such that

$$f(\beta) = P(\beta) + \frac{f^{(n)}(\chi)}{n!}(\beta - \alpha)^n.$$

The theorem tells us that a sufficiently smooth function can be approximated by a polynomial of degree $n-1$ and that we can estimate the error if we know the bounds of $|f^{(n)}(x)|$. In later sections Taylor polynomials (or expansions) of a single variable function are computed with respect to vector valued functions (as will be clear from context). As for computing Taylor expansions of vector valued functions the formula becomes almost indistinguishable from the single variable case, provided we use the derivative defined in section 6. We refer to [2] for details. Before we get too exited and blindly apply the theorem, we must consider some shortcomings.
Example. *Consider the function:*

$$f(x) = \begin{cases} e^{-\frac{1}{x^2}} & \text{for } x \neq 0 \\ 0 \quad \text{for } x=0 \end{cases}$$

It is a C^∞ function and all the derivatives evaluated at $x = 0$ are zero, and thus the Taylor series converges to a function which is identically zero, but only represents the function $f(x)$ at a single point. The repercussion is that we must take great care of which neighbourhoods we choose to take our expansion in.

3.1 One-step methods

Having acquainted ourselves with the Taylor series and importantly the error term, we return to the topic of solving ODEs. In general, a numerical method to solve an IVP is an approximation of the exact flow map of the differential equation, at chosen points of evaluation. As such, there is always an element of error in the numerically generated solution. For the method is to be useful, we must be able to control this error, at least on short enough time steps (as we explain in this section). The reason as to why we introduced the Taylor series is that it gives us this control for Runge-Kutta methods. To make this precise we need to introduce the concepts of *one step methods*, *local and global errors*, *convergence* and *order*. For this discussion

we consider the IVP with:

$$f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$y' = f(t, y) \quad y(t_0) = y_0.$$

Noting that as we proceed to solve systems, the ideas presented here should apply component-wise, as we will proceed to do in later sections.

Definition 5. *In order to compute a numerical solution to the IVP, we consider a subdivision of the time interval $[t_0, T]$ we wish to solve over. By taking $\frac{T-t_0}{n} = h$ we choose the points of evaluation to be $t_i = t_{i-1} + h$ for $i = 1 \dots n$.*
Notation. When we write the numerical solution at the point t_n we will always use lower subscripts, y_n. For the analytical solution we will use $y(t_n)$.

Definition 6. *A one-step method is a numerical method of the IVP, which successively computes a new point of approximation from a previous point of approximation.*

$$y_{n+1} = y_n + h\phi(t_n, y_n, h).$$

Where ϕ is called the increment, or step function.

In the next section we will explicitly state the form of this function, but in the general case we assume that this function is continuous in all its arguments and is sufficiently smooth, allowing us to compute its Taylor approximation.

Definition 7. *Local discretisation error τ_i at the point t_i is defined as the difference between the analytical and numerical solution at this point. Assuming that both were exact prior to the beginning of the step.*

$$y(t_i) - y_i = \tau_i.$$

Remark. In the litterature, the local discretisation error is sometimes referred to as truncation error. An equivalent formulation of the local error is sometimes expressed by taking:

$$\tau_{i+1} = y(t_i + h) - y_i - h\phi(t_i, y(t_i), h)$$
$$\frac{\tau_{i+1}}{h} = \frac{y(t_i + h) - y_i}{h} - \phi(t_i, y(t_i), h).$$

Since its in our interest for the local error to vanish as we make $h \to 0$, we see that:

$$\lim_{h \to 0} \frac{\tau_{i+1}}{h} = y'(t_i) - \phi(t_i, y(t_i), 0). \tag{9}$$

Therefore the local error vanishes if and only if:

$$\begin{aligned} y'(t_i) &= \phi(t_i, y(t_i), 0) \\ f(t_i, y(t_i)) &= \phi(t_i, y(t_i), 0). \end{aligned}$$

This leads to the following definition:

Definition 8. *The one step numerical method of the IVP is said to be consistent with the differential equation if the local error can be made arbitrarily small. That is, for any $\epsilon > 0$ there exists a positive $h(\epsilon)$ such that $|\tau_i| < \epsilon$ for $0 < h < h(\epsilon)$, and any two points $(t_i, y(t_i)), (t_{i+1}, y(t_{i+1}))$.*

By virtue of Taylor's theorem, if the true solution $y(t)$ is sufficiently smooth, we express the solution as a polynomial with the error term being some constant times a power of h which is the degree. We denote this term by $O(h^p)$.[5]

$$\begin{aligned} y(t_i + h) &= y(t_i) + y'(t_i)h + \cdots + O(h^{p+1}) \\ y_{i+1} &= y(t_i) + \phi(t_i, y(t_i), h). \end{aligned}$$

If we can take a Taylor expansion of the step function and match these to the terms of $y(t_i + h)$ up to the term with h^p, the local error reduces to:

$$y(t_i + h) - y_{i+1} = O(h^{p+1}).$$

Definition 9. *The numerical method is said to have order of accuracy p, if*

$$|\tau_i| \leq Kh^{p+1} \quad p \geq 1.$$

A mere knowledge of the local error is not sufficient for our purposes, as it does not take into account that with each step of computation, we have already made an error in the approximation of the points we use for the next step. As a result the error is carried along the computation, leading to an error at the final step which is different from the local error.

Definition 10. *The global error e_n is the difference between the true and numerical solution after n points of evaluation.*

$$y(t_n) - y_n = e_n.$$

[5]For details on big O notation we refer to [11] page 81.

For the concept of convergence of a numerical method, there is no one accepted definition, and different authors use different ways to ensure convergence. It is however necessary for the global error to disappear as we take smaller step sizes. It turns out that under certain assumptions we can put an upper bound on the global error of a one step method in terms of the local error, as the following theorem shows.

Theorem 3. *Given the one step method with a step function $\phi(t, y, h)$ which is continuous with respect to its arguments, and satisfies a Lipschitz condition with respect to y such that the Lipschitz constant is L. Then the global error e_n after n computations satisfies:*

$$|e_n| \leq e^{L(t_n - t_0)}|e_0| + \frac{e^{L(t_n - t_0)} - 1}{hL}\tau.$$

Where $\tau = \max_{i=0\ldots n} |\tau_i|$.

Proof. Taking the definition of the one step method,

$$y_{n+1} = y_n + h\phi(t_n, y_n, h)$$

and subtracting this from the local error, then by algebraic manipulation we get:

$$\begin{aligned} e_{n+1} &= \tau_{n+1} + e_n + h(\phi(t_n, y(t_n), h) - \phi(t_n, y_n, h)) \\ |e_{n+1}| &\leq |\tau_{n+1}| + |e_n| + h|\phi(t_n, y(t_n), h) - \phi(t_n, y_n, h)|. \end{aligned}$$

By the Lipschitz condition this reduces to:

$$\begin{aligned} |e_{n+1}| &\leq |\tau_{n+1}| + |e_n| + hL|e_n| \\ |e_{n+1}| &\leq |\tau_{n+1}| + (1 + hL)|e_n|. \end{aligned}$$

We can then replace the local error by $\tau = \max_{i=0\ldots n} |\tau_i|$, and recursively compute,

$$\begin{aligned} |e_1| &\leq (1 + hL)|e_0| + \tau \\ &\vdots \\ |e_n| &\leq (1 + hL)^n|e_0| + \frac{[(1 + hL)^n - 1)]}{hL}\tau. \end{aligned}$$

The last inequality relies on the fact that $\frac{z^n - 1}{z - 1} = z^{n-1} + z^{n-2} + \cdots + z + 1$. From definition 5 and the fact that $1 + hL \leq e^{hL}$, it follows that:

$$|e_n| \leq e^{L(T - t_0)}|e_0| + \frac{[e^{L(T - t_0)} - 1)]}{hL}\tau.$$

□

We have implicitly assumed that the analytical solution exists over the interval in question. If we wish to make this precise we can assume the conditions of the *Picard and Lindelöf* theorem, that is we are working within some rectangle with Ω as before.The essence of Theorem 3 suggests that if the local error approaches zero as $h \to 0$ then the global error "converges to zero" assuming that $e_0 \to 0$ when h approaches zero. Indeed, if the method is consistent, then we can be sure this is the case, as the following theorem shows.

Theorem 4. *If the analytical and numerical solutions of the IVP lie within some Ω as with the Picard and Lindelöf existence and uniqueness theorem. Together with the step function satisfying the Lipschitz and consistency conditions,and uniformly continuous. Then if the approximations y_n that are generated for $t_n = t_0 + nh$ are made with successively smaller values of h than h_0, the numerical solution converges to the analytical solution of the IVP in the sense that:*

$$|y(t_n) - y_n| \to 0 \text{ as } h \to 0 \text{ and } t_n \to t \in [t_0, T].$$

Proof. Taking $h = \frac{T-t_0}{N}$ with the positive integer N sufficiently large as to ensure that $h \leq h_0$. By virtue of the initial conditions $e_0 = 0$, the theorem of the global error for $n = 1, \ldots, N$ reduces to:

$$|e_n| \leq \frac{e^{L(t_n - t_0)} - 1}{hL}\tau.$$

Where $\tau = \max_{i=0,\ldots,n-1} |\tau_i|$. Since we are assuming consistency, we rewrite the local error for $n = 0, 1, \ldots N-1$ as:

$$\frac{\tau_{n+1}}{h} = [\frac{y(t_n + h) - y(t_n)}{h} - f(t_n, y(t_n))] + [\phi(t_n, y(t_n), 0) - \phi(t_n, y(t_n), h)].$$

Applying the mean value theorem, there exists an $\alpha \in [t_{n+1}, t_n]$ such that the left bracket reduces to $y'(\alpha) - y'(t_n)$. The consistency condition implies that $y'(t)$ is uniformly continuous on $[t_0, T]$ since $\phi(t, y(t), 0)$ is. Therefore, for every $\epsilon > 0$ there exists a $h_1(\epsilon)$ making:

$$|y'(\alpha) - y'(t_n)| \leq \tfrac{\epsilon}{2} \text{ for } h < h_1(\epsilon).$$

Similarly for ϕ, which is uniformly continuous there exists a $h_2(\epsilon)$ such that:

$$|\phi(t_i, y(t_i), 0) - \phi(t_i, y(t_i), h)| \leq \tfrac{\epsilon}{2} \text{ for } h < h_2(\epsilon).$$

Proceeding as in any classic epsilon-delta type proof we let $h(\epsilon) = \min(h_1(\epsilon), h_2(\epsilon))$ and conclude that,

$$\frac{|\tau_n|}{h} \leq \epsilon$$

for $h < h(\epsilon)$. Therefore $|y(t_n) - y_n| \to 0$ as $h \to 0$.
Writing:

$$|y(t) - y_n| \leq |y(t) - y(t_n)| + |y(t_n) - y_n|,$$

it becomes clear that since $t_n \to t \in [t_0, T]$ when $h \to 0$, by uniform continuity of y the numerical solution converges to the solution of the IVP. □

We have shown that a one step method of solution to the IVP which satisfies consistency and Lipschitz conditions and uniform continuity will become arbitrarily close to the solution of the IVP as we reduce the step size h. For the remainder of this essay we assume that the ODEs we examine satisfy the conditions of existence and uniqueness. That is, we consider bounded continuously differentiable functions which satisfy a Lipschitz condition. Furthermore we assume that both the differential equation and its solution are sufficiently smooth to take the Taylor series of desired order. Before we turn to the general Runge-Kutta methods, we introduce the *Euler method* as a special case of these as to illustrate the ideas of this section.

4 The Euler Method

The numerical method to solve the IVP devised by Euler, enjoys great historical importance as an early method of solving ODEs[6]. It is commonly a "first choice" method when approaching a new problem, as it is relatively simple to compute. If the method fails its task, then one might proceed to compute the solution with a more elaborate method. But for many purposes, the Euler method provides a great deal of insight.
Given a first order ODE or a system of first order ODEs with a given initial condition $t = t_0$ we know the value of the function at this point, i.e. the slope. Given any $t \geq t_0$ we can approximate the function at t by the method of linear interpolation. We estimate $y'(t)$ by $f(t, y(t)) \approx f(t_0, y(t_0))$ for $t \in [t_0, t_0 + h]$ where h is a positive value. If we integrate the differential equation from t_0 to $t_0 + h$ and use the fundamental theorem of calculus, one gets

$$y(t_0 + h) - y(t_0) = \int_{t_0}^{t_0+h} f(t, y(t))dt.$$

If we approximate the integral with a rectangle (that is, assume $f(t, y(t)) \approx f(t_0, y(t_0)))$

$$\int_{t_0}^{t_0+h} f(t, y(t))dt \approx hf(t_0, y(t_0)).$$

Combining the two expressions and using the old approximations one gets a scheme:

$$\begin{cases} y_1 = y_0 + hf(t_0, y_0) \\ \vdots \\ y_n = y_{n-1} + hf(t_{n-1}, y_{n-1}). \end{cases}$$

This is the Euler method, and since $y_n = y_{n-1} + hf(t_{n-1}, y_{n-1})$, this should remind us of the discussion of the step function in the previous section. By design, this function is the ODE itself, and since we assume that this function is Lipschitz as to ensure existence and uniqueness, the step function satisfies this as well. We will use these properties to prove convergence in due course. For now we take this for granted and acquaint ourselves with the computational aspect of the method. We begin with a definition.

Definition 11. *The Euler polygon is a linear operator $P_n(y_0, y_1, \ldots, y_n)$ that from the defined recursion yields the piecewise linear interpolating function $\tilde{y}$, which is linear in each interval $[t_n, t_{n+1}]$ and $\tilde{y}(t_n) = y_n$.*

[6]The method was originally introduced in Euler's work of 1768 [5].

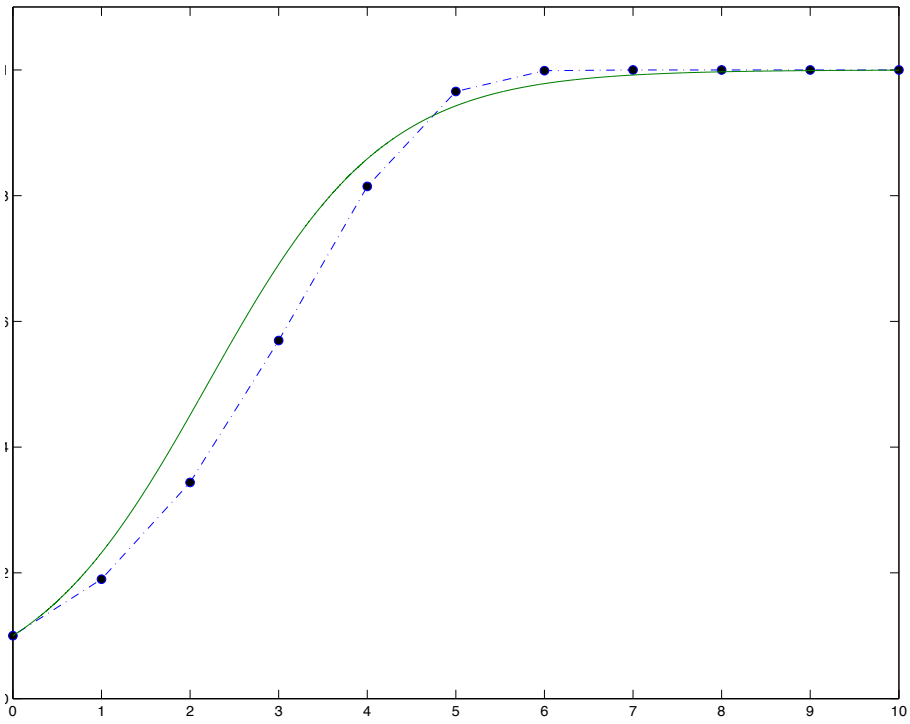


Figure 1: The Euler polygon and analytical solution for the logistic equation on $t = [0, 10]$, settling down to the value $y = 1$.

Example. *The Logistic Equation*

$$y' = y(1 - y).$$

This ODE is a special case of the Logistic Equation It is non-linear, with a wide range of uses in applied mathematics, especially predator-prey models in biology. If the reader is familiar with the concept of carrying capacity of an ecological system, the shape of the solution is the so called sigmoid curve. Furthermore, in the discrete case it can be viewed as a recurrence relation that whilst simple in appearance, gives rise to chaotic behaviour.

With $t = 0$ *and* $y(0) = \frac{1}{10}$, *we pick the step size* $h = 1$.

$$y_1 = y_0 + f(t_0, y_0)$$

$$y_1 = \tfrac{1}{10} + f(0, \tfrac{1}{10}) = \tfrac{1}{10} + \tfrac{9}{100} = \tfrac{19}{100}.$$

In *Figure 1*, the Euler polygon $\tilde{y}$ is plotted together with the exact solution (which will not be discussed here). The *Matlab code* was generated from the function in the appendix.

Example. *Consider the system defined on* $\Omega = \mathbb{R} \times \mathbb{R}^2$

$$y_1' = \frac{3}{2} - \frac{1}{10}y_1 + \frac{3}{40}y_2 \tag{10}$$

$$y_2' = 3 + \frac{1}{10}y_1 - \frac{1}{5}y_2 \tag{11}$$

with the initial conditions $y_1(0) = 25,\ \ y_2(0) = 15$. *Let* Y' *and* $Y(0)$ *denote:*

$$Y' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} \qquad Y(0) = \begin{bmatrix} 25 \\ 15 \end{bmatrix}$$

$$Y' = \begin{bmatrix} \frac{3}{2} - \frac{1}{10}y_1 + \frac{3}{40}y_2 \\ 3 + \frac{1}{10}y_1 - \frac{1}{5}y_2 \end{bmatrix}.$$

We set $h = 0.1$ *and apply the Euler method to get;*

$$Y_1 = \begin{bmatrix} 25 \\ 15 \end{bmatrix} + (0.1)\begin{bmatrix} \frac{3}{2} - \frac{1}{10}(25) + (\frac{3}{40})(15) \\ 3 + \frac{1}{10}(25) - \frac{1}{5}(15) \end{bmatrix} = \begin{bmatrix} 25.0125 \\ 15.25 \end{bmatrix} \tag{12}$$

$$Y_2 = \begin{bmatrix} 25.0125 \\ 15.25 \end{bmatrix} + (0.1)\begin{bmatrix} \frac{3}{2} - \frac{1}{10}(25.0125) + \frac{3}{40}(15.25) \\ 3 + \frac{1}{10}(25.0125) - \frac{1}{5}(15.25) \end{bmatrix} = \begin{bmatrix} 25.03 \\ 15.50 \end{bmatrix}. \tag{13}$$

Having come to grasp the computational procedure of the Euler method, we turn to convergence, and proceed as outlined in the previous section. By theorem 4, it is sufficient for us to show consistency, since the step function by construction is Lipschitz and thus uniformly continuous.

Proposition 5. *The Euler method converges to the true solution if* f *is Lipschitz and* C^2.

For the proof of this proposition, we consider a subdivision of the interval of interest into integral portions $\frac{T-t_0}{n} = h$ as to yield the set of points of evaluation of the method

$$\{t_0, t_1 \dots t_n\} \subset \mathbb{R}.$$

Where $t_i = t_{i-1} + h$. Since the ODE is C^2 we take the Taylor expansion of the analytical solution:

$$y(t_i + h) = y(t_i) + hf(t_i, y(t_i)) + \frac{f'(\alpha)}{2}h^2, \tag{14}$$

for some $\alpha \in [t_i, t_{i+1}]$. Recalling the definition of the local error,

$$\begin{aligned} \tau_{i+1} &= y(t_{i+1}) - y_{i+1} \\ y(t_i + h) - y_{i+1} &= \frac{f'(\alpha)}{2}h^2. \end{aligned}$$

In passing, we should mention that this is the reason as to why some refer to the local error as truncation error. We could finish here, and refer to the theorems of the last section, but for clarity we continue by bounding the global error in terms of the local error.

$$\begin{aligned} |y(t_{i+1}) - y_{i+1}| &= |y(t_i) - y_i + hf(t_i, y(t_i)) - hf(t_i, y_i) + \tau_i| \\ |y(t_{i+1}) - y_{i+1}| &\leq |y(t_i) - y_i| + h|f(t_i, y(t_i)) - f(t_i, y_i)| + |\tau_i| \\ |y(t_{i+1}) - y_{i+1}| &\leq |e_i| + hL|e_i| + |\tau_i| \\ |e_{i+1}| &\leq |e_i|(1 + hL) + |\tau_i|. \end{aligned}$$

Picking $K = \max_{t\in[t_0,T]} |f'(t, y(t))|$ such that $|\tau| = \frac{K}{2}h^2$ we write recursively:

$$\begin{aligned} |e_{n+1}| &\leq |e_n|(1 + hL) + |\tau| \\ &\leq |e_{n-1}|(1 + hL)^2 + (1 + (1 + hL)|\tau| \\ &\vdots \\ &\leq |e_0|(1 + hL)^{n+1} + \frac{(1 + hL)^{n+1} - 1}{(1 + hL) - 1)}|\tau|. \end{aligned}$$

Since we know that at the initial conditions, the true and numerical solutions coincide, $e_0 = 0$ then we can examine e_n, noting that $hn = T - t_0$.

$$\begin{aligned} |e_n| &\leq \frac{(1 + hL)^n - 1}{hL}\tau \\ |e_n| &\leq \frac{e^{Lhn} - 1}{hL}\tau \\ |e_n| &\leq \frac{e^{L(T-t_0)} - 1}{2L}Kh. \end{aligned}$$

The global error after n steps of the Euler method is bounded by a real constant times the step size. From the discussion of the last section, the Euler method is consistent with order of accuracy 1, and we have provided an explicit bound on the global error, and so the method is said to converge. As we turn our attention to general Runge Kutta methods, ensuring convergence in the manner presented here would prove tedious and unnecessary, instead we prove consistency. This is done through the *order conditions.*

> "Det var i andra dimensionen så jag tog Euler metoden ... tills en röst i mitt huvud sa: Du ska prova Runge-Kuttan,"
>
> From the song Runge-Kuttan, lyrics and music by Mathias Lundgren.

5 Runge-Kutta Methods

As we saw in the section on the Euler method, the idea was to take the initial value problem and propagate the solution forward by a sequence of small time steps. With each step the slope of the numerical solution is computed from the current position, and as such a one-step method. The numerically computed points of solution will therefore make an error at the first step, where the next derivative is evaluated and so the error accumulates as the method proceeds through the sequence of time steps. Theoretically, we are not too worried about this error, since we did indeed prove that this method converges to the true solution when the step size h goes to zero. Practically however, the aim of applying a numerical method to an ODE is to compute the numerical solution quickly and accurately. If we want an accurate Euler solution to an ODE we have to compromise between increasing accuracy (smaller h) and calculating more time-steps.

Before the advent of electronic computers, one can imagine the distraught faces of the "computers" as they were told "we need better accuracy - halve the step size". It was C.Runge who tackled this problem in his 1895 paper[13] by introducing the idea of multiple evaluations of the derivative at each time step. Together with Heun (1900), new methods were developed which applied the Euler method with one or two additionally introduced steps. It was Kutta (1901)[8] who provided the scheme of what we today call a Runge-Kutta method, with its characteristic feature of evaluating the function a number of times at each step and using a weighted linear combination of the evaluations to provide increased accuracy and efficiency.

A numerical method is a Runge-Kutta method of its scheme satisfies the definition set forth by Kutta. There are many such methods and because of the advantages of some of them, a few are particularly famous. The most famous example is that of the classical Runge Kutta method or RK4, but also includes the now familiar Euler method.

Let $\frac{dy}{dt} = f(t, y(t))$ be our ODE in explicit form, defined on the open set $\Omega \subset \mathbb{R} \times \mathbb{R}^n$ with initial conditions $y(t_0) = y_0$, where $y(t) = (y_1(t), y_2(t), ...y_n(t)) \in \mathbb{R}^n$.

We are interested in a numerical approximation of the continuously differentiable solution $y(t)$ of the IVP over a given interval for $t_0 \leq t \leq T$. We form a partition of the interval into segments of equal length h which we call step size, $\frac{T-t_0}{n} = h$.

Definition 12. *The family of explicit Runge Kutta (RK) methods of s stages is given by the scheme:*

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i.$$

Where k_i are the stages of the method

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + c_2 h, y_n + h a_{21} k_1(t_n, y_n)) \\ k_3 &= f(t_n + c_3 h, y_n + h(a_{31} k_1(t_n, y_n) + a_{32} k_2(t_n, y_n))) \\ &\vdots \\ k_s &= f(t_n + c_s h, y_n + h \sum_{j=1}^{s-1} a_{sj} k_j). \end{aligned}$$

The a_{ij}, b_i, c_i are real coefficients specifying the constructed method.

This scheme specifies the family of Runge Kutta methods. If we choose $s \in \mathbb{N}$ we get an s-stage Runge Kutta method. Each step of the method consists of a y_n added to a linear combination of the s stages of the computation. This linear combination uses the coefficients b_i, c_i and a_{ij} together with the step size to provide the new point y_{n+1} in $\mathbb{R}^n$. As such, we consider the RK methods one step methods, since no additional knowledge from the current position is required. The step function takes the form of:

$$\phi(t, y, h) = \sum_{i=1}^{s} b_i k_i.$$

For the method to be consistent, we take h to be zero. It becomes immediately clear that we must impose the condition that $\sum_i^s b_i = 1$. It turns out, that in order to provide a specific Runge-Kutta method the coefficients determine its exact nature. Not all choices of these provide a method, indeed they obey a set of constraints stemming from our desire to correspond the method to the Taylor series of the true solution. To illustrate this point we consider the simplest Runge-Kutta method, the Euler method.

We pick a step size h and compute a 1-stage Runge Kutta method for the IVP.

$$y_{n+1} = y_n + h\sum_{i=1}^{1} c_i k_i = y_n + hb_1 k_1 \tag{15}$$

$$y_1 = y_0 + hb_1 f(t_0, y(t_0)). \tag{16}$$

By the Taylor expansion of the true solution we get.

$$y(t_0 + h) = y(t_0) + hy'(t_0) + O(h^2) = y_0 + hf(t_0, y(t_0)) + O(h^2).$$

If $b_1 = 1$ the method is identical as that of the Euler method. This should be of little surprise as we begin at an initial point and use only one computation of the derivative (slope) at this point with a weight b_1. Since there is only one step, we have a trivial linear combination. We have already proved that this method converges, its global error is in proportion to h and so we call it a first order RK method, as the next definition clarifies.

Definition 13. *A Runge-Kutta method has order p if:*

$$|y(t_0 + h) - y_1| \leq Kh^{p+1} \text{ for a positive real constant } K$$

This occurs when the Taylor series for the exact solution of $y(t_0 + h)$ and y_1 coincide to the term h^p. By comparison of the method to the Taylor expansions of the true solution we specify values for our coefficients and also learn how the local error behaves. A RK method has order p if the local truncation error behaves like $O(h^{p+1})$. Indeed, we are assuming that both the exact and the numerical solution possesses a Taylor expansion in our desired neighbourhood in the first place. For the remainder of this paper, we take this assumption for granted.

Definition 14. *The order conditions are the equations that the coefficients of the RK method have to satisfy to be of order p.*

Therefore, proving the order conditions, ensures that the RK method is convergent. Since this implies that the method is consistent of some order, and by construction, satisfies the assumptions of the theorems of section 3. This will be done in the subsequent sections, and herby proceed to familiarise the reader with some RK methods of low order.

0				
c_2	a_{21}			
c_3	a_{31}	a_{32}		
$\vdots$				
c_s	a_{s1}	a_{s2}	$\dots$	$a_{s,s-1}$
	b_1	b_2	$\dots$	b_s

Table 1: Butcher Tableau

If we want to compute an s-stage RK method, it is often common to place the coefficients of the method in a Butcher tableau: Where the c column determines the times where the stage is computed, and the b row is the weights of the linear combinations. Since the methods we examine are explicit the tableau is strictly lower triangular, that is: $a_{ij} = 0$ unless $i > j$. It is also custom to impose that c_i satisfies:

$$c_i = \sum_j a_{ij}.$$

If this condition is satisfied, the derivation of the order conditions may be considered for the autonomous case only[7]. To help the reader we compute a Runge-Kutta 2-stage method.

$$k_1 = f(t_0, y_0) \tag{17}$$
$$k_2 = f(t_0 + c_2 h, y_0 + h a_{21} k_1(t_0, y_0)) \tag{18}$$
$$y_1 = y_0 + h \sum_{i=1}^{2} b_i k_i = y_0 + h b_1 k_1 + h b_2 k_2. \tag{19}$$

We assume that the exact solution has a Taylor expansion, given by:

$$y(t_0 + h) = y_0 + h f(t_0, y_0) + \tfrac{1}{2} \tfrac{df}{dt}(t_0, y_0) h^2 + O(h^3).$$

We rewrite this in terms of the total derivative rule to get:

$$y_0 + h f(t_0, y_0) + \tfrac{1}{2} [\tfrac{\partial f}{\partial t}(t_0, y_0) + \tfrac{\partial f}{\partial y}(t_0, y_0) f(t_0, y_0)] h^2 + O(h^3).$$

It is our task to equate $y_1 = y_0 + h b_1 k_1 + h b_2 k_2$ with the Taylor expansion of the exact solution to arrive at the order conditions.

[7]We will take this condition for granted, but the interested reader is referred to page 143 of Harier. As an historical note, Kutta simply assumed this condition in his work.

This is done by calculating a Taylor series for k_2.

$$\begin{aligned} k_2 &= f(t_0 + c_2 h, y_0 + h a_{21} k_1(t_0, y_0)) && (20) \\ k_2 &= f(t_0, y_0) + c_2 h \frac{\partial f}{\partial t}(t_0, y_0) + a_{21} h \frac{\partial f}{\partial y}(t_0, y_0) f(t_0, y_0) + O(h^3). && (21) \end{aligned}$$

Thus the numerical solution $y_1 = y_0 + hc_1k_1 + hc_2k_2$ can be written on the form;

$$y_1 = y_0 + hb_1 f_0 + hb_2[f(t_0, y_0) + c_2 h \frac{\partial f}{\partial t}(t_0, y_0) + a_{21} h \frac{\partial f}{\partial y}(t_0, y_0) f(t_0, y_0)] + O(h^3). \quad (22)$$

We take the difference between the numerical and exact solutions to specify the coefficients.
$[y_0 + hf(t_0, y(t_0)) + \frac{1}{2}\frac{df}{dt}(t_0, y(t_0))h^2]$
$-[y_0 + hb_1 f(t_0, y(t_0)) + hb_2[f(t_0, y_0) + c_2 h \frac{\partial f}{\partial t}(t_0, y_0) + a_{21} h \frac{\partial f}{\partial y}(t_0, y_0) f(t_0, y_0)]] = 0.$

This equation is satisfied if:

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 a_{21} &= \frac{1}{2} \\ c_2 b_2 &= \frac{1}{2}. \end{aligned}$$

Notice that we get a set of three algebraic polynomial equations with four unknowns, these are the order conditions. We thereby proceed to solve these by parametrizing and present the second order method in the Butcher tableau. If we choose $b_1 = 0$ and $b_2 = 1$, the tableau looks like:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}.$$

Taking the coefficients we write the numerical method as:

$$y_1 = y_0 + hf(t_0 + \tfrac{h}{2}, y_0 + \tfrac{h}{2} f(t_0, y_0)).$$

This is the midpoint method, one of the first RK methods to be developed by Runge in 1895. The first RK methods that were constructed had geometrical interpretations stemming from quadrature problems. In this case by considering Gauss midpoint formula and extending it to the IVP. This is not the only second order method as we can parametrise by $b_1 = \frac{1}{4}$ and proceed to solve the equations.

$$\begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

$$y_1 = y_0 + h\tfrac{1}{4}f(t_0, y_0) + h\tfrac{3}{4}f(t_0 + \tfrac{2}{3}, y_0 + \tfrac{2h}{3}f(t_0, y_0)).$$

We recall from definition that a RK method is of order p, if the taylor series coincide up to and including p. The computed methods are of order 2, since we can match up to the second derivatives. By adding additional stages of the method, we can match higher derivatives of the true solution. In the case of three stages, we can construct order 3 methods, and proceeding to 4 stages we can create an order 4 method. Arguably, one of the most popular RK methods is sometimes referred to as "the" Runge Kutta method. It has become so popular that it is an essential part of every scientist or engineer's bag of tricks (or desktop application for that matter). For many problems the method is both fast and accurate for adequate step sizes (being of order 4), and when implemented in computer code requires little or no understanding of the computation involved. By taking a method of 4 stages:

$$y_1 = y_0 + h(b_1k_1 + b_2k_2 + b_3k_3 + b_4k_4).$$

To provide a fourth order method, the coefficients must match up to the 4th order term of the taylor expansion of the true solution. This procedure is an exercise in the rules of differential calculus, and the computation becomes very tedious. Indeed Kutta himself did provide the order conditions for the fourth order, but without any comments. "The" Runge-Kutta method (RK4) results from a particular parametrisation which results in the tableau.

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

Example. *In the appendix we showcase the methods presented in this section by graphing the numerical solutions to the IVP corresponding to Hooke's law.*

As we allow for higher and higher orders of our RK method, at each step we must compute successively higher and more complicated expressions for the derivatives. These will be increasingly difficult to match to the Taylor series as to determine the order conditions. Initially it seems as the order of the method is the number of stages of the method, this is indeed true for orders up to 4, but not in general. The number of stages required to construct a method of any order is still an open problem.

Order	1	2	3	4	5	6	7	8	9
Min number of stages	1	2	3	4	6	7	9	11	11

Table 2: Orders versus Stages

This was shown by the work of Butcher, who in addition provided a new method in terms of trees to calculate the derivatives of the true and numerical solutions of the IVP. The relation between rooted trees and taking derivatives of compositions of functions was first explored by A.Caley in 1857[3]. In turn we calculate the Taylor series for both the true and numerical solutions using Butcher's abstraction and arrive at the order conditions without the term by term matching used in this section. To bridge the gap between the derivatives and the trees, tensor notation provides a clear and convenient correspondence. The next section introduces this notation before we turn to derive the order conditions in terms of rooted trees.

6 Derivation of the order conditions

We have already discussed how the order conditions for the RK methods arise, and we saw how the derivatives quickly increase in their complexity as we strive for RK methods of higher order. In this section we combine the theory of trees with the tensor notation to present a graphical representation of the derivatives. If we also use the set of labelled trees, together with some functions on this set, we derive the order conditions without resorting to extensive computations and tedious bookkeeping. Due to the difficulties of making a short and concise treatise on Tensors, the next few pages are by no means self contained and many proofs are omitted. The reader not familiar with the ideas presented is referred to any good text on multilinear algebra and/or tensors, such as [10].

6.1 Multilinear maps, Tensors and Einstein notation

Definition 15. *A collection of real vector spaces $V_1, V_2, \ldots, V_k$ and W, with a map $T : V_1 \times V_2 \times \cdots \times V_k \to W$, is called multilinear if it is separately linear on each of its variables. i.e. For each v_i and v_i' in V_i and scalars α, β in $\mathbb{R}$ the map satisfies:*

$$\begin{aligned} T(v_1, \ldots, v_{i-1}, \alpha v_i + \beta v_i', \ldots v_k) &= \alpha T(v_1, \ldots, v_{i-1}, v_i, \ldots v_k) \\ &+ \beta T(v_1, \ldots, v_{i-1}, v_i', \ldots v_k). \end{aligned}$$

Example. *Consider the map $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, defined by the usual "dot" product. Such that for $(v, w) \in \mathbb{R}^n \times \mathbb{R}^n$,*

$$v \cdot w = \sum_{i=1}^{n} v_i w_i.$$

This defines a multilinear map (which we call bilinear by virtue of the single cartesian product), since:

$$\begin{aligned} (\alpha v + \beta v') \cdot w &= \alpha \sum_{i=1}^{n} v_i w_i + \beta \sum_{i=1}^{n} v_i' w_i \\ &= \alpha (v \cdot w) + \beta (v' \cdot w) \end{aligned}$$

For scalars α and β, and the same argument holds for w.

Definition 16. *For an n-dimensional vector space V and its dual $V^\star$, denote by $(V)^l$ the l fold cartesian product $V_1 \times V_2 \times \cdots \times V_l$. Then, for the two non negative integers k, l we define the vector space V_l^k to consist of all multilinear maps:*

$$T : (V)^l \times (V^\star)^k \to \mathbb{R}.$$

These are called tensors of type (k, l) over V. We often refer to these tensors according to their rank: $rank(T) = k + l$.

Resultantly, a tensor $T \in V_l^k$ has rank $k + l$ and acts on a set of l vectors and a set of k dual vectors. Since we posses a basis $\{e_i\}_{i=1}^n$ of V, there is a corresponding induced basis of $V^\star$, $\{f^j\}_{j=1}^n$. Together they determine a basis for V_l^k. We claim that this is a vector space since, we can use the component-wise addition and scalar multiplication inherited from multilinear maps. As such we consider the tensor T as a multidimensional array of scalars, where the rank is the number of indexes, and using the basis for the vector space we can evaluate the tensor on a set of vectors and dual vectors.

To any $T \in V_l^k$, we associate a set of scalars in $\mathbb{R}$, which we index with upper and lower indices. Since each index ranges from $1 \dots n$ we have a total of n^{k+l} components of the tensor. We define these components by evaluating T on the relevant basis vectors.

$$T(e_{(i_1)}, \dots e_{(i_l)}, f^{(j_1)}, \dots, f^{(j_k)}) = T^{(j_1)\dots(j_k)}_{(i_1)\dots(i_l)}.$$

The evaluation of T on a general set of vectors with $s = 1 \dots l$ and $p = 1 \dots k$:

$$\begin{aligned} v_s &= \sum_{i=1}^{n} v_s^i e_i \\ \alpha^p &= \sum_{j=1}^{n} \alpha_j^p f^j. \end{aligned}$$

Takes the following form from multi linearity:

$$T(v_1, \dots, v_l, \alpha^1, \dots, \alpha^k) = \sum_{i_1=1}^{n} \cdots \sum_{i_l=1}^{n} \sum_{j_i=1}^{n} \cdots \sum_{j_k=1}^{n} T^{(j_1)\dots(j_k)}_{(i_1)\dots(i_l)} v_1^{i_1} \dots v_l^{i_l} \alpha^1_{j_1} \dots \alpha^k_{j_k}.$$

The extensive summation signs become cumbersome, and are therefore treated according to the *Einstein summation convention*. The idea is to omit the summation and regard it as implied whenever there is a match between a pair of upper and lower indexes in the tensor.

$$T(v_1, \dots, v_l, \alpha^1, \dots, \alpha^k) = T^{(j_1)\dots(j_k)}_{(i_1)\dots(i_l)} v_1^{i_1} \dots v_l^{i_l} \alpha^1_{j_1} \dots \alpha^k_{j_k}.$$

We clarify these ideas in brief by considering examples of tensors of small rank. It is customary to regard rank zero tensors as the underlying field ($\mathbb{R}$).

Example. *Taking $T \in V_1^0$ we get a type $(0,1)$ tensor. These are the linear maps $f : V \to \mathbb{R}$, and thus constitutes the vector space $V^\star$. $(0,1)$ tensors are therefore dual vectors.*
Example. *Conversely, taking $T \in V_0^1$ we get a type $(1,0)$ tensor - also of rank one, but this is a linear map $g : V^\star \to \mathbb{R}$. Since this itself is a vectorspace we get the dual of the dual $V^{\star\star}$ which without further exposition is isomorphic to V. We can therefore consider type $(1,0)$ tensors as vectors.*
Example. *Consider the linear maps $T : V \times V^\star \to \mathbb{R}$, they constitute rank two tensors of type $(1,1)$. With the knowledge that the vector spaces are n-dimensional, we write an arbitrary vector $v \in V$ using the basis $\{e_i\}_{i=1}^n$ as: $v = [v^1 \dots v^n]^T$.*
The linear function ℓ in $V^\star$, has the induced basis $\{\ell_j\}_{j=1}^n$, and we write $\ell = [\ell_1, \dots, \ell_n]$.
From the familiar vector arithmetic:

$$\ell(v) = [\ell_1, \dots, \ell_n] \begin{bmatrix} v^1 \\ \vdots \\ v^n \end{bmatrix} = \ell_1 v^1 + \dots + \ell_m v^m = \ell_i v^i.$$

Where the last term implies a summation, according to the Einstein convention.
Example. *It is often advantageous to construct a general "inner product" tensor. This allows us to quantify the notion of "distance" through a norm and as such, develop a metric space. Taking a tensor g of the form $(0,2)$, we consider the bilinear map $g : V \times V \to \mathbb{R}$, which satisfies:*

$$g(v,v) \geq 0 \tag{23}$$
$$g(v,w) = g(w,v) \tag{24}$$

Where property (1) is positive definite, that is $g(v,v) = 0$ only for $v = 0$. Property (2) is called symmetry. These properties determine an inner product on the space V. The bilinear map has an $n \times n$ matrix representation, which is acquired through evaluation of the function on the basis vectors.

$$g_{ij} = g(e_i, e_j).$$

By the requirement of symmetry of the map:

$$g_{ij} = g(e_i, e_j) = g(e_j, e_i) = g_{ji},$$

the matrix must be symmetric. The evaluation of this tensor on two vectors v, w with the basis $\{e_i\}_{i=1}^n$ takes the form from ordinary matrix multiplication:

$$[v^1, \ldots, v^n] \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ & & \vdots & \\ g_{n1} & g_{n2} & \cdots & g_{nn} \end{bmatrix} \begin{bmatrix} w^1 \\ \vdots \\ w^n \end{bmatrix}.$$

Which according to the Einstein convention reduces to:

$$g(v, w) = g_{ij} v^i w^j.$$

If we let the components of g_{ij} be the identity matrix, we get the usual dot product. We can however define other symmetric matrices for the tensor to construct an inner product. In the theory of Special Relativity, the "metric" is formed from this inner product, with the relaxation of the condition of being positive definitive, by the matrix:

$$g_{ij} = \begin{bmatrix} \text{-1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In the example where we paired a vector and a dual vector to arrive at the relation $\ell_i v^i$, which produces a scalar when the sum is taken (implied by Einstein convention), we say that we have *contracted* the tensor. This is a generalised form of trace operation where we can match vectors with dual vectors to reduce the rank of the tensor.

Definition 17. *The contraction operation on a type (p, q) tensor produces a type $(p-1, q-1)$ tensor. This is done by naturally pairing one vector space V_i and its dual $V_j^\star$ which produces a scalar after summation. The general procedure for contracting, is to identify one upper and one lower index in the tensor, setting them equal and taking the sum.*

Example. *The Riemann Curvature tensor $R^\mu_{\alpha\nu\beta}$ (ignoring its input arguments) can be contracted by setting ν equal to μ and performing a sum. The result*

$$R_{\alpha\beta} = R^\mu_{\alpha\mu\beta},$$

is called the Ricci tensor. Which is one of the cornerstones of the Einstein Field Equations of General Relativity.

Definition 18. *The symmetric group on the k elements $\{1 \dots k\}$ is denoted by S_k and constitutes all permutations of this set. $\sigma(i)$ denotes the value of the map on the i-th element. We say that a tensor is symmetric if it is unaffected by any permutation of its arguments.*

$$T(v_{\sigma(1)}, \dots v_{\sigma(k)}) = T(v_1, \dots v_k).$$

Since the components of the tensor are determined by the basis vectors, a symmetric tensor will be a symmetric array of coefficients, as the example of the metric tensor showed.

Example. *The source of the gravitational field is the energy and momentum of particles in space-time. This requires specification of energy density, which relies on the Stress-Energy Tensor T. This is a rank 2 tensor, defined as the flux of the α component of four-momentum across the surface defined by constant x^β. This tensor is symmetric since interchanging the arguments produces the same result. Using the dx notation common to General Relativity:*

$$T^{\alpha\beta} = T(dx^\alpha, dx^\beta) = T(dx^\beta, dx^\alpha) = T^{\beta\alpha}.$$

The interested reader should consult [14] for details and proof of this fact.

Since it is in our interest to take the derivatives of multivariate functions when we compute the derivatives of the IVP in order to find the Taylor expansion, we introduce multivariate derivatives as multilinear maps. Then, taking the k-th derivative produces a symmetric tensor of order k, which will allow us to connect the tensor notation introduced in this section with the trees in the next. To do this, the notion of a derivative for higher dimensions is presented over general vector spaces. However, it is worthwhile to devote special attention to the function, $f : \mathbb{R}^n \to \mathbb{R}$,

$$f(y) \in \mathbb{R} \text{ and } y \in \mathbb{R}^n.$$

Definition 19. *Let V and W be normed vector spaces, with norms $||\ ||_V$ and $||\ ||_W$ respectively. Furthermore, Ω is an open subset of V. A function $f : \Omega \to W$ is called differentiable at y in V if there exists a continuous linear map $T : V \to W$ such that:*

$$\lim_{h \to 0} \frac{||f(y+h) - f(y) - T(h)||_W}{||h||_V} = 0$$

In this form over general normed vector spaces, this definition is usually referred to as the Frechet[8] derivative, and like the single variable derivative, the map T is unique and we adopt the notation $Df(y)$ to mean the derivative of f at y.

[8]The Frechet derivative conceptualises the derivative on Banach spaces, that is - complete normed vector spaces. We are interested in $\mathbb{R}^n$ and $\mathbb{R}$ which under the appropriate norms become Banach spaces. However, the need for completeness is not necessary here.

In order to consider the higher derivatives such as $D(Df(y))$ (if they exist), we fall into the realm of multilinear maps. We justify this statement in brief by considering the linear maps as vector space homomorphisms, denoted by $Hom(\cdot,\cdot)$. We identify that if f is differentiable for every y in Ω, we get a linear map:

$$Df(y) : V \to W.$$

Therefore we get a map into a new vector space $Hom(V,W)$ according to:

$$Df : \Omega \to Hom(V,W).$$

If this map in turn is differentiable, we continue the process assuming differentiability at each step:

$$\begin{gathered} D^2 f : \Omega \to Hom(V, Hom(V,W)) \\ \vdots \\ D^k f : \Omega \to \underbrace{Hom(V, Hom(V, \dots Hom(V,W)}_{\text{k times}}\underbrace{)\dots)}_{\text{k times}}. \end{gathered}$$

It turns out that this vector space of k homomorphisms is isomorphic to k linear maps[9]. As a consequence, we may identify the two isomorphic vector spaces and therefore consider the k-th derivative $D^k f(y)$ of a function $f : \Omega \to W$ as a multilinear map with k arguments, writing $D^k f(y)(v_1, ..., v_k)$, with v_i in V. Recalling the properties of tensors, we associate the k-th derivative with the tensor $D^k f(y) : V^k \to \mathbb{R}$. The basis for V is denoted by $\{e_i\}_{i=1}^n$ as before. From the discussions we know that the coefficients of the tensor are determined by the values on its basis vectors, when $V = \mathbb{R}^n$ and Ω is an open subset of $\mathbb{R}^n$ these take the from according to the theorem:

Theorem 5. *Let $f : \Omega \to \mathbb{R}$ be a k-times differentiable function. Then for each y in Ω, the coefficients of the tensor $D^p f(y)$ with $p = 1 \dots k$ are given by:*

$$a_{i_1 \dots i_k} = \frac{\partial^k f(y)}{\partial y_{i_k} \dots \partial y_{i_1}}.$$

[9]Omitting the proof, this is a fundamental result of tensors and multilinear maps. The reader is referred to the literature for the details. Particularily [9], where information on Banach spaces can be found as well.

Proof. The theorem will be proved by induction on the order of the derivative. This is done by proving that:

$$D^k f(y)(v_1, \ldots, v_k) = \sum_{i_1}^{n} \sum_{i_2}^{n} \cdots \sum_{i_k}^{n} \frac{\partial^k f(y)}{\partial y_{i_k} \ldots \partial y_{i_1}} v_1^{i_1} v_2^{i_2} \ldots v_k^{i_k}$$

holds for $p = 1 \ldots k$.
The basis case constitutes $p = 1$: Assuming some familiarity with elementary vector calculus $Df(y)(v)$ is the directional derivative of f in the direction of v at y. Therefore we can express:

$$Df(y)(v) = \nabla f(y) \cdot v.$$

Where ∇ is the gradient operator and $\cdot$ is the dot product. Computing this expression yields,

$$Df(y)(v) = \sum_{i}^{n} \frac{\partial f(y)}{\partial y_i} v^i.$$

This proves the case $p = 1$. Assuming then, that the theorem holds for $p = 1, 2, \ldots, s$ where $s < k$. Let $F(y) = D^s f(y)(v_1, \ldots, v_s)$, then by hypothesis the function $F : \Omega \to \mathbb{R}$ is differentiable:

$$DF(y)(v_{s+1}) = D^{s+1} f(y)(v_1, \ldots, v_s, v_{s+1}).$$

The left hand side is the directional derivative in the direction of v_{s+1}. Taking the gradient of this function and the dot product with this vector gives:

$$DF(y)(v_{s+1}) = \nabla F(y) \cdot v_{s+1} = \sum_{i}^{n} \frac{\partial F(y)}{\partial y_i} v_{s+1}^i.$$

Combining this expression with the induction hypothesis gives the desired result

$$D^{s+1} f(y)(v_1, \ldots, v_{s+1}) = \sum_{i_1}^{n} \sum_{i_2}^{n} \cdots \sum_{i_{s+1}}^{n} \frac{\partial^{s+1} f(y)}{\partial y_{i_{s+1}} \ldots \partial y_{i_1}} v_1^{i_1} v_2^{i_2} \ldots v_{s+1}^{i_{s+1}}.$$

The theorem is true by induction. □

Theorem 6. *If $f : \Omega \to \mathbb{R}$ is a C^k map. Then for all y in Ω, the multilinear map is symmetric.*

Proof. In the case of f being a C^2 map, the symmetry refers to the equality of the mixed partial derivatives of $D^2 f(y)$, this is the well known *Schwarz theorem*. Taking this theorem for granted we use this as our basis case for inducting on k. Therefore assume the theorem up to $k-1$, that is:

$$D^{k-1} f(y)(v_{\sigma(2)}, \ldots, v_{\sigma(k)}) = D^{k-1} f(y)(v_2, \ldots, v_k). \tag{25}$$

Let $g(y) = D^{k-2} f(y)(v_3, \ldots, v_k)$, then by assumption g^2 and we apply Schwarz theorem to conclude:

$$\begin{aligned} D^2 g(y)(v_1, v_2) &= D^2 g(y)(v_2, v_1) \\ D^k f(y)(v_1, v_2, \ldots, v_k) &= D^k f(y)(v_2, v_1, \ldots, v_k). \end{aligned}$$

Furthermore, for any permutation $\sigma \in S_{k-1}$ acting on the set $\{2, \ldots, k\}$, equation (25) holds and we can write

$$\begin{aligned} D^k f(y)(v_1, \ldots, v_k) &= D(D^{k-1} f(y)(v_2, \ldots, v_k))(v_1) \\ &= D(D^{k-1} f(y)(v_{\sigma(2)}, \ldots, v_{\sigma(k)}))(v_1) \\ &= D^k f(y)(v_1, v_{\sigma(2)}, \ldots, v_{\sigma(k)}). \end{aligned}$$

Since the whole symmetric group S_k can be generated from the all permutations which leave 1 unchanged and the transposition of 1 and 2, the theorem is proven. □

We have shown that the derivatives can be expressed as multilinear maps (or tensors) according to theorem 5, we alleviate ourselves from the use of the summation signs by adopting the Einstein convention, to produce:

$$\frac{\partial^k f(y)}{\partial y_{i_k} \ldots \partial y_{i_1}} v_1^{i_1} v_2^{i_2} \ldots v_k^{i_k}.$$

Noting that it is possible to pair the upper and lower indices in this expression, we can perform the contraction operation on each pair to produce a scalar, which is the nature of the expression - as the tensor maps to $\mathbb{R}$. This is the notation used throughout the derivation of the order conditions, and its application will become clear over the next few pages.

6.2 Derivation

From the previous discussions on ODEs, without loss of generality, assume that we have an autonomous system:

$$\begin{bmatrix} y_0'(t) \\ y_1'(t) \\ \vdots \\ y_n(t)' \end{bmatrix} = \begin{bmatrix} 1 \\ f_1(y_0(t), y(t), \ldots, y_n(t)) \\ \vdots \\ f_n(y_0(t), y(t), \ldots, y_n(t)) \end{bmatrix}.$$

Which is defined by $f : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ on the domain $\Omega \subset \mathbb{R}^{n+1}$, with the initial conditions $y(t_0) = y_0$ where $t_0 \in \mathbb{R}$ and $y_0 \in \mathbb{R}^n$.

Remark. We change the notation from $y_i' = f_i(y_0(t), y_1(t), \ldots, y_n(t))$ to superscripts, which we write with capital letters. This is to make the connection with the tensor notation of the last section, and thus the J component will be written as:

$$(y^J)' = f^J(y^0, \ldots, y^n) \in \mathbb{R}.$$

We proceed to compute the derivatives of the exact solution for each component y^J in this notation, to do this we represent our system by:

$$\frac{dy(t)}{dt} = f(y(t)). \tag{26}$$

Let us assume that this differential equation satisfies the conditions of existence and uniqueness, together with being sufficiently smooth to take the Taylor expansion of desired order. Supposing that the true solution $y(t)$ also possesses derivatives of ample order we calculate the Taylor expansion of this solution at t_0. Recalling that we examine the true solution for a small deviation h from t_0, and the numerical solution y_1 located at $t_0 + h$.

Since the differential equation is a multivariate function, taking the derivatives of y^J with respect to t relies on taking partial derivatives and sums according to the chain rule. The tensor notation provides a convenient way to express this if we let

$$f^J_K = \tfrac{\partial f^J}{\partial y^K}.$$

As we continue to take derivatives, we consider the number of independent indexes "attached" to the function as the rank (disregarding J), and we can use this to "contract" the indices much in the same way as with tensors. The first few derivatives clarifies this idea.

The first derivative

$$y^J(t_0)^{(1)} = f^J(y(t_0)) = f^J(y_0) = f_0^J \tag{27}$$

is only indexed by J, as such we call it rank 0 (being a scalar). The elegance of this notation is particularly illustrated for the second and higher order derivatives. Employing the chain rule we get:

$$y^J(t_0)^{(2)} = f^J(y)^{(1)} y^{(1)}|_{y=y_0}$$

Writing this in summation form as required yields:

$$y^J(t_0)^{(2)} = \sum_K f_K^J(y)(y^K)^{(1)}\Big|_{y=y_0} = \sum_K f_K^J(y) f^K(y)\Big|_{y=y_0}. \tag{28}$$

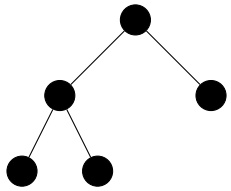
Here $f_K^J(y)$ is interpreted in this notation as a tensor of rank 1 and $f^K(y)$ as a tensor of rank 1. In this case we perform the contraction operation on the tensor with one vector to get the scalar $y^J(t_0)^{(2)}$. This occurs for each derivative by the nature of the chain rule, where the number of upper indexes should equal the number of lower indexes to perform this contraction. That is, the k-th derivative requires k vectors to perform the contraction. From now on every evaluation will be in $y = y_0$.
When taking the next derivative, an additional complication occurs in the form of the product rule. It produces two separate sums where we add an additional index L which we sum over. Together with the chain rule this produces the expression:

$$y^J(t_0)^{(3)} = \sum_{KL} f_{KL}^J f^L f^K + \sum_{KL} f_K^J f_L^K f^L. \tag{29}$$

For the first summand, a rank two tensor is contracted by two vectors. The second summand produces a scalar after two successive contractions. It is not immediately clear what will happen as we continue taking higher derivatives, the complexity quickly grows as we combine the rules of differential calculus. Furthermore, the tensor notation will not of itself lead to a more compact form of computing the derivatives. Instead, we introduce the aforementioned objects called trees to provide an abstraction, this will not necessarily simplify the computations, but allows the exact and numerical solutions of our system to be compared without the matching of individual derivatives employed in previous sections.

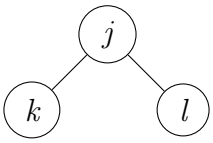
Definition 20. *A tree* $\boldsymbol{t}$ *is an undirected graph in which any two vertices are connected by exactly one simple path. We denote it with* $\boldsymbol{t} = (V, E)$ *where* V *is the set of vertices and* E *is the set of edges. Any connected graph without cycles is a tree.*

Example.



We label the vertices of the tree in the sense that if $\mathbb{A} = \{i, j, k, l, ...\}$ is our alphabet and $\boldsymbol{t} = (V, E)$ then, if there is a function $\phi : V \to \mathbb{A}$, we acquire a labelled tree.

Example.



Definition 21. *The number of vertices of the tree* $\boldsymbol{t}$ *is called the order of the tree and is denoted by* $\mid \boldsymbol{t} \mid = q$.

Definition 22. *Let* $\mathbb{A}$ *be our alphabet with an imposed ordering of the indices* $\{j < k < l < m \dots\}$. *We let* A_q *be the subset consisting of the first* q *indices. Then a rooted labelled tree of order* q *is the son-father mapping:*

$$\boldsymbol{t} : A_q \backslash \{j\} \to A_q.$$

Where $\{j\}$ *is the root of the tree, and all elements* $k \in A_q \backslash \{j\}$ *satisfy* $t(k) < k$. *We call this a son-father mapping since* k *is the son of* $t(k)$.

Therefore the tree defined by the map $\boldsymbol{t}$ will consist of sons, fathers, grandfathers all the way to the patriarch of the dynasty j. The set of all these mappings, is the set of all labelled trees of order q,which we call LT_q.

Example. *Let the alphabet be* $\mathbb{A} = \{j < k < l\}$*, then the son-father mapping* $t : A_3 \backslash \{j\} \to A_3$ *defines:*

$$\boldsymbol{t} = \begin{cases} t(l) = k \\ t(k) = j \end{cases} \longrightarrow \begin{array}{c} j \\ k \\ l \end{array}, \quad \boldsymbol{t} = \begin{cases} t(l) = j \\ t(k) = j \end{cases} \longrightarrow \begin{array}{ccc} & j & \\ k & & l \end{array}.$$

Definition 23. *For a labelled tree* $\boldsymbol{t} \in LT_q$ *we call*

$$F^J(\boldsymbol{t})(y) = \sum_{K,L,\ldots} f^J_{K\ldots}(y) f^K_{\ldots}(y) f^L(y) \ldots$$

the corresponding elementary differential. We sum over the $q-1$ *indicies from* $A_q \backslash \{j\}$ *which we denote with capitals* $K, L \ldots$*. Inside the summation we take the product of* q $f(y)$*'s with the upper indexes being the vertices of the tree and the lower index on each* f *denotes the sons of the upper index.*

To clarify this concept, we compute some elementary differentials corresponding to trees of small order with $\mathbb{A} = \{j, k, l, m, ...\}$ our ordered alphabet. For simplicity, we assume that LT_0 is the empty tree $\emptyset$ and its elementary differential is y^J. In the case of $LT_1$the tree only has one vertex j, and as such the elementary differential is simply f^J.

$$\begin{array}{c} \text{j} \end{array} \to f^J.$$

The next possible tree has two vertices and the associated elementary differential is that with respect to the second vertex k, which has no sons.

$$\begin{array}{c} \text{j} \\ \text{k} \end{array} \to f^J_K f^K.$$

When the tree has three vertices (2 excluding root) we have two possible trees, each with its own differential and the order in which we index is irrelevant by symmetry.

$$\begin{array}{ccc} & \text{j} & \\ \text{k} & & \text{l} \end{array} \to f^J_{KL}(f^K f^L).$$

$$\begin{array}{c} \text{j} \\ \text{k} \\ \text{l} \end{array} \to f^J_K(f^K_L f^L).$$

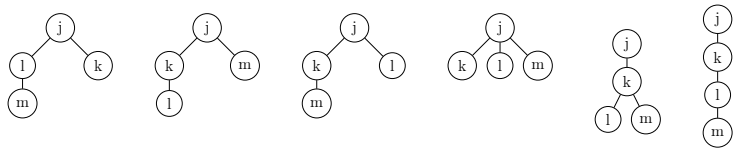
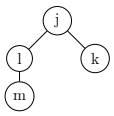


Table 3: The six trees of LT_4

A final example with four vertices should now be clear how to compute.



$$\rightarrow f^J_{KL}(f^L_M f^M f^K).$$

As the order of the trees q increases, we get a larger collection of different trees. This is because we can take each tree of order $q-1$ and choose where to put the new vertex (the son) to get a tree of order q. For trees with $q=4$, we begin with two trees, each with three vertices. Since we can connect the new vertex to a total of 3×2 other vertices, we have the six trees of table 3. Some of the trees in this table look very similar, apart from the labelling of the vertices. If we can permute the labels of a tree (disregarding the root) $\boldsymbol{t}_1 \in LT_q$ to correspond to the labels of $\boldsymbol{t}_2 \in LT_q$ we regard the two trees as being identical. Indeed, if there exists a permutation,

$$\pi : A_q \rightarrow A_q$$

where $\pi(\boldsymbol{t}_1) = \boldsymbol{t}_2$ and the root maps to itself, this defines an equivalence relation amongst the trees in LT_q.

Proof. Clearly $\boldsymbol{t}_1 \sim \boldsymbol{t}_1$, by the identity permutation.
$\boldsymbol{t}_1 \sim \boldsymbol{t}_2 \implies \boldsymbol{t}_2 \sim \boldsymbol{t}_1$ by inverse permutation.
$\boldsymbol{t}_1 \sim \boldsymbol{t}_2$ and $\mathbf{t}_2 \sim \boldsymbol{t}_3 \implies \boldsymbol{t}_1 \sim \boldsymbol{t}_3$ by composition of permutations. □

The crucial point however, is the fact that the trees for which this permutation holds, all have the same elementary differential. Since the permutation simply rearranges the labels - the elementary differential will look the same with a mere exchange of labels.

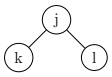
Definition 24. *Two trees are in the same equivalence class $\boldsymbol{t} \sim \boldsymbol{t}'$ if and only if the two trees have the same elementary differential. The set $T_q = LT_q/\sim$ is the partition of the labelled rooted trees into equivalence classes according to this relation.*

Definition 25. *Let $\alpha(\boldsymbol{t})$ denote the number of elements in the equivalence class of $\boldsymbol{t} \in T_q$. It is the number of permutations of the labelling of the vertices.*

By now, it should be clear that we can construct elementary differentials from our trees. If we return to our problem of calculating the derivatives of the true solution, we can see that the expressions where $f_K^J = \frac{\partial f^J}{\partial y^K}$ match our notation for the elementary differentials. It is therefore natural for us to try to write the exact solution in terms of the elementary differentials and trees. In the cases of the first and second derivatives of the true solution, it is clear from the preceding discussion, that these are equal to the elementary differentials of all trees of order 1 and 2 respectively. In each of these cases $|LT_1| = |LT_2| = 1$. Examining the third derivative of the exact solution, which was expressed as:

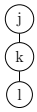
$$y^J(t_0)^{(3)} = \sum_{KL} f_{KL}^J f^L f^K + \sum_{KL} f_K^J f_L^K f^L.$$

There are two labelled trees in LT_3 and each corresponding to one of the summands.



$\rightarrow f_{KL}^J(f^K f^L)$

and



$\rightarrow f_K^J(f_L^K f^L).$

In the case of the fourth derivative, the exact solution of $(y^J(t_0))^{(4)}$ looks as follows:

$$\sum_{KLM} f_{KLM}^J f^L f^K f^M + 3\sum_{KLM} f_{KL}^J f_L^K f^L f^M + \sum_{KLM} f_K^J f_{LM}^K f^L f^M + \sum_{KLM} f_K^J f_L^K f_M^L f^M.$$

This corresponds to taking the elementary differentials of trees of order 4. The six trees of LT_4 were expressed in the previous table, and using their structure we can present the exact solution as:

$$\mathrm{y}^{(4)}(t_0) = F^J(\quad)(\mathrm{y}_0) + 3F^J(\quad)(y_0) + F^J(\quad)(y_0) + F^J(\quad)(y_0)$$

The elementary differential with the coefficient 3, should be of no surprise. If we recall table 3, there were 3 different trees which corresponded to this differential. The rest of them were representatives of their own class. This suggests that we can take successively higher derivatives, as we move to trees of higher order. The examples up to order 3 suggest that if we want to get a certain derivative, all we need to do is go to the known elementary differentials previous to the one we want and "contract" it by adding new vertices to our labelled trees. We illustrate this in the case of the third derivative of the true solution, we can take its derivative by adding new vertices to the labelled trees as follows; Taking the first summand, we add a new label M for each possible way to add the label m to the corresponding tree.

$$\sum_{KL} f^J_{KL} f^L f^K \rightarrow \begin{cases} \sum_{KLM} f^J_{KLM} f^L f^K f^M & (1) \\ \sum_{KLM} f^J_{KL} f^L_M f^M f^K & (2) \\ \sum_{KLM} f^J_{KL} f^L f^K_M f^M & (3) \end{cases}$$

$$\sum_{KL} f^J_K f^K_L f^L \rightarrow \begin{cases} \sum_{KLM} f^J_{KM} f^M f^K_L f^L & (4) \\ \sum_{KLM} f^J_K f^K_{LM} f^M f^L & (5) \\ \sum_{KLM} f^J_K f^K_L f^L_M f^M & (6) \end{cases}$$

The trees 2,3,4 all correspond to the same elementary differential, so we group them together with a real coefficient, in this case 3. and we yield the expression for $y^{(4)}(t_0)$. This coefficient is $\alpha(\boldsymbol{t})$, the number of elements in the equivalence class of $\boldsymbol{t}$. This leads us to the first of the two main theorems to prove the order conditions.

Theorem 7. *The exact solution of the autonomous ODE system satisfies*

$$(y^J)^{(q)}|_{y=y_0} = \sum_{t \in LT_q} F^J(\boldsymbol{t})(y_0) = \sum_{t \in T_q} \alpha(t) F^J(\boldsymbol{t})(y_0).$$

We prove this theorem by induction on q, to do this we need to introduce a lemma adapted from Butcher [2].

Lemma 4. *Let $A_q = A_{q-1} \cup \{z\}$ be a subset of the ordered alphabet $\mathbb{A}$, with cardinality q. Each member of A_{q-1} is strictly less than z. Let $\boldsymbol{t}$ be a member of LT_{q-1}.*
Then the derivative of the elementary differential

$$\tfrac{d}{dt} F(\boldsymbol{t})(y(t))$$

is the sum of $F(\boldsymbol{t}')(y(t))$ over all trees $\boldsymbol{t}' \in LT_q$ such that the subtree formed by removing z from the set of vertices is $\boldsymbol{t}$.

Proof. In the case where we only have one element j (the root), we get $A_2 = \{j, z\}$. Then $\boldsymbol{t} \in LT_1$ has the corresponding elementary differential $f^J(y(t))$. Taking the time derivative yields:

$$\tfrac{d}{dt} f^J(y(t)) = f^J_Z f^Z.$$

Which is precisely the sum of the single tree in LT_2 where $\boldsymbol{t} \in LT_1$ is obtained after removing the vertex z. This provides the base of our induction proof. Continuing, we let $A_q = \{j\} \cup B_1 \cup \cdots \cup B_k \cup \{z\}$. Where the lemma holds for $\{j\} \cup B_1 \cup \{z\}$, $\{j\} \cup B_1 \cup B_2 \cup \{z\}$ all the way to B_k, with each a disjoint subset of A_q.

By assumption, if $\boldsymbol{t} \in LT_{q-1}$ then $\boldsymbol{t}$ can be expressed as the combination of the trees $\boldsymbol{t}_i \in LT_{q_i}$ joined with the root j. The elementary differential of this tree can therefore be expressed as:

$$F(\boldsymbol{t})^J(y(t)) = f^J_{J_1 J_2 \ldots J_k}(f^{J_1}_{\ldots} \ldots)(f^{J_2}_{\ldots} \ldots) \ldots (f^{J_k}_{\ldots} \ldots).$$

Where $f^{J_i}_{\ldots} \ldots$ denotes the elementary differential with respect to the tree $\boldsymbol{t_i}$ and labels in the set B_i. Taking the time derivative of this expression corresponds to a summation of $E_0 + E_1 + \ldots E_k$ terms. Where:

$$\begin{aligned} E_0 &= f^J_{J_1 J_2 \ldots J_k Z}(f^{J_1}_{\ldots} \ldots) \ldots (f^{J_k}_{\ldots} \ldots)(f^Z) \\ E_i &= f^J_{J_1 J_2 \ldots J_k}(f^{J_1}_{\ldots} \ldots) \ldots (\frac{d}{dt} f^{J_i}_{\ldots} \ldots) \ldots (f^{J_k}_{\ldots} \ldots). \end{aligned}$$

E_0 is evaluated to $F^J(\boldsymbol{t}_1 \boldsymbol{t}_2 \ldots \boldsymbol{t}_k \boldsymbol{t}_0)(y(t))$, with the same combination of trees to the root, and in addition $\boldsymbol{t}_0$, which is the label z. For each term E_i is the sum of all terms on the form of $\frac{d}{dt} F^{J_i}(\boldsymbol{t}_i)(y(t))$, which is $F^{J_i}(\boldsymbol{t}_i)(y(t))$ replaced by a sum of terms of the form $F(\boldsymbol{u}_i)(y(t))$, where $\boldsymbol{u}_i$ is formed from $\boldsymbol{t}_i$ by adding an additional leaf labelled by z. The lemma follows by combining all these terms contributing to the derivative. □

The proof of theorem 7 is now straightforward.

Proof. We prove this theorem by induction on q. It has already been seen that the theorem holds for $q = 0, 1, 2, 3, 4$ and as such we assume that the theorem holds for $q = 0, 1, \dots q-1$. The first step is to note that we can go from LT_{q-1} to all of LT_q by adding a new vertex $\{q\}$ to every possible vertex of all the trees in LT_{q-1}. Secondly, we can derive the elementary differentials of LT_q from LT_{q-1}. We do this by taking each tree in LT_{q-1} and its corresponding elementary differential. For simplicity we consider $\boldsymbol{t} \in LT_{q-1}$ with the corresponding elementary differential $F^J(\boldsymbol{t})(y_0)$. For each addition of the vertex $\{q\}$ to $\boldsymbol{t}$ we produce a new elementary differential from the old one which corresponds to a tree in LT_q, as we showed in the lemma.

Since we assume the theorem holds up to $q-1$, we know that:

$$(y^J)^{(q-1)}|_{y=y_0} = \sum_{\boldsymbol{t} \in LT_{q-1}} F^J(\boldsymbol{t})(y_0).$$

If we take the derivative with respect to t, we know that differentiating an elementary differential corresponds to adding a new summation index to each elementary differential, i.e adding the new vertex in each possible place, and taking their sum. This becomes the sum of the elementary differentials of all the trees of LT_q. Since we have already shown the cases up to $q = 4$, the theorem is true by induction. We can group the equivalence classes together by $\alpha(\boldsymbol{t})$ without loss of generality to yield the second equality. □

We have finally used all this cumbersome notation, trees and heart wrenching derivatives to arrive at a compact result. We now turn to the second phase of the derivation of the order conditions, the Taylor expansion of the numerical solution. We wish to write the numerical solution much in the same way that we did for the exact i.e. we wish to find a correspondence to the trees. Indeed this is possible through a famous formula for the derivatives of composite functions, *Faa Di Bruno's formula*. We will not prove this lemma, but merely state that the form in which we express it, satisfies its assumptions.

We consider the same autonomous system that we used to derive the exact solution. Taking the Runge-Kutta scheme and for each stage k_i we would normally write:

$$k_i = f(t_n + c_i h, y_n + h \sum_{j=1}^{i-1} a_{ij} k_j).$$

Since the system is autonomous, and to simplify our calculations we will write $k_i = f(g_i)$ where;

$$g_i^J = y_0^J + \sum_{j=1}^{i-1} a_{ij} h f^J(g_j^1, \ldots, g_j^n) \; i = 1 \ldots s \tag{30}$$

$$y_1^J = y_0^J + \sum_{j=1}^{s} b_j h f^J(g_j^1, \ldots, g_j^n). \tag{31}$$

We must compare the Taylor series of y_1^J with the exact solution. We proceed to compute the derivatives of y_1^J and g_i^J with respect to h, evaluated at $h = 0$. Upon inspection of the two preceding equations they are both of the form $h\phi(h)$, if we take the derivatives and evaluate at $h = 0$ we can make use of Leibniz formula,

$$(h\phi(h))^{(q)} = q(\phi(h))^{(q-1)}. \tag{32}$$

Then

$$\begin{aligned} (g_i^J)^{(0)}\big|_{h=0} &= y_0^J \\ (g_i^J)^{(1)}\big|_{h=0} &= \sum_j a_{ij} f^J\big|_{y=y_0}. \end{aligned}$$

We compute the second derivative from Leibniz formula, we must therefore compute $f^J(g_j)$ first.

$$(f^J(g_j))^{(1)} = \textstyle\sum_K f_K^J(g_j)(g_j^K)^{(1)}.$$

Here, the notation from the previous section becomes important. We let f_K^J denote the partial derivative $\frac{\partial f^J}{\partial y^K}$. If we remember the chain rule, the derivatives can be written as a sum over the partial derivatives. Furthermore, we use this notation as it naturally gives the connection with the labelled trees.

$$(g_i^J)^{(2)}\big|_{h=0} = 2 \sum_{j,k} a_{ij} a_{jk} \sum_K f_K^J f^K\big|_{y=y_0}.$$

Continuing taking derivatives with the appropriate substitutions;

$$(g_i^J)^{(3)}\big|_{h=0} = 3 \sum_{j,k,l} a_{ij} a_{jk} a_{jl} \sum_{KL} f_{KL}^J f^K f^L + 6 \sum_{j,k,l} a_{ij} a_{jk} a_{kl} \sum_{KL} f_K^J f_L^K f^L.$$

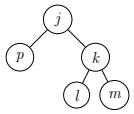
We can see in the expression for the third derivative, that there is a correspondence between the indices a_{jk} in the same way that the upper and lower indices are linked on the derivatives. This connection suggests that we view the derivatives graphically, considering labelled trees. Before we make this leap, we must make some definitions.

Definition 26. *Let $\boldsymbol{t}$ be a tree with the root j. We denote the elementary weight by*

$$\Phi_j(\boldsymbol{t}) = \sum_{k,l\ldots} a_{jk} a_{\ldots} \ldots$$

the sum over the remaining $q-1$ indices $k, l \ldots$ where all fathers stand two by two with their sons as indices.

Example. *Given the tree $\boldsymbol{t}$ in LT_5 of the form:*

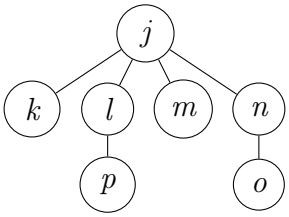


Then the elementary weight:

$$\Phi_j(\boldsymbol{t}) = \sum_{k,l,m,p} a_{jp} a_{jk} a_{km} a_{kl}.$$

Definition 27. *LS_q is the set of special labelled trees. These are the trees with no ramifications except at the root.*

Example.



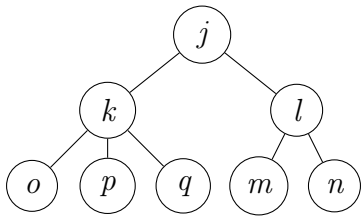
Lemma 5. *(Faa di Bruno's formula) For $q \geq 1$*

$$(f^J(g))^{(q-1)} = \sum_{\boldsymbol{u} \in LS_q} \sum_{K_1,\ldots K_m} f^J_{K_1,\ldots K_m}(g)(g^{K_1})^{\delta_1} \ldots (g^{K_m})^{\delta_m}.$$

$\boldsymbol{u}$ is a special labelled tree of order q and m is the number of its branches. Each δ_i are the number of vertices on these branches. For the lemma to hold, these positive integers must satisfy the constraint $q = 1 + \delta_1 + \cdots + \delta_m$ which is satisfied by the nature of the trees in LS_q.

Definition 28. *For $\boldsymbol{t} \in LT_q$ let $\gamma(\boldsymbol{t})$ be the positive integer obtained when we multiply the order of the tree (q) with all orders of the trees obtained after removing the roots in succession from $\boldsymbol{t}$.*

Example.



The tree is of order 8, removing the root of $\boldsymbol{t}$ gives two trees, each of order 4 and 3 respectively. Removing their roots gives trees of order 1. Therefore $\gamma(\boldsymbol{t}) = 8 \times 4 \times 3$.

Combining these ideas we proceed to prove the second of our two necessary theorems.

Theorem 8. *The derivatives of g_i^J satisfy*

$$(g_i^J)^{(q)}|_{h=0} = \sum_{\boldsymbol{t} \in LT_q} \gamma(\boldsymbol{t}) \sum_j a_{ij} \Phi_j(\boldsymbol{t}) F^J(\boldsymbol{t})(y_0).$$

The numerical solution y_1^j satisfies

$$(y_1^J)^{(q)}|_{h=0} = \sum_{\boldsymbol{t} \in LT_q} \gamma(\boldsymbol{t}) \sum_j b_j \Phi_j(\boldsymbol{t}) F^J(\boldsymbol{t})(y_0).$$

Proof. Noting that the form of the two expressions are very similar, we proceed to prove this for g_i^J only. Once again we proceed by an induction proof on q. From the previous discussion, we used Leibniz's formula and with its aid we can assume that:

$$(g_i^J)^{(q)}|_{h=0} = q \sum_j a_{ij} (f^J(g_j))^{(q-1)}|_{y=y_0}.$$

From Faa Di Bruno's formula we can express this as a rather complicated expression using LS_q and LT_{δ_i}. This allows us to use the induction hypothesis with $\delta_i < q$ to rewrite the sum as:

$$(g_i^J)^{(q)}|_{h=0} = q \sum_{\boldsymbol{u}\in SL_q} \sum_{\boldsymbol{t}_1\in LT_{\delta_1}} \cdots \sum_{\boldsymbol{t}_m\in LT_{\delta_m}} \gamma(\boldsymbol{t}_1)\ldots\gamma(\boldsymbol{t}_m)$$
$$\sum_j a_{ij} \sum_{k_1} a_{jk_1}\Phi_{k_1}(\boldsymbol{t}_1)\cdots\sum_{k_m} a_{jk_m}\Phi_{k_m}(\boldsymbol{t}_m)$$
$$\sum_{K_1,K_2,\ldots,K_m} f^J_{K_1,K_2,\ldots,K_m}(y_0)F^{K_1}(\boldsymbol{t}_1)(y_0)^{K_m}(\boldsymbol{t}_m)(y_0).$$

If we can create a bijective correspondence from each collection $(\boldsymbol{u},\boldsymbol{t}_1,\ldots,\boldsymbol{t}_m)$ with $\boldsymbol{u}\in LS_q$ and $\boldsymbol{t}_i\in LT_{\delta_i}$ to the set LT_q. Then the expressions in the equality can be reduced to functions of $\boldsymbol{t}$ in LT_q of the form

$$\gamma(\boldsymbol{t}) = q\gamma(\boldsymbol{t}_1)\ldots\gamma(\boldsymbol{t}_m) \tag{33}$$
$$F^J(\boldsymbol{t})(y) = \sum_{K_1,K_2,\ldots,K_m} f^J_{K_1,K_2,\ldots,K_m}(y_0)F^{K_1}(\boldsymbol{t}_1)(y_0)^{K_m}(\boldsymbol{t}_m)(y_0) \tag{34}$$
$$\Phi_j(\boldsymbol{t}) = \sum_{k_1,\ldots k_m} a_{jk_1}\ldots a_{jk_m}\Phi_{k_1}(\boldsymbol{t}_1)\ldots\Phi_{k_m}(\boldsymbol{t}_m). \tag{35}$$

Taking this statement for granted now allows us to express $(g_i^J)^{(q)}|_{h=0}$ as the sum over LT_q, of the following form:

$$(g_i^J)^{(q)}|_{h=0} = \textstyle\sum_{\boldsymbol{t}\in LT_q}\gamma(\boldsymbol{t})\sum_j a_{ij}\Phi_j(\boldsymbol{t})F^J(\boldsymbol{t})(y_0).$$

By the similarity of the formulas, we can simply replace b_j with a_{ij}, and we yield the expression for the numerical solution for the first RK step:

$$(y_1^J)^{(q)}|_{h=0} = \textstyle\sum_{\boldsymbol{t}\in LT_q}\gamma(\boldsymbol{t})\sum_j b_j\Phi_j(\boldsymbol{t})F^J(\boldsymbol{t})(y_0).$$

For the case where $q=1$ the formula predicts:

$$(\mathrm{y}_1^J)^{(1)} = \textstyle\sum_{\boldsymbol{t}\in LT_1}\gamma(\boldsymbol{t})\sum_j b_j\Phi_j(\boldsymbol{t})F^J(\boldsymbol{t})(y_0).$$

Which by the trivial nature of the tree functions involved gives

$$\sum_j b_j f^J(y_0), \tag{36}$$

which was calculated previously, and the theorem is true by induction. □

The bijection required can be constructed if we take the collection $(\boldsymbol{u}, \boldsymbol{t}_1, \ldots, \boldsymbol{t}_m)$, replacing the branches of $\boldsymbol{u}$ with $\boldsymbol{t}_1, \ldots, \boldsymbol{t}_m$.

With the derivatives of the true and numerical solution expressed in terms of trees, the Runge-Kutta order conditions now becomes a matching of the derivatives and we have proved our main result:

Theorem 9. *A Runge-Kutta method is of order p if*

$$\sum_j b_j \Phi_j(\boldsymbol{t}) = \frac{1}{\gamma(\boldsymbol{t})}$$

for all trees $\boldsymbol{t}$ of order $\leq p$.

This can be made into an if and only if statement, given that the elementary differentials are *independent*. The proof of this is beyond the scope of this paper, but the interested reader should consider the following proof sketch,

Proof. By showing that for every $\boldsymbol{t} \in T_q$ there exists a system of differential equations such that the elementary differential $F^J(\boldsymbol{t})(y_0)$ of this tree at the initial point evaluates to 1. For all other trees, their elementary differentials at the initial point evaluate to 0. □

Whenever we can satisfy the order conditions, we place a bound of the local error of the method in terms of the Taylor series. As a result of the theorems in section 3, we guarantee convergence of the method constructed, and one may happily apply the method to a wide range of problems. Should the method fail its task when implemented, one might have encountered a *stiff equation*. Tackling such difficulties would be the topic of another paper.

"Prova Runge-Kuttan ... den är bra! applicera och må bra!"

From the song Runge-Kuttan, lyrics and music by Mathias Lundgren.

Appendix

This appendix provides a brief illustration of the implementation of Runge-Kutta methods constructed in section 5. The paper has dealt mainly with the order of convergence, in theory this is essential for a qualitative approximation, but does not give the complete story. Given some differential equation we wish to solve, it is not enough to simply pick a working method and a small enough step size to guarantee a good approximation. There are essentially two answers to this, one practical and one theoretical.

- Cost:
  The smaller the step size, the more computational steps have to be made to cover the whole interval of approximation. Therefore there is a tradeoff between accuracy and number of computations - a serious concern when implementing by hand, but also by computer. The computers used for calculation have finite precision in their arithmetic and limitations on running time, as a result the step size must adhere to physical realities and not theoretical results. It is commonplace in applications to place a bound on the total error in the computation (truncation, accumulation from roundoff etc) and have a subroutine in the algorithm ensuring the limit is not exceeded, and if it is, reduce the step size accordingly, this is called adaptive step size. In some problems this restriction on the total error makes the cost of computation unfeasible (usually stiff problems). We can picture this by thinking of a solution curve with long periods where the derivative is rather constant and short periods where the derivative changes rapidly. Had we implemented RK4 with a relatively large step size (constant), we could have obtained an accurate approximation over the long time period, but when the derivative changes rapidly the step size is too large for good accuracy. A naive approach is then to use a step size small enough to conquer this difficult section, but then we have paid too much for the "easy" section. A compromise seems necessary, but a final caveat is that even for an adaptive method, the "difficult" section might force the computer to make the step size impractically small. The explicit Runge-Kutta methods are by no means "universal solvers" for these types of problems, and emphasis in recent years have been the adaptation of RK methods which can handle such difficulties (often implicit)[10].

[10]This is treated in the second volume of [7]

- Stability:
  The theory of stability in dynamical systems is deep and as such any short treatment can only encompass a small part of the theory. For in depth treatises we refer to [7] [2]. To describe the subject in a few sentences we consider the following mechanical system. A small mass is constrained to move along inner wall of a one meter inclined cylinder. We pick an initial condition for the mass' motion corresponding to some placement of the mass at the top of the inclined cylinder and a position on the circle making its base. It is intuitive that an initial condition on the bottom of the circle gives a straight line, and a solution corresponding to a slightly perturbed initial condition will oscillate around the non perturbed. On the other hand, the initial condition diametrically opposite, gives a straight line motion - but solutions corresponding to this initial condition perturbed, do not remain "close" to the original solution. When approximating solutions with one step methods, analysing the stability of the approximated solution becomes important since if the solution is unstable, the errors made in one step of computation, will bring the next step further from the true solution, and the computation ultimately fails to describe the differential equation. There are numerous ways to deal with this issue, and modern methods are often designed with stability considerations in mind.

We discuss these considerations and implementation procedures as we examine two related oscillatory systems, Simple Harmonic Motion and the Van Der Pol equation.

Simple Harmonic Motion

In this section we discuss some elementary stability issues and accuracy of the methods of section 5, whilst solving the equation for Hooke's law for a particle under spring action presented in section 1. Consider the linear system:

$$\frac{dy}{dt} = Ay.$$

Assuming that the $n \times n$ matrix A has a basis of eigenvectors, the general solution would be of the form:

$$y(t) = \sum_{i=1}^{n} C_i e^{\lambda_i t} v_i,$$

where λ_i are the eigenvalues of A, and v_i the corresponding eigenvector and C_i are constants. Since such a linear system only possesses one equilibrium point (the origin). We can analyse its stability by looking at the eigenvalues of A. If the eigenvalues are all negative, the origin is stable,as the solution will remain close to the origin as time goes to infinity. Other restrictions can be placed upon the eigenvalues, to guarantee different types of stability, but will not be dealt with here. It is important to note that we examine stability for linear systems in this way, for non-linear equations the analysis is different. However, linearisation theorems for equilibrium points reduce much non-linear stability theory to the linear case, and for many purposes suffices. Since we are interested in how our one-step methods handle stability issues we consider approximating this system with the Euler method.

$$y_{n+1} = y_n + hAy_n$$
$$y_{n+1} = (I + hA)y_n.$$

We expand the last line in the eigenbasis to yield:

$$(I + hA)(a_1^n v_1 + \ldots a_n^n v_n). \tag{37}$$

Since each v_i is an eigenvector we compactly write:

$$y_{n+1} = \sum_{i=1}^{n} a_i^n (1 + h\lambda_i) v_i.$$

We can also express y_{n+1} as:

$$y_{n+1} = \sum_{i=1}^{n} a_i^{n+1} v_i.$$

Upon combining these equations we get:

$$a_i^{n+1} = (1 + h\lambda_i)a_i^n.$$

From this expression we see that the origin is a stable point of approximation if:

$$|1 + h\lambda_i| \leq 1 \quad i = 1 \ldots n.$$

We call the set of such points in the complex plane (the eigenvalues may be complex) the region of stability for Euler's method. The implication is that, in order to provide a stable approximation, one must consider not only the step size, but also the eigenvalues of the system itself. Resultantly, there are eigenvalues for which Euler's method cannot be stable, regardless of the step size. For such problems, a different method should be chosen, for which the eigenvalues do lie in its region of stability. For higher order RK methods applied to this linear system we may write:

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

Where if the method is explicit $\Psi(h\lambda)$ is a polynomial function - the stability function, for which the origin is stable if $|\Psi(h\lambda)| \leq 1$. Since the explicit Runge-Kutta methods in this paper are constructed to coincide with the Taylor polynomial of corresponding degree, it is clear that second order methods will have a stability function $1 + h\lambda + \frac{(h\lambda)^2}{2}$, and so forth for higher order methods. The set of points satisfying $|\Psi(h\lambda)| \leq 1$ increases with order, but since such polynomials are unbounded, no explicit RK method can have a stability region covering the whole complex plane, and therefore the correct choice of method becomes paramount. We now turn to the problem proposed at the beginning of this section, approximating a second order linear equation. We will consider the eigenvalues of the system and comment on stability.

$$\frac{d^2y}{dt^2} = -(\frac{k}{m})y \qquad \frac{k}{m} \in \mathbb{R}. \tag{38}$$

We impose that the particle is displaced one meter from its equilibrium position at the initial time $t = 0$. Furthermore we assume that the particle at this time has zero velocity.

$$\begin{aligned} y(0) &= 1 \\ \frac{dy(0)}{dt} &= 0. \end{aligned}$$

We saw that the solution to this equation, without imposing the initial conditions were:

$$y(t) = c_1 e^{iat} + c_2 e^{-iat}.$$

Where the c_1, c_2 are two real constants. The eigenvalues $\pm ia$ are obtained from the characteristic equation of the ODE. By the nature of the initial conditions, we reduce to:

$$\begin{aligned} c_1 + c_2 &= 1 \\ c_1 &= c_2 \\ c_1 = c_2 &= \frac{1}{2}. \end{aligned}$$

Factoring we can express this as:

$$c_1(e^{iat} + e^{-iat}).$$

Rewriting in polar form of complex numbers and using the properties of even and odd functions we reduce the exact solution to

$$y(t) = cos(t). \tag{39}$$

For simplicity we have assumed that $a = 1$, That is, the ratio of the mass and spring constants are equal to one. We make a note of the imaginary eigenvalues $\pm i$ as we proceed to solve this equation numerically. Transforming this second order equation into a first order system by a change of variables;

$$\begin{aligned} \frac{dy}{dt} &= v \\ \frac{dv}{dt} &= -y. \end{aligned}$$

We investigate the solutions on the time interval $[0, 50]$, using the *Matlab* computational software, for each of the methods presented in the section on Runge Kutta methods. To implement these we construct algorithms using a specific Runge Kutta scheme and save these as a function, which matlab calls. The results are then plotted against the analytic solution.

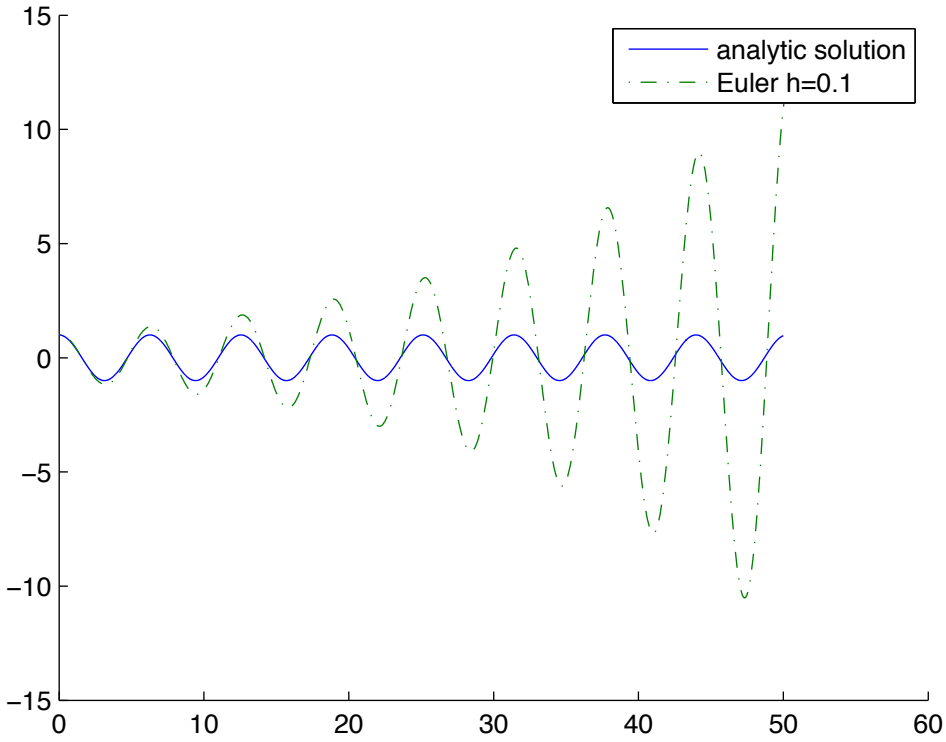


Figure 2: Analytic versus Euler solution

Euler Method:

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}.$$

Even with a generous step size of 0.1, as time increases, the numerical solution predicts a larger amplitude of oscillation. Apart from being notably different to the analytical solution, this would physically correspond to a violation of energy conservation. As such we would have to compute with yet a smaller step size, or revert our attention to the higher order methods which clearly give better approximations, even for larger step sizes. However, this only improves our approximation locally. Over large time intervals, no reduction in step size helps due to stability considerations of Euler's method. The eigenvalues of the problem are purely imaginary, and the stability region of Euler's method does not contain the imaginary axis, exept the origin. Resultantly, we must use a different method for large intervals.

Matlab Code

```
function [ tout, yout ] = euler( F, t_beg, t_end, h, Y_0 )

tout = [t_beg];
yout = [Y_0(:)'];

for t=t_beg:h:t_end
    Y_0 = Y_0 + h*F(t, Y_0);
    tout = [tout; t+h];
    yout = [yout; Y_0(:)'];
end

end
```

```
f = @(t, Y) -Y(1);
F = @(t, Y) [Y(2); f(t, Y)]; %Functions from ODE and system.
Y_0 = [1; 0]; %Initial condition.

t_beg = 0;
t_end = 50;        % Defines the time interval we wish to solve over [0,50]

hold all;            % Plots analytical and Euler solutions on same graph.


t = t_beg:0.001:t_end;
plot(t, cos(t));
legends = {'analytic solution'}; % Plots analytical solution with label.

%Calling the Euler method and plotting.

for h = [0.1]; % Stepsize.
    [tout, yout] = euler(F, t_beg, t_end, h, Y_0);
    plot(tout, yout(:,1),':');
    legends = [legends {strcat('euler h=', num2str(h))}];
end

legend(legends);
```

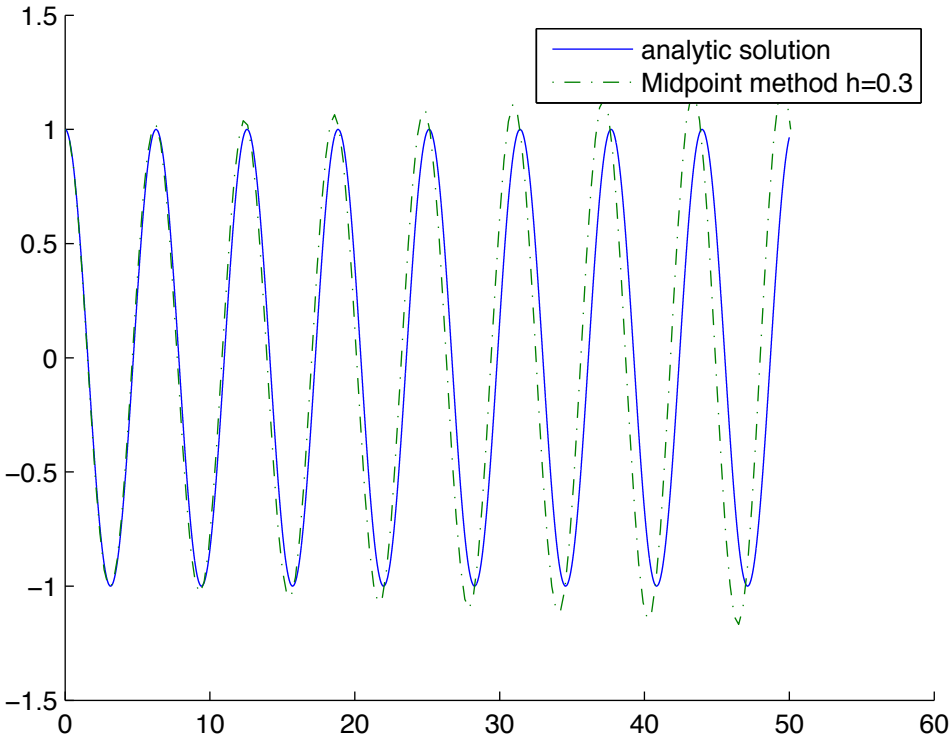


Figure 3: Analytic versus Midpoint solution

Midpoint Method:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}.$$

With the midpoint method, we see that the use of a second order method provides a more accurate approximation to the problem, the large oscillations of Eulers method have been reduced over the interval. However, we have only provided a more accurate solution locally, since the stability region of the midpoint method does not include $\pm i$. If we were to extend the interval, similar behaviour to Euler's method would be seen.

```
function [ tout, yout ] = midpoint( F, t_beg, t_end, h, Y_0 )

% Follows the same design structure as the euler function, but implements
% second order runge kutta instead.

tout = [t_beg];
yout = [Y_0(:)'];

for t=t_beg:h:t_end
    k1 = F(t, Y_0);
    k2 = F(t + 0.5*h, Y_0 + 0.5*h*k1);
    Y_0 = Y_0 + h*k2 ;
    tout = [tout; t+h];
    yout = [yout; Y_0(:)'];
end

end
```

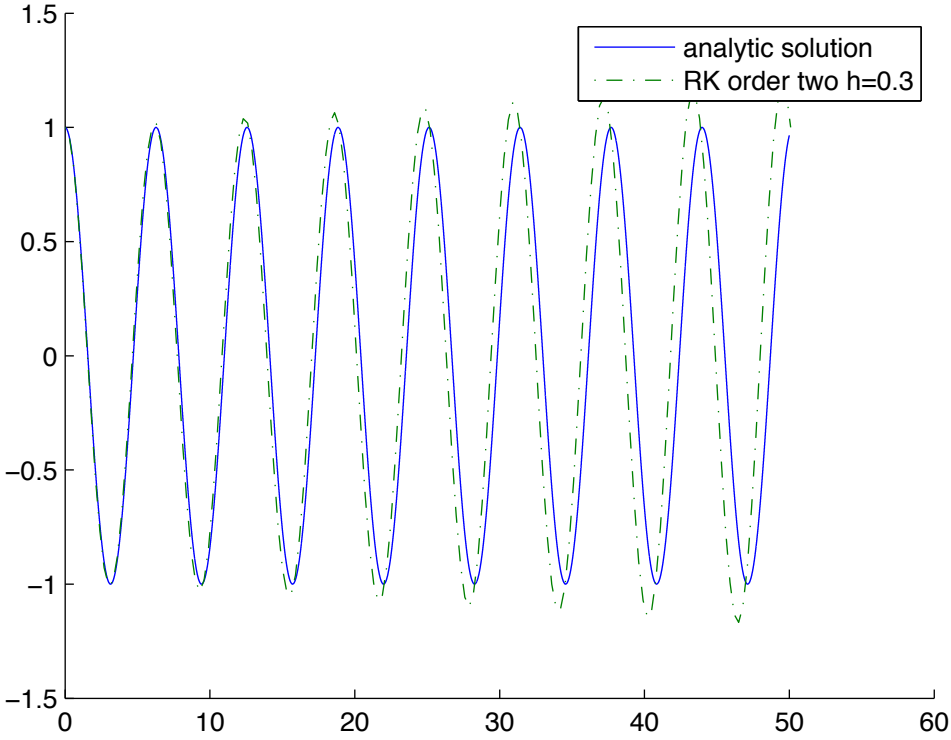


Figure 4: Analytic versus second order solution

Second order RK, parametrised by $b_1 = \frac{1}{4}$.

$$\begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

```
function [ tout, yout ] = custtwo( F, t_beg, t_end, h, Y_0 )

tout = [t_beg];
yout = [Y_0(:)'];

for t=t_beg:h:t_end
    k1 = F(t, Y_0);
    k2 = F(t + (2/3)*h, Y_0 + (2/3)*h*k1);
    Y_0 = Y_0 + h*((1/4)*k1 + (3/4)*k2);
    tout = [tout; t+h];
    yout = [yout; Y_0(:)'];
end

end
```

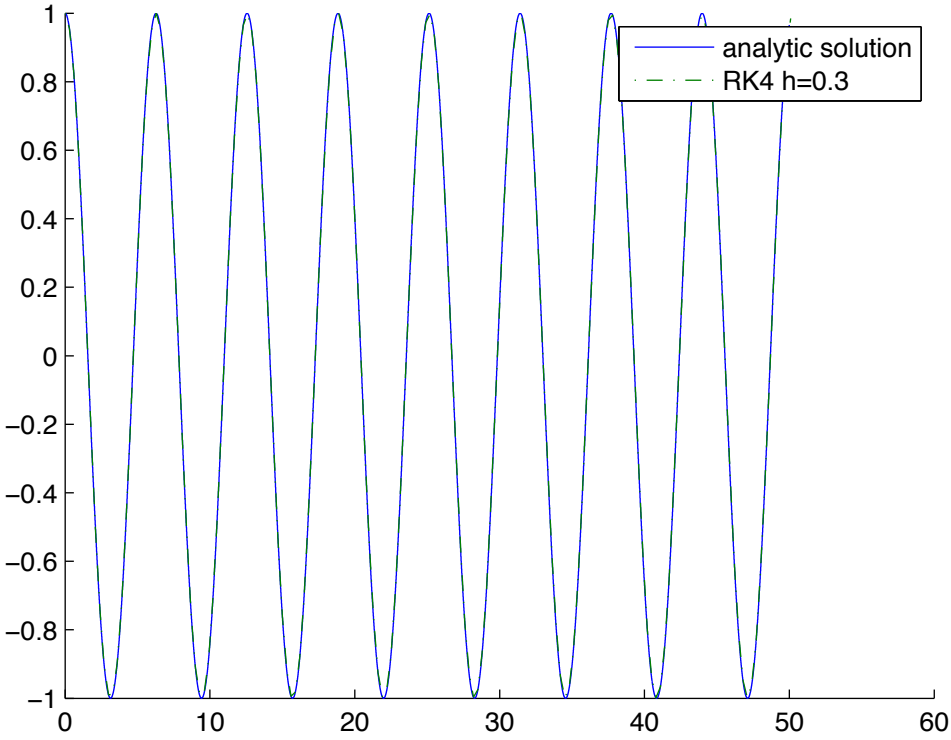


Figure 5: Analytic versus RK4 solution

RK4:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

The RK4 method provides an excellent approximation to the analytical solution, which does not diverge over the interval. The reason for this is that the stability polynomial of RK4 contains the imaginary line for small enough step sizes, making the method stable for this problem.

```
function [ tout, yout ] = rk4( F, t_beg, t_end, h, Y_0 )

tout = [t_beg];
yout = [Y_0(:)'];

for t=t_beg:h:t_end
    k1 = F(t, Y_0);
    k2 = F(t + 0.5*h, Y_0 + 0.5*h*k1);
    k3 = F(t + 0.5*h, Y_0 + 0.5*h*k2);
    k4 = F(t + h, Y_0 + h*k3);
    Y_0 = Y_0 + 1/6*h*(k1 + 2*k2 + 2*k3 + k4);
    tout = [tout; t+h];
    yout = [yout; Y_0(:)'];
end

end
```

Van der Pol Equation

In this section we examine the Van der Pol equation, named after Balthazaar Van der Pol who studied it in the early twentieth century. When applied to electrical circuits its form reminds us of the damped and/or driven harmonic oscillator in mechanics and is one of the early systems discovered involving analogue computation. The equation represents a non-linear oscillating system often used to test numerical methods, but also possesses a wide range of useful applications, ranging from over-clocking your iphone to heart beats.

$$y'' - \mu(1 - y^2)y' + y = 0 \tag{40}$$

When $\mu = 0$ the equation reduces to Hooke's law, a second order linear equation for which the dynamics of the explicit Runge-Kutta methods constructed in the paper were analysed in the previous section. We saw that the quality of the approximations relied on both step size and the method chosen, in particular the only stable method was RK4. For μ being non-zero the problem becomes much more intractable, consider for instance:

$$y'' + \beta y' + y = 0 \tag{41}$$

The damped oscillator, for which solutions corresponding to $\beta > 0$ are decaying oscillations and exponentially growing solutions for $\beta < 0$. We interpret this as the "damping/boosting" factor which either add or remove energy from the system. Van der Pol had the insight to take this damping term and replace it with the non-linear expression $y'(1 - y^2)\mu$. The effect of this replacement is that, depending on the absolute value of y and the sign of μ, this term governs the resulting growth/decay behaviour. The importance of this term is governed by the magnitude of μ, as we shall see in the following computations (positive case only). For large values, we are able to illustrate an example of a stiff system and its problems relating to cost. In order to make this precise, linearisation and stability analysis would be required. We therefore argue loosely that, as can be seen from subsequent plots, large values of μ give rise to alternating slow and fast transients in the derivative, which can bee seen after turning the equation into a two dimensional system and evaluating its Jacobian at $(0, 0)$:

$$\begin{bmatrix} 0 & 1 \\ -1 & \mu \end{bmatrix}.$$

From analysis of this matrix one can deduce that large values of μ, give rise to large rates of change in the solution, and the eigenvalues are real and positive so that the "steady state solution" $(0, 0)$ is unstable. These features characterise a stiff equation, although no precise mathematical definition exists. We will show that for small values of μ, Euler and RK4 provide good local approximations, but for higher values other methods have to be employed. It should also be noted that the Van der Pol equation satisfies Lienard's Theorem and therefore has a unique limit cycle for which all non-zero initial conditions settle to, this is in stark contrast to Hooke's law, where different initial conditions give concentric circles in phase space. The Matlab script can be copied into an .m file, allowing the file to run will successively show each method applied to the problem. From this one can get an appreciation for the large running time for some of the algorithms and also experiment with a change the parameters to the problem. Only a few graphs are presented here.

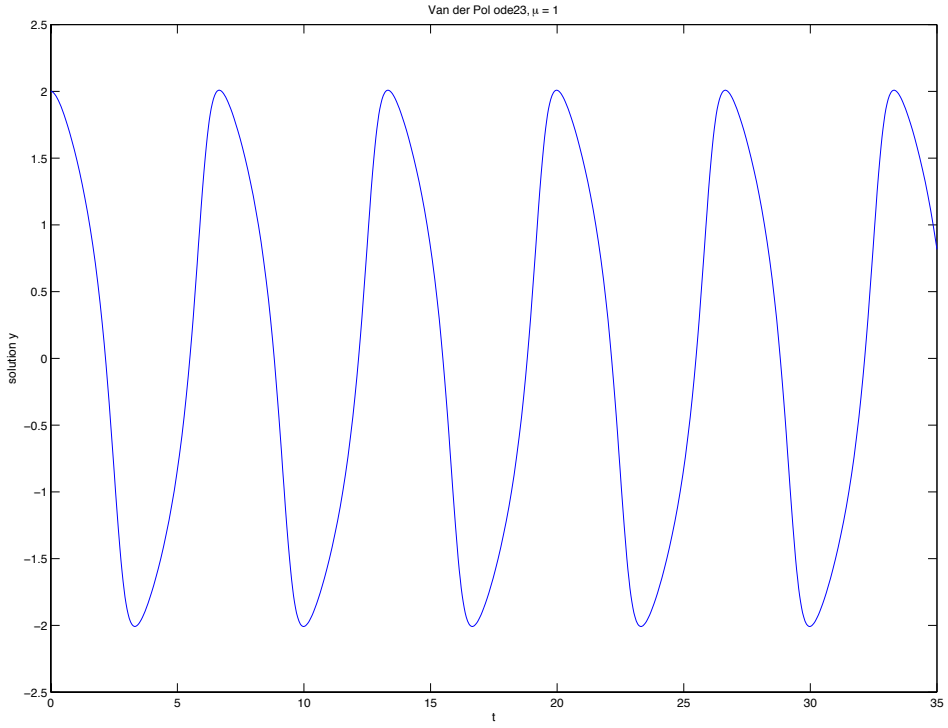


Figure 6: Van der Pol solution by "ode23" with $\mu = 1$.

After writing the equation as a system by introducing the auxiliary variable $v = y'$ and specifying initial conditions as $[2, 0]^T$ (as can be seen in Matlab code). We approximate the solution for small values of μ. The behaviour is not drastically different from the harmonic oscillator. Both Euler's method and RK4 are able to make good local approximations. The solutions should be compared to the plot by the Matlab solver "ode23" which is a combination of second and third order RK methods for which we can implement adaptive step size according to user set tolerance, for the following computations the tolerance is set to 1×10^{-4}.

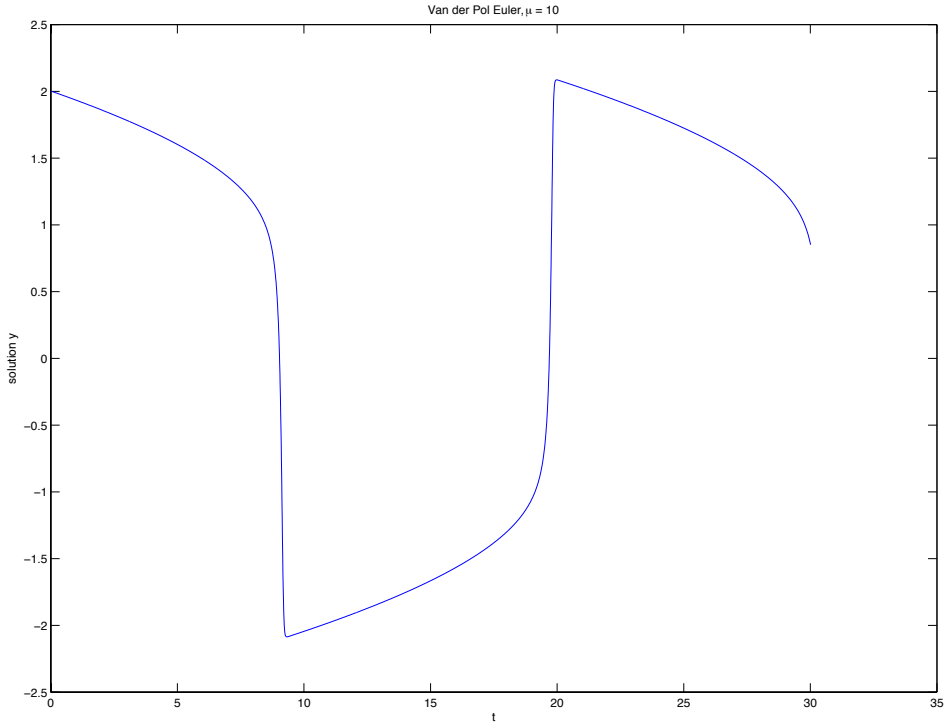


Figure 7: Van der Pol solution by Eulers method with $\mu = 10$, $h = 0.01$.

With $\mu = 10$ the problem becomes more difficult to solve. Euler and RK4 provide decent approximations while being costly since the step size had to decrease by a factor of 10 to provide any solution curve at all, keeping a step size of 0.1 will make a horizontal line which explodes to positive or negative infinity when the equation makes its "transition" from slow to rapid growth. Furthermore larger values of μ makes the period of oscillation longer, and the cost increases due to this as well.

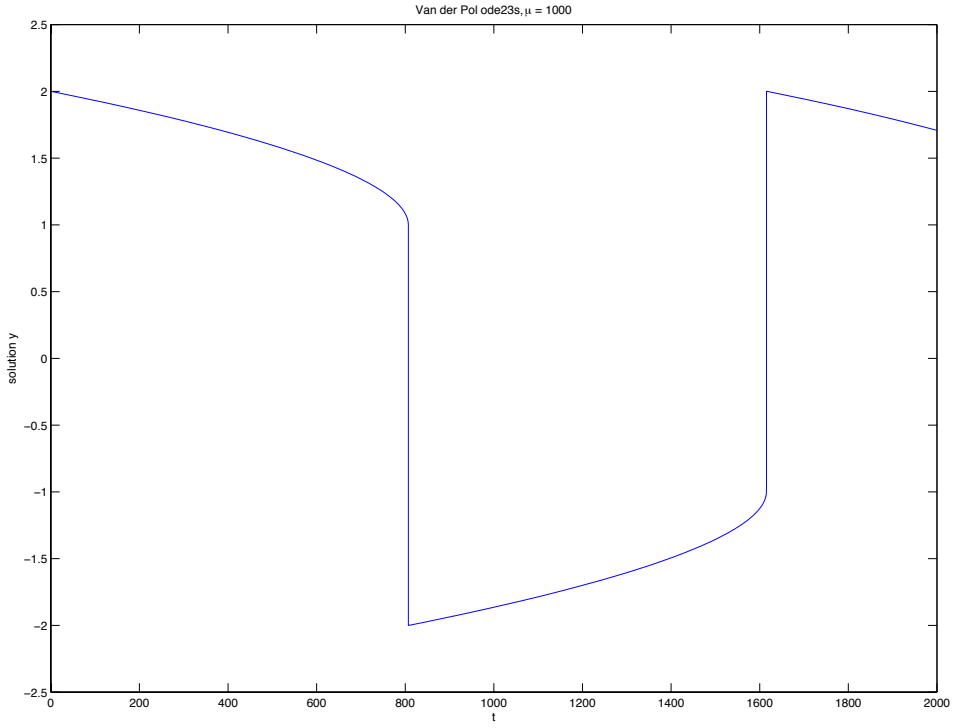


Figure 8: Van der Pol solution by "ode23s" $\mu = 1000$.

For even larger values of μ, the period becomes large and the problem becomes very stiff. None of the constant step size explicit methods provide accurate solutions, however the variable step solver "ode23" with adaptive step size and error tolerance is able to accurately approximate the solution, at extreme expense and running time. The running time for this algorithm on one of the university computers amounts to minutes, and this is only an interval of 1200 seconds. On the other hand to inbuilt matlab solver "ode23s" is designed to handle stiff problems and computes in a few seconds the solution over an interval of 2000 seconds.

```
%% Van der Pol equation
%% Overview
% In this script, we approximate the Van der Pol equation using explicit RK
% methods and built in matlab solvers to illustrate some common
% difficulities in implementation.


%% Script
clear all
close all

%interval of computation
t_beg=0;
t_end=35;
tspan = [t_beg,t_end];

%stepsize
h=0.1;
%initial condition
y0 = [2; 0];
Mu =1;
%Function exists as a matlab demonstration under the name vanderpoldemo.
F = @(t,y) vanderpoldemo(t,y,Mu);

%Error tolerance as specified by user for built in matlab solvers.
tol = 1e-4;

%% Beginning example with mu=1.
% With a small or modest size of the parameter mu, there is little
% difficulity in implementing the methods seen so far, and the methods
% remain accurate.

[t,y] = euler(F, t_beg, t_end, h, y0);

% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
```

```
title('Van der Pol Euler, \mu = 1')


%%
% Plot of RK4
pause
[t,y] = rk4(F, t_beg, t_end, h, y0);

% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol RK4, \mu = 1')

%%
% Using matlab ode23 solver with error tolerance.
pause
opts = odeset('RelTol',tol);
[t,y] = ode23(F,tspan,y0,opts);

% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol ode23, \mu = 1')

%% Example leading to breakdown of methods.
% Letting the parameter Mu=50, we immediately run into difficulities.
%the  problem becomes stiff and the methods employed so far break down. It
%then becomes clear that we must use stiff solvers.

pause
clear Mu;

t_beg=0;
t_end=30;
tspan = [t_beg,t_end];
```

```
%value of Mu
Mu =10;
F = @(t,y) vanderpoldemo(t,y,Mu);
%stepsize
h=0.01;

[t,y] = euler(F, t_beg, t_end, h, y0);

% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol Euler, \mu = 10')

pause

[t,y] = rk4(F, t_beg, t_end, h, y0);

% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol RK4, \mu = 10')

pause

[t,y] = ode23(F, tspan, y0, opts);
% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol ode23, \mu = 10')

pause

[t,y] = ode23s(F, tspan, y0);
% Plot of the solution
plot(t,y(:,1),'-')
```

```
xlabel('t')
ylabel('solution y')
title('Van der Pol ode23s, \mu = 10')

pause

[t,y] = ode113(F, tspan, y0,opts);
% Plot of the solution
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol ode113, \mu = 10')


%%
% Tackling a truly stiff problem, Mu=1000.
pause
clear Mu;

t_beg=0;
t_end=2000;
tspan = [t_beg,t_end];

%value of Mu
Mu =1000;
F = @(t,y) vanderpoldemo(t,y,Mu);

[t,y] = ode23s(F, tspan, y0);
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol ode23s, \mu = 1000')

pause

%using the ode23 solver becomes extremly costly, but is able to produce
%acceptable approximations within our specified error. It is very easy to
%crash matlab using this method, even if the parameters are changed only
```

```
%slightly.

t_beg=0;
t_end=1200;
tspan = [t_beg,t_end];

[t,y] = ode23(F, tspan, y0, opts);
plot(t,y(:,1),'-')
xlabel('t')
ylabel('solution y')
title('Van der Pol ode23, \mu = 1000')
```

References

[1] J. C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20:247–260, 1996.

[2] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 2 edition, 2008.

[3] A. Cayley. On the theory of the analytical forms called trees. *The Philosophical Magazine*, 13, 1857.

[4] B. Chartres and R. Stepleman. A general theory of convergence of numerical methods. *SIAM Journal on Numerical Analysis*, 9(3), 1972.

[5] L. Euler. Institutionum calculi integralis. *Opera Omnia vol XI*, 1, 1768.

[6] Goldstein et al. *Classical Mechanics*. Addison Wesley, 3 edition, 2002.

[7] E. Hairer, , S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, Berlin, 1987.

[8] W. Kutta. Beitrag zur näherungsweisen integration totaler differentialgleichungen. *Z.Math. Phys*, 46:435–453, 1901.

[9] S. Lang. *Introduction to differentiable manifolds*. Wiley, New York, 1 edition, 1962.

[10] A. Lichnerowicz. *Elements of Tensor Calculus*. Methuen, London, 4 edition, 1962.

[11] J. Matousek and J. Nesetril. *Invitation to discrete mathematics*. Oxford University Press, New York, 2 edition, 2008.

[12] J. M. Ortega. The newton kantorovich theorem. *American Mathematical Monthly*, 75(6):658–660, Jun 1968.

[13] C. Runge. Über die numerische auflösung von differentialgleichungen. *Math Ann*, 46:167–178, 1895.

[14] B. Schutz. *A First Course in General Relativity*. Cambridge, 2 edition, 2009.