Optimization of the Runge-Kutta Method for the Steady State Advection-Diffusion Equation

Paulina Ibek

Bachelor's thesis
2023:K12

Lund University

Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Optimization of the Runge-Kutta Method for the Steady State Advection-Diffusion Equation

Paulina Ibek

Supervisor: Philipp Birken

Lund
May 26, 2023

Abstract

In this thesis, the steady state version of the advection-diffusion equation is solved numerically using a Runge-Kutta method. In the advection-diffusion equation, different terms such as the bulk velocity of the fluid, or the diffusion constant must be defined, which are problem dependant. Those are collectively called input parameters. In order for the Runge-Kutta method to converge towards the correct solution, appropriate method parameters in the method must be chosen. The optimal method parameters, for which the method converges the fastest, depend on the specific input parameters in question. For a large set of input parameters, optimal method parameters are found using an optimization technique called particle swarm optimization. An attempt to find a relation between the input parameters and method parameters is made, with the goal of developing a procedure for finding satisfactory method parameters given a specific problem.

Populärvetenskaplig sammfattning

Advektion-diffusion ekvationen beskriver hur partiklar rör sig i flödesströmmar, samtidigt som de sprider ut sig. Den kan användas till att modellera hur föroreningar sprider sig i atmosfären eller i haven, eller för att förstå hur olika vätskor blandas med varandra. Advektion-diffusion ekvationen är svår att lösa för hand, och oftast hittas ungefärliga lösningar med hjälp av datorer istället. En metod som man kan använda sig av heter Runge-Kutta metoden. Det man behöver göra, utöver att skriva in advektion-diffusion ekvationen, är att definiera viktiga egenskaper hos problemet, som exempelvis vätskans hastighet. Runge-Kutta metoden är inte självgående, utan man behöver utöver detta justera vissa inställningar för att den ska kunna lösa problemet. En svårighet som uppstår är att de bästa inställningarna beror på vilket problem som betraktas. Vissa inställningar fungerar utmärkt för vissa flödesprobelm, men kanske misslyckas totalt med att hitta lösningen för andra flödesproblem. Att hitta bra inställningar för Runge-Kutta metoden är inte så tidskrävande om man ska göra det en gång, men ska man lösa många problem blir det väldigt omständligt. Därför vore det fördelaktigt att kunna se direkt på flödesproblemet ungefär vilka inställnigar Runge-Kutta metoden bör ha för att kunna hitta en lösning. Syftet med avhandlingen är alltså att försöka hitta ett samband mellan olika flödesproblem och inställningar i Runge-Kutta metoden, för att underlätta hela problemlösningsprocessen.

Acknowledgements

This thesis marks the last step of my bachelor's degree in mathematics. The past three years have been invaluable to me and I am grateful for the knowledge and experiences gained during my undergraduate studies. I am looking forward to learning even more during my upcoming master's degree.

First and foremost, I would like to thank my supervisor Philipp Birken. Thank you for suggesting such an interesting project, which turned out to be right up my lane. I really enjoyed working with, and combining the different areas of numerical analysis which this project entailed. I would also like to thank you for your invaluable time and feedback, which helped me improve not only the quality of the content of this thesis, but also my academic writing style. Last but not least, I would like to thank you for your patience.

Finally, I would like to thank all my friends and family who supported me throughout my degree.

Contents

Chapter 1

Introduction

The advection-diffusion equation is a partial differential equation that describes the behaviour of a substance that is transported by a fluid and simultaneously undergoes diffusion. It finds applications in various fields and can for instance be used to model the mixing of two fluids or the distribution of substances such as pollutants in the atmosphere. While there are analytical solutions to some forms of the equation, most are too complex and require numerical treatment. One way of solving the steady-state version of the equation is by transforming it to a linear equation system by applying the finite difference method. One can then perform pseudo transient continuation with the help of the Runge-Kutta method. With each iteration, the method moves forward in the fictitious time and closer toward the solution of the equation.

Since the Runge-Kutta method has a couple of in-built parameters, like the size of the time step, one challenge with this approach is tuning those parameters so that the method converges. The optimal, so called method parameters, are problem-specific and depend on various factors such as the velocity of the fluid, the diffusion constant and the resolution of the domain of interest. Hence, they have to be tuned every time a new problem is encountered.

The aim of this thesis is to find the method parameters for a two step Runge-Kutta method using an optimization method called particle swarm optimization across various input parameters in order to solve the steady state version of the advection diffusion equation. The objective is to find a model that can relate the input parameters and the method parameters, which is done using the least squares method. The relation found between the two is used together with the input parameters in order to predict method parameters for which the Runge-Kutta method should converge. The ultimate goal is to develop a strategy for predicting the method parameters for any future problem, thus streamlining the problem solving process.

Chapter 2

Background

2.1 Runge-Kutta method

One of the simplest ways to solve ODE's numerically is the explicit Runge-Kutta method, which can solve initial value problems of the form

$$\frac{d\mathrm{y}}{dt} = \mathbf{f}(t, \mathrm{y}) \qquad \mathrm{y}(0) = \mathrm{y}_0.$$

The equation describes the rate of change of y as a function of the current value of y as well as the current time. An initial value at $\mathrm{y}(t = 0)$ must be defined, otherwise the equation has infinitely many solutions. The idea behind a Runge-Kutta method is that the solution to an ODE at time t can be approximated by evaluating its derivative in different points of the curve in a small time interval. The general formula is given by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i,$$

where $\mathbf{y}_{n+1}$ and $\mathbf{y}_n$ are the solutions at times $n + 1$ and n, respectively, Δt is the time step and $\mathbf{k}_i$ are the evaluated derivatives which are given the weights b_i. The solution to $\mathbf{y}_{n+1}$ is found by taking the solution from the previous time step, and moving along the calculated derivatives, k_i, which are defined as

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\
\mathbf{k}_2 &= \mathbf{f}(t_n + c_2\Delta t, \mathbf{y}_n + (a_{21}\mathbf{k}_1)\Delta t) \\
\mathbf{k}_3 &= \mathbf{f}(t_n + c_3\Delta t, \mathbf{y}_n + (a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2)\Delta t) \\
&\vdots \\
\mathbf{k}_s &= \mathbf{f}(t_n + c_s\Delta t, \mathbf{y}_n + (a_{s1}\mathbf{k}_1 + a_{s2}\mathbf{k}_2 + \cdots + a_{s,s-k}\mathbf{k}_{s-1})\Delta t),
\end{aligned}$$

where c_i are called nodes. The number of points in which one evaluates the derivative varies between the different versions of the method. More accurate results are obtained with small step sizes [1, chap. 12.5] [2, chap. 6]. For a Runge-Kutta method, one must specify the number of stages, that is, the number of derivatives which the method utilizes, as well as the coefficients a, b, and c. Those are usually compiled in a so called Butcher array, see table 2.1.

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s.
\end{array}$$

Table 2.1: Generalized Butcher array for the Runge-Kutta methods.

In this project, a 2-stage Runge-Kutta method will be used, which has the following Butcher array:

$$\begin{array}{c|cc} 0 & & \\ \alpha & \alpha & \\ \hline & 0 & 1, \end{array}$$

Table 2.2: Butcher array for a 2-stage Runge-Kutta.

where the α parameter (as well as the step size, Δt) will be optimized.

After selecting the number of stages of the Runge-Kutta method, one can apply additional constraints in order to achieve a specific order for the method. If a method has the order p, it means that the local truncation error (i.e. the error caused by the approximative nature of the method) is $\mathcal{O}(\Delta t^{p+1})$, where Δt is the step size used. The higher the order, the more conditions must be fulfilled by all the parameters in the butcher array. Since methods with lower stages have very few parameters, it becomes impossible for them to have a large order. To achieve higher order (and hence better accuracy) for the method, it is thus required to add more stages, so that there are enough parameters to fulfill all the requirements that give a higher order. The requirement to have order $p = 1$ is $\sum_{j=1}^{s} b_j = 1$, which we see is fulfilled by the method chosen in this task. For $p = 2$ we must have $\sum_{j=1}^{s} b_j c_j = \frac{1}{2}$ together with the previous requirement. This we see is not fulfilled, but could be if we changed the Butcher array to

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

Table 2.3: Butcher array for a 2-stage Runge-Kutta of order 2.

However, we cannot construct a Runge-Kutta method of stage 2 with an order higher than 2. In fact, the number of conditions required for a certain order increases faster than the order itself

Order	1	2	3	4	5	6	7	8
Conditions	1	2	4	8	17	37	85	200

Table 2.4: Relation between order of a method and the required conditions.

Up until $p = 4$, one can find Runge-Kutta methods such that $s = p$, so the stage can be equal to the desired order. For $p \geq 5$ there are no explicit Runge-Kutta methods of order p such that $p = s$. For $p \geq 7$ the Runge-Kutta method of order p will have at least $s = p + 2$ and for $p \geq 8$ the stage must be at least $s = p + 3$.

Apart from the errors from the numerical methods themselves, there are also round off errors, which arise because the computer performing the calculations must store the values in a limited number of bits. For a 64-bit machine, these errors will be on the order of $\approx 10^{-16}$, which in most cases, is much smaller than the truncation error [3].

2.2 Particle swarm optimization

Definition: *An optimization problem is a problem of determining the best solution from a set of feasible solutions, where the notion of 'best' is defined in terms of an objective function and possibly some inequality and equality constraints.* [4]

Optimization problems involve two main components: an objective function, also known as a fitness function or loss function, and parameters. Broadly speaking, a fitness function is a measure of how well a certain solution performs with regard to an objective. The aim of an optimization problem is to find parameters, P in a parameter space Ω, that minimize (or maximize) the fitness function, f

$$P \in \Omega \subset \mathbb{R}^M : \min f(P), \qquad f : \mathbb{R}^M \to \mathbb{R}. \tag{2.1}$$

Finding such parameters is often a difficult task since the parameter space can be very large. For this purpose, many optimization methods have been developed, one of which is the particle swarm optimization (PSO), which is a population based stochastic optimization technique. The idea behind PSO is that it initializes with multiple so called particles in different points, P_i, in the parameter space, Ω. P_i is commonly referred to as the particle's position, and is simply a set of parameter values. The fitness function, f, is evaluated for all P_i, and the particles are then moved towards a new position using a velocity. The aim is to find a position in the parameter space that minimizes the fitness function. This position, or set of parameter values, is oftentimes referred to as the best position. The number of particles is determined by $N = 10 + 2\sqrt{10}$, where M is the number of dimensions of the fitness function. Both the position and velocity are updated for each iteration and particle according to

$$P_i^{t+1} = P_i^t + V_i^{t+1} \tag{2.2}$$

$$V_i^{t+1} = wV_i^t + c_1 r_1 (P^t_{personal(i)} - P_i^t) + c_2 r_2 (P^t_{global} - P_i^t). \tag{2.3}$$

Equation (2.2) shows the position update for particle i and iteration $t + 1$. Equation (2.3) shows the update of velocity V_i^{t+1}, where w is an inertia term, c_1 and c_2 are called cognitive and social terms respectively and r_1 and r_2 are two random numbers in [0, 1]. $P^t_{personal(i)}$ is each particle's best position, that is, the P_i^t that minimizes f, see equation (2.1). P^t_{global} is the global best best position. The velocity V_i^{t+1} directs each particle i towards its own best position, $P^t_{personal(i)}$ with strength $c_1 r_1$, and the best position of the entire group P^t_{global} with strength $c_2 r_2$. The inertia term w together with the velocity from the previous iteration, V_i^t, keeps the motion from being too erratic.

The inertia term, w, determines how easy it is for a particle the change its velocity. The lower the term, the easier it becomes for the particles to change direction; consequently they become more affected by the best solutions. On the contrary, high inertia terms facilitate the exploration of the search space. However, one should not set the term to a value larger than one, since this will cause the particle to diverge. Similarly, the cognitive and social terms, c_1 and c_2, indicate how much each particle gravitates towards its personal and global best position. With a very high c_1 term, the particles will almost exclusively care about their own solutions, which will most likely not lead to convergence of the method, that is, the particles will not approach the same local minimum. On the other hand, if c_2 is high, the particles will not explore the search space adequately. Finally, r_1 and r_2 adjust the acceleration towards the personal and global best solution stochastically.

For any optimization method it is important to both target the best position in the parameter space, as well as look for new, potentially better solutions. If the particles are strongly drawn to known good solutions, it is quite probable that that they will get stuck in local minima. On the other hand, if the particles explore the search map with little regard to the known best positions, the method will become ineffective, since it would almost be equivalent to testing solutions for randomly chosen parameters. Finding a balance between the inertia, cognitive and social terms is a crucial step for obtaining good results. In general, it can be very difficult to tune them since their performance heavily depends on the shape of the fitness landscape. Luckily, there is a fully automated version of the particle swarm optimization called Fuzzy Self-Tuning Particle Swarm Optimization, or FST-PSO, where the aforementioned terms are dynamically controlled. In essence, each particle has its own terms for inertia, as well as coefficients associated with the social and cognitive factors and the maximum and the minimum velocity. The exact rules that tune the different hyper parameters can be found in the appendix [5].

2.3 Least squares method

A common problem in numerical linear algebra, and regular linear algebra for that matter, is solving the linear equation system $Ax = b$. Overdetermined systems, that is, systems with more equations than unknowns, tend not to have a solution, but this problem can occur in other equations as well. In fact, the equation $Ax = b$ only has a solution iff $b \in$

range(A). With an appropriate $\hat{x}$, however, one could minimize the residual, r, which is defined as

$$A\hat{x} - b = r \neq 0.$$

Since the residual is a vector, a norm must be chosen for measuring its size. With 2-norm, we have the so called least squares problem, as defined in [6, p. 77-78]: *Given* $A \in \mathbb{R}^{m\times n}, m \geq n, b \in \mathbb{R}^m$, *find*

$$\hat{x} \in \mathbb{R} : \min ||b - A\hat{x}||_2. \tag{2.4}$$

As mentioned earlier, $Ax = b$ only has a solution if b lies in the range of A. If it does not, one can project b onto the range of A. If we let P_A be the orthogonal projection from $\mathbb{R}^m$ onto the range of A, we get that $\hat{x}$ is the solution of $Ax = P_A b$. In this case we want $Ax - b \perp$ range(A), which means $A^T(Ax - b) = 0$. Hence, we have that a solution can be found through $A^T A\hat{x} = A^T b$, which are called the normal equations of the least squares problem.

It is important to note that the least square solution must not be unique. In fact, if the matrix A consists of linearly dependent columns, the least square problem has infinitely many solutions. In order for the solution to be unique A has to have full rank. In this case, we have the solution [6, chap. 11]

$$\hat{x} = (A^T A)^{-1} A^T b. \tag{2.5}$$

2.4 Pseudo transient continuation

In general, the solution of a PDE, such as the advection-diffusion equation, might vary with time, and it can sometimes be interesting to study the solution when $t \to \infty$. This is a so called steady state solution of the PDE, and it can be viewed as the solution after the system stabilizes. One method of finding the steady state solution is called pseudo transient continuation (PTC), which finds

$$u* = \lim_{t \to \infty} u(t) \tag{2.6}$$

for equations of the form

$$u_t = F(u), \qquad u(t = 0) = u_0. \tag{2.7}$$

The premise of PTC is that if a steady state solution, $u*$, exists, then the derivative of the solution with respect to time at some point should become zero, $u_t = 0$. This means that one wants to find $F(u) = 0$, which is standard problem in numerical analysis and can be solved using, for instance Newton's method. Newton's method, however, relies on a good initial guess, which is usually not available. Another approach of solving such a problem is by using the Runge-Kutta method, since equation (2.7) is an initial value problem. As the solution stabilizes, the Runge-Kutta method will find u* after sufficiently many iterations [7].

Pseudo transient continuation can be used similarly in solving problems without time dependence on the form of

$$F(u) = 0 \tag{2.8}$$

by introducing a fictitious time. By setting $u_t = F(u) = 0$ and declaring some initial condition $u(t = 0) = u_0$, the solution to the equation can be found using the Runge-Kutta method. Since by construction, there is no time dependence and $u_t = 0$, finding the solution to the original equation (2.8), will be equivalent to finding a steady state solution of an initial value problem. For each iteration, the Runge-Kutta method will find solutions in the fictitious time, which will come closer and closer to $u_t = 0$, and hence the solution to equation (2.8). In this method, the initial condition does not have to be close to the steady state solution, but the better the guess, the fewer iterations needed for finding the solution.

2.5 Stationary linear methods

Definition: *A stationary linear method for solving a linear system $Ax = b$, is an iterative method of the form:*

$$x_{i+1} = Mx_i + N^{-1}b \tag{2.9}$$

with $M, N \in \mathbb{R}^{m\times m}, N$ nonsingular. It is linearly convergent to the solution $A^{-1}b$ iff

$$\rho(M) < 1, \qquad M = I - N^{-1}A \quad [1]. \tag{2.10}$$

The purpose of a stationary linear method is to improve the solution to a linear system through iteration, until it reaches an acceptable level of accuracy. The method involves starting with some initial guess for the solution (x_0), and applying the iterative update rule until it converges to the solution.

Chapter 3

Advection-diffusion equation

The advection-diffusion equation is an equation which combines the equations of advection (also sometimes called convection) and diffusion, into one. Advection is the process of particles (or other physical quantities) moving due to the bulk motion of a fluid. Diffusion in turn, is the movement of particles from high concentrations to low concentrations. The general form of the advection-diffusion equation is

$$\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u) - \nabla \cdot (\vec{v} u) + R, \tag{3.1}$$

where u could be the concentration of the substance of interest, D is the diffusion constant, $\vec{v}$ is the bulk velocity of the fluid as a function of both the spatial and time coordinates, and R describes if there are so called sources or sinks of the studied quantities. To be more precise, a positive R indicates that the measured quantity u is created in the system, while a negative R suggests that the quantity is destroyed [8]. It is also worth mentioning that the diffusion has a unit of m^2/s and usually has a small value. A small molecule like glucose, for instance, has a diffusivity of around $10^{-9}\mathrm{m}^2/\mathrm{s}$ [9] and the diffusivity of carbon dioxide is around $10^{-7}\mathrm{m}^2/\mathrm{s}$ [10]. The left hand side of equation (3.1) describes how the measured quantity u changes over time. The first and second term in the equation to the right hand side describe diffusion and advection respectively. The equation which will be studied in this paper is the steady state advection-diffusion equation, which means that the quantity measured does not change over time, that is $\frac{\partial u}{\partial t} = 0$. Since we also just consider the problem in one dimension, the advection-diffusion equation can be rewritten as

$$v\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2} + R,$$

with the velocity of the bulk v becoming a scalar. By setting $R = 1$ and $b = D/v$ and by using the notation $\frac{\partial u}{\partial x} = u_x$, we obtain the final form of the equation

$$u_x = bu_{xx} + 1. \tag{3.2}$$

In this thesis we will only consider the case $x \in [0, 1]$. Instead of solving equation (3.2) analytically it is treated numerically with the finite difference method. This is done by dividing the domain into N smaller subsections, each of length Δx and approximating the derivative and the second derivative of the function. From this we have

$$u_x(x_i) \approx \frac{u_i - u_{i-1}}{\Delta x} \qquad bu_{xx}(x_i) \approx b\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2},$$

where the index i signifies the point on the grid. Having discretized the equation, one can define

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} \qquad A = \frac{1}{\Delta x}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{pmatrix}$$

and

$$B = \frac{b}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \end{pmatrix} \qquad e = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Equation (3.2) can thus be written as a linear equation system of the form

$$\frac{1}{\Delta x}\begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} = \frac{b}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -2 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \tag{3.3}$$

which simplifies to $(A-B)u = e$ or $e-(A-b)u = 0$. This is an equation on the same form as equation (2.8), which means it can be solved using pseudo transient continuation with the Runge-Kutta method. The initial condition considered in this thesis will be $u(0) = 0$. However, we want to make sure that the method converges towards the solution. Since the Runge-Kutta method in this case serves the purpose of a stationary linear method, one can rewrite one step of the Runge-Kutta method on the form of equation (2.9), and examine if it fulfills the convergence criteria in equation (2.10). A Runge-Kutta update in our case with the 2-step method can be written as

$$u_{i+1} = u_i + \Delta t(b_1 k_1 + b_2 k_2),$$

where b_1 and b_2 are taken from the Butcher array and k_1 and k_2 are the two slopes of the function. From the Butcher array in table 2.2, we have that $b_1 = 0$ and $b_2 = 1$, which simplifies the expression to

$$u_{i+1} = u_i + \Delta t k_2.$$

Using the definition of k_2 this becomes

$$u_{i+1} = u_i + \Delta t \mathbf{f}(u + \alpha \Delta t k_1),$$

where $\mathbf{f}$ is the function $u_t = e - (A - B)u$. Writing out the definition of k_1 as well as the explicit expression for the function $\mathbf{f}$, we obtain

$$u_{i+1} = u_i + \Delta t(e - (A - B)(u_i + \alpha \Delta t(e - (A - B)u_i))).$$

Rearranging the equation we finally obtain the expression

$$u_{i+1} = (I - \Delta t(A - B) + \Delta t(A - B)\alpha \Delta t(A - B))u_i + (\Delta t + \Delta t(A - B)\alpha \Delta t)e.$$

Thus we must show that

$$\rho(I - \Delta t(A - B) + \Delta t(A - B)\alpha \Delta t(A - B)) < 1$$

as well as

$$I - \Delta t(A - B) + \Delta t(A - B)\alpha \Delta t(A - B) = I - (\Delta t + \Delta t(A - B)\alpha \Delta t)(A - B).$$

These equations were examined by plugging them into the code, which will be described in the nextcoming section. There it was found that the criteria were fulfilled in some cases, but not in other, which will be discussed later.

In figure 3.1, the convergence of the Runge-Kutta method toward the solution of one of the discrete points in equation (3.3) is displayed.

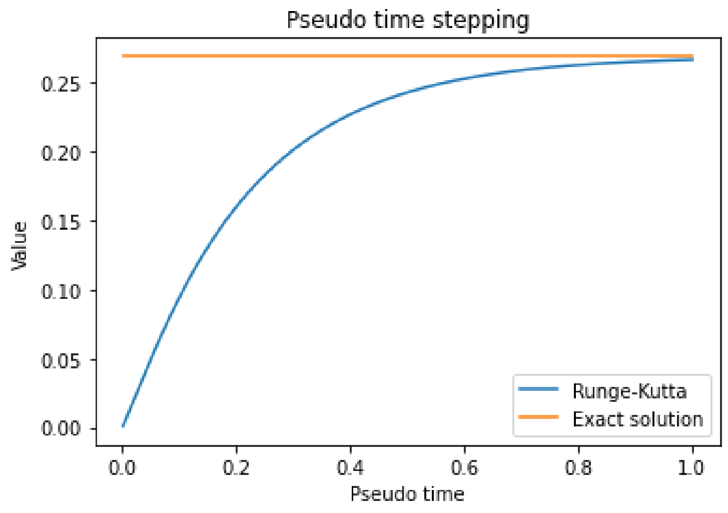
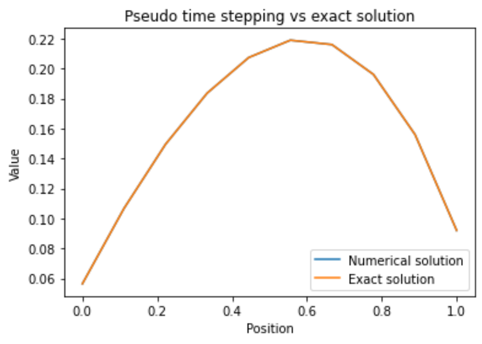
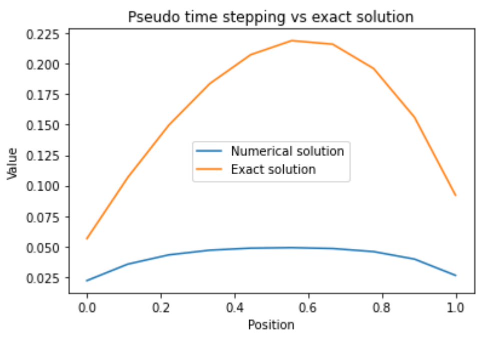


Figure 3.1: Visualization of how the Runge-Kutta method finds the solution to the problem over the fictitious time.

The results from the pseudo transient continuation method can be compared to the exact solution of the linear equation system (3.3). The parameters used in figure 3.2 are $b = 0.5$, $\Delta x = \frac{1}{10}$ and the time step $\Delta t = 0.00005$. The 2-stage Runge-Kutta was used with $\alpha = 0.1$ from the Butcher array in table 2.2. The initial conditions are $u(t = 0) = 0$. The solution becomes more accurate as the number of iterations increases.



(a) Comparison between the pseudo time iteration result with the exact solution after 100000 iterations.



(b) Comparison between the pseudo time iteration result with the exact solution after 1000 iterations.

Figure 3.2: By using pseudo time stepping with $t \rightarrow \infty$, we can obtain the solution to the convection-diffusion problem, with initial conditions 0.

.

Chapter 4

Machine learning step

We have now found that the Runge-Kutta method used as pseudo transient continuation can solve the advection-diffusion equation when it is rewritten as a linear equation system. However, the method does not always converge towards the solution. The convergence depends on finding appropriate values for the method parameters in the Runge-Kutta method

$$\Delta t \qquad \alpha$$

for each pair of problem parameters (which will be referred to as input parameters in this thesis) in the advection-diffusion equation (3.3)

$$\Delta x \qquad b.$$

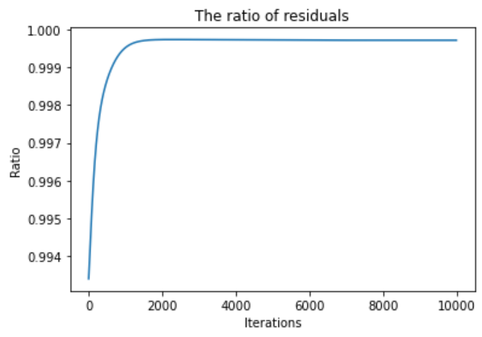
In order to find the method parameters given a pair of input parameters, the fuzzy particle swarm optimization will be used. The objective of the optimization will be not only to find method parameters for which the method converges, but for which the method converges the fastest. Hence, for the optimization, we want to find a fitness function which measures the convergence rate. Even in the case of convergence, the Runge-Kutta must be iterated multiple times before it reaches the solution. This means that equation (3.3) will have a residual, which should decrease with each iteration j

$$r_j = (A - B)u_j - e.$$

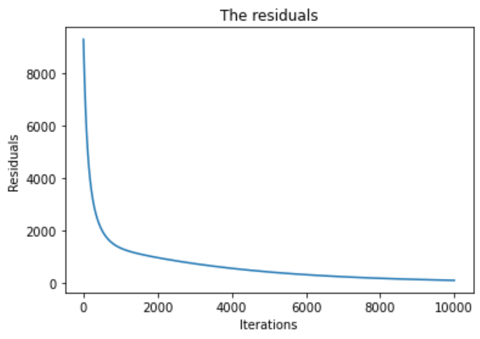
The convergence rate can be measured as a quotient of the residual from the previous and current iteration. If the quotient is low, it means that the residuals decrease fast. Consequently, the method should converge quickly. In this case it is not necessary to iterate the Runge-Kutta method a large number of times. Hence, the fitness function in this problem will be defined as

$$f(\Delta t, \alpha) = \frac{r_{10}}{r_9},$$

which shall be minimized. As the Runge-Kutta method converges, each subsequent residual will shrink at a decreasing rate. In case the method diverges, the quotient will be larger than one. An example of how the quotient of residuals and the residuals themselves can evolve in case of convergence can be seen in figure 4.1.



(a) The residual quotient, or the convergence rate as a function of number of iterations.



(b) The diminishing residual over number of iterations.

Figure 4.1: The residual and convergence rate as a function of the number of iterations. The parameters were $\Delta x = \frac{1}{11}$, $\alpha = 0.1$, $b = 0.5$, $\Delta t = 5 \cdot 10^{-5}$ and max iterations 10000.

The optimization step will be conducted by inserting a pair of input parameters, which define the advection-diffusion problem. The fuzzy particle optimization will search for method parameters in the Runge-Kutta method that minimize the fitness function. The optimization method is run for 100 iteration, after which the optimized method parameters and the best value for the loss function are stored. This process is repeated for multiple pairs of input parameters.

Finally, we want to express the optimized method parameters

$$\begin{pmatrix} \Delta t & \alpha \end{pmatrix}$$

as a function of the input parameters

$$\begin{pmatrix} \Delta x & b \end{pmatrix}.$$

An initial assumption is that there is a linear relation between the two. Thus, the following ansatz is made:

$$\begin{pmatrix} \Delta x & b \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \Delta t & \alpha \end{pmatrix}. \tag{4.1}$$

Having found such a relation, any pair Δx and b can be inserted into the left term of equation (4.1), and ideally obtain parameters Δt and α for which the Runge-Kutta method converges. Having optimised all method parameters, all combinations of Δx and b are put into a matrix, and all values for the optimized parameters Δt and α into their respective vectors. With more than one pair of input parameters and method parameters, the linear equation system (4.1) becomes overdetermined. Since such equations rarely have solutions, the problem can be defined as a least squares problem. As an example, for only two values for each of the two input parameters, the objective would be to find relation vectors

$$\begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} : min \left\| \begin{pmatrix} \Delta t_1 \\ \Delta t_2 \\ \Delta t_3 \\ \Delta t_4 \end{pmatrix} - \begin{pmatrix} \Delta x_1 & b_1 \\ \Delta x_2 & b_1 \\ \Delta x_1 & b_2 \\ \Delta x_2 & b_2 \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix} \right\|_2 \tag{4.2}$$

and

$$\begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} : min \left\| \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} - \begin{pmatrix} \Delta x_1 & b_1 \\ \Delta x_2 & b_1 \\ \Delta x_1 & b_2 \\ \Delta x_2 & b_2 \end{pmatrix} \begin{pmatrix} a_{12} \\ a_{22} \end{pmatrix} \right\|_2 . \tag{4.3}$$

This problem can be solved using for instance equation (2.5).

Having found the relation vectors between the input parameters and the respective method parameters, one can now multiply the input parameters with the relation vectors in order to see what method parameters this produces. The method parameters obtained through these relation vectors will be called calculated method parameters and denoted α_r and Δt_r. In the case that the linear equation system has a solution, α_r and Δt_r should be identical to the method parameters obtained through optimization. Realistically this will not be the case, but if the relation is in fact linear, then the calculated method parameters should be close to the optimised parameters. In order to assess the quality of the linear relation model, one can perform pseudo transient continuation with the calculated method parameters, α_r and Δt_r, and measure the quotient of residuals. If the quotient is less than one for all α_r and Δt_r, the model is satisfactory and can be used to calculate method parameters for other problems as well.

There is a possibility that the relation between the input and method parameters is not linear and that the predictions made by this model will perform poorly. In this case an affine as well as a logarithmic relation will be investigated. An affine model is of the form

$$\begin{pmatrix} \Delta x & b & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} = \begin{pmatrix} \Delta t & \alpha \end{pmatrix} \tag{4.4}$$

and the logarithmic model is

$$\begin{pmatrix} \log(\Delta x) & \log(b) & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} = \begin{pmatrix} \Delta t & \alpha \end{pmatrix}. \tag{4.5}$$

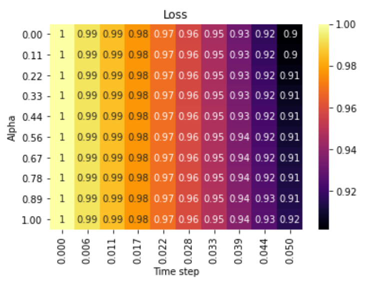
These equations can be separated so that each method parameter is treated individually, in which case they can be turned into least squares problems and solved using equation (2.5).

Chapter 5

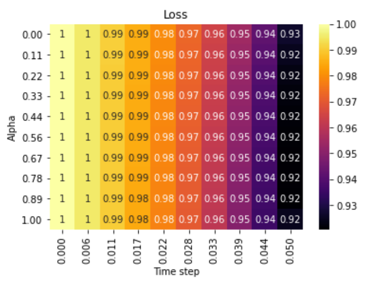
Results

All code produced for this thesis was written in Python. In-built functions were implemented for the fuzzy particle optimization and for solving the least squares problems. The rest of the code was written by the author of this thesis.

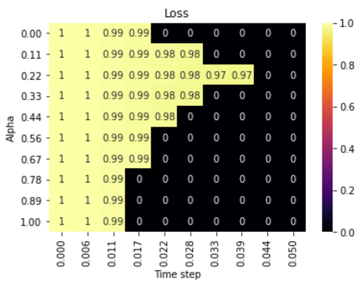
As a first investigation, the fitness function will be examined for a few chosen pairs of input parameters, b and Δx. This is executed by varying the method parameters α and Δt for each pair of input parameters, and displaying the corresponding residual quotient. The method parameters which yield the lowest quotient are the ones that should be found through optimization. The study of the fitness function also helps set reasonable search spaces for the method parameters in the optimization step later on. In this first investigation b was set to 0, 0.5 and 1, and Δx to 1/4, 1/10 and 1/50. In all cases α was varied between 0 and 1 and Δt between 0 and 0.05.



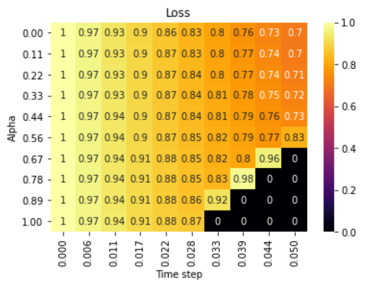
(a) $b = 0,\ \Delta x = 1/4$



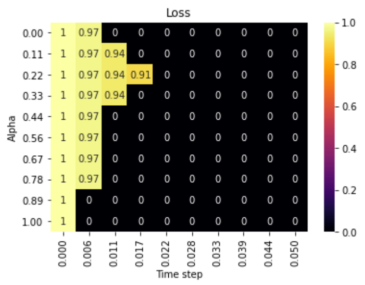
(b) $b = 0,\ \Delta x = 1/10$



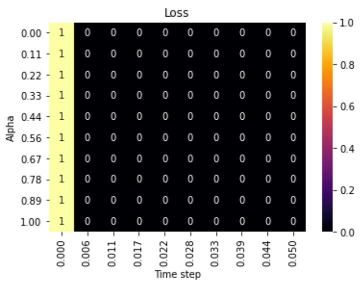
(c) $b = 0,\ \Delta x = 1/50$



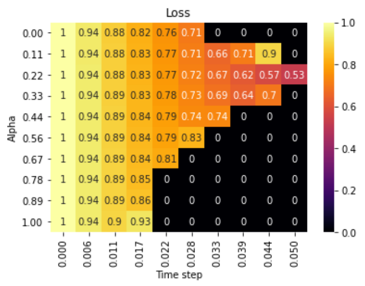
(d) $b = 0.5,\ \Delta x = 1/4$



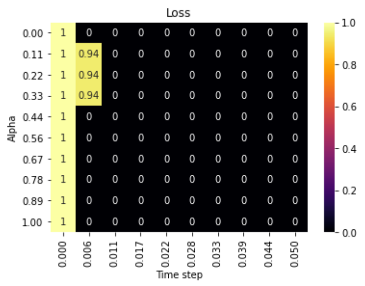
(e) $b = 0.5,\ \Delta x = 1/10$



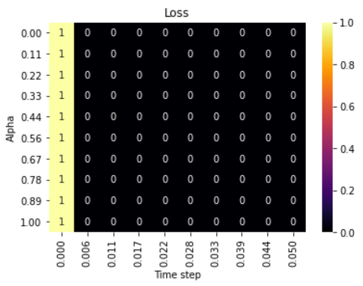
(f) $b = 0.5,\ \Delta x = 1/50$



(g) $b = 1,\ \Delta x = 1/4$



(h) $b = 1,\ \Delta x = 1/10$

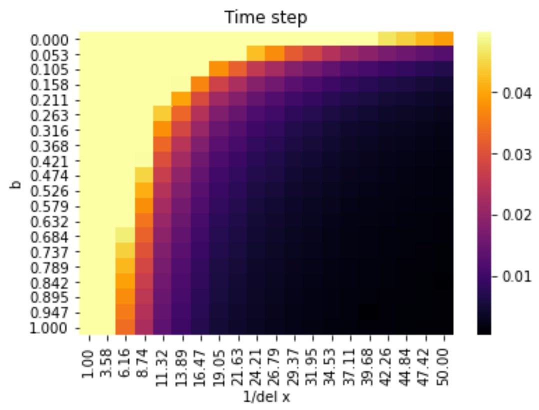


(i) $b = 1,\ \Delta x = 1/50$

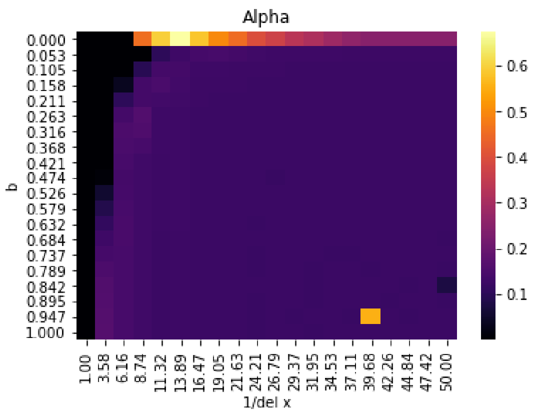
Figure 5.1: Loss function across varying α and Δt for different problems.

The values displayed in the heat maps show the loss function, where all values for which the method diverges (loss values > 1), were set to 0 for clarity. We see that for $b = 0$ and large Δx, the fitness function is smaller than one in all cases, and can reach very small values for specific method parameters, which results in fast convergence of the method. In this case it seems like larger time steps yield better results, and perhaps the best results would be obtained for time steps larger than 0.5, that is values outside of the investigated interval. For larger b, and especially smaller Δx, it is noticeable not only that smaller time steps yield better results, but also that the method diverges for a majority of method parameters. Since we want to find the best method parameters through optimization, and not by looking at the loss function ourselves, it is essential that we choose a parameter space for which the optimization will be effective. By increasing the time step interval we might reach lower loss values for certain problems, but since the search space will be enlarged, it might become difficult to find any good parameters for other problems, due to the convergence area becoming relatively smaller. Since consistent results are preferred in this case, the search intervals will be set to $[0, 1]$ for α and $[0, 0.05]$ for Δt.

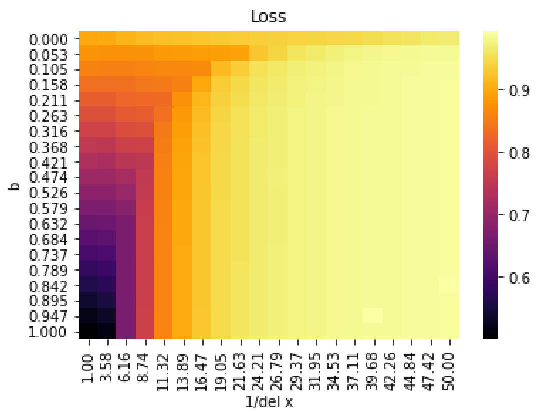
In this next step, optimal method parameters for problems with $b \in [0, 1]$ and $\Delta x \in [1, 1/50]$ are found using particle swarm optimization. The intervals are divided into 20 equal sections each, resulting in 400 problems for which the method parameters are found. The initial condition is $u(t = 0) = 0$.



(a) Optimized time step for different input variables.



(b) Optimized α in the RK method for different input variables.



(c) The value of the fitness function at the end of the optimization for different input variables.

Figure 5.2: Results for the optimization.

In figure 5.2, we notice that with increasing b and decreasing Δx, the optimal time step decreases, which is consistent with the behaviour for the fitness functions observed in figure 5.1. For α we notice a constant behaviour almost throughout all investigated problems. We also see that the fitness function is the smallest for small b and large Δx, which again agrees with the first observation. The largest value observed in the obtained fitness function is 0.9965, which means that for all the found method parameters the method will converge. To verify that the optimization method itself converges to the best value, the fitness value is plotted for $b = 1$ and $\Delta x = 1/4$ throughout the 100 iterations of particle swarm optimization.

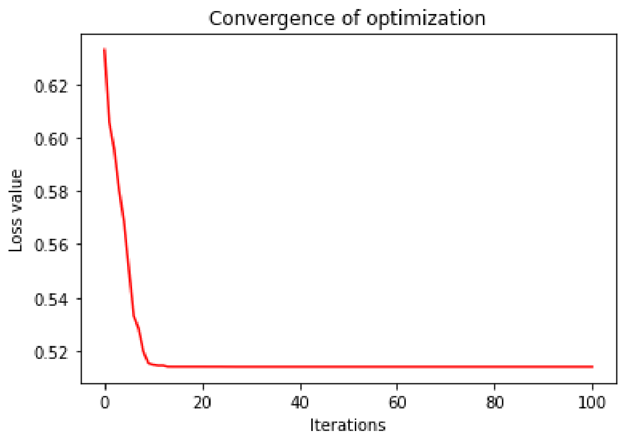
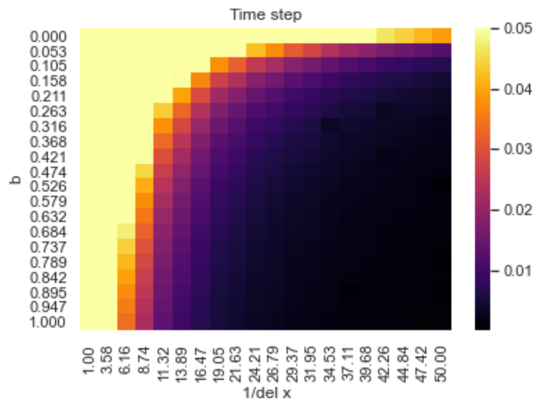


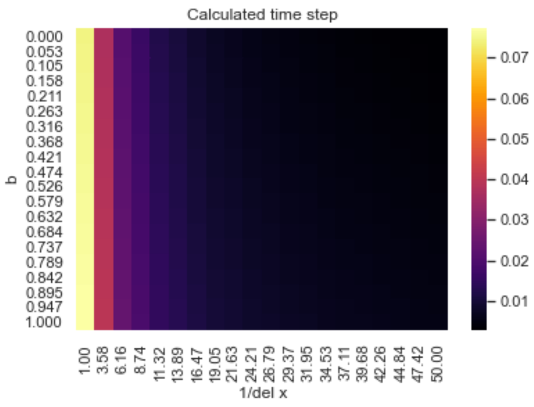
Figure 5.3: Loss value over 100 iterations of PSO

In figure 5.3 we see that the fitness seems to stabilize already after around 10 iterations. One can also compare this optimized value for the fitness function with the optimal value in figure 5.1 (g). The optimized fitness lands at a value of 0.514018, while the optimal value is around 0.53. By increasing the resolution of the fitness function in Figure 5.1 (g) to $200 \cdot 200$, the actual optimal fitness falls to around 0.514018. Therefore, one can conclude that the optimization works very well for this specific problem.

Having optimized the parameters, equation (4.2) and (4.3) will be solved. The same input parameters will then be multiplied with the extracted relation vectors to obtain the calculated method parameters α_r and Δt_r. The idea is that if the relation between the input parameters and the method parameters is linear, this will produce calculated method parameters which are very similar to the optimized ones.



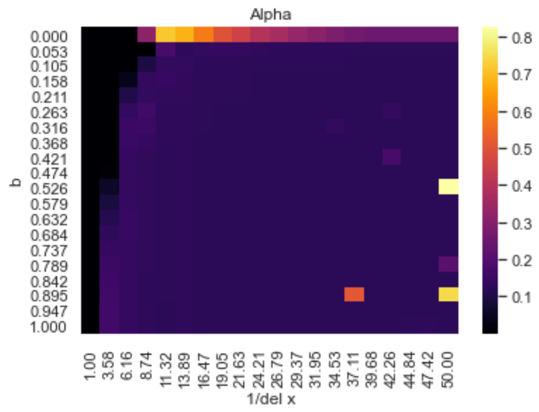
(a) Optimized Δt



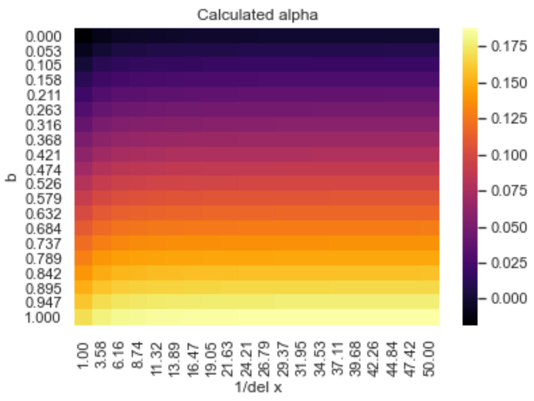
(b) Calculated Δt

Figure 5.4: Comparison between the time step generated by optimization and by the linear relation function.

In figure 5.4 we see that the calculated time step decreases with Δx, which is observed in the optimized time step as well. However, we see very little dependence on b, which is not observed in the optimal case.



(a) Optimized α



(b) Calculated α

Figure 5.5: Comparison between the α generated by optimization and by the linear relation function.

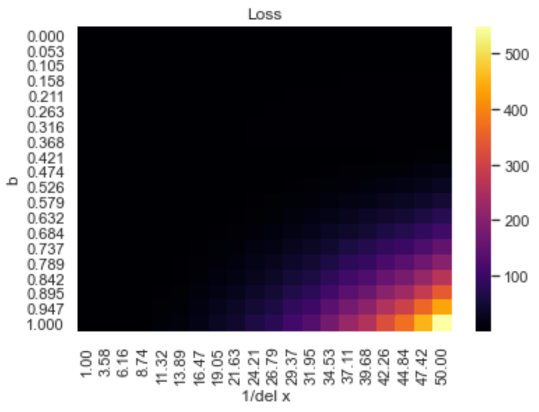
For α we see that the linear relation worked poorly. The vector that related the input parameters to Δt is

$$\begin{pmatrix} a_{11} \\ a_{12} \end{pmatrix} = \begin{pmatrix} 0.1489525 \\ 0.00268776 \end{pmatrix}$$

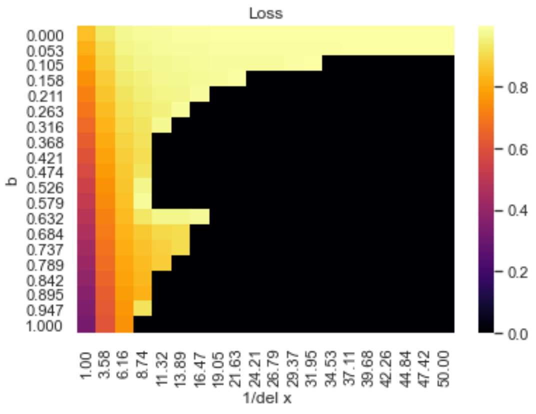
which had the residual $r_{\Delta t} = 0.10464933$ and the vector that related the input parameters to α is

$$\begin{pmatrix} a_{21} \\ a_{22} \end{pmatrix} = \begin{pmatrix} -0.03653204 \\ 0.18822946 \end{pmatrix}$$

with a residual of $r_{alpha} = 5.38207431$. The predicted values were then inserted into the Runge-Kutta method, which was iterated 10 times and the fitness function was evaluated. If all the values in the fitness function are below 1, then the Runge-Kutta method converges for all method parameters found through the linear relation.



(a) Loss function with all values

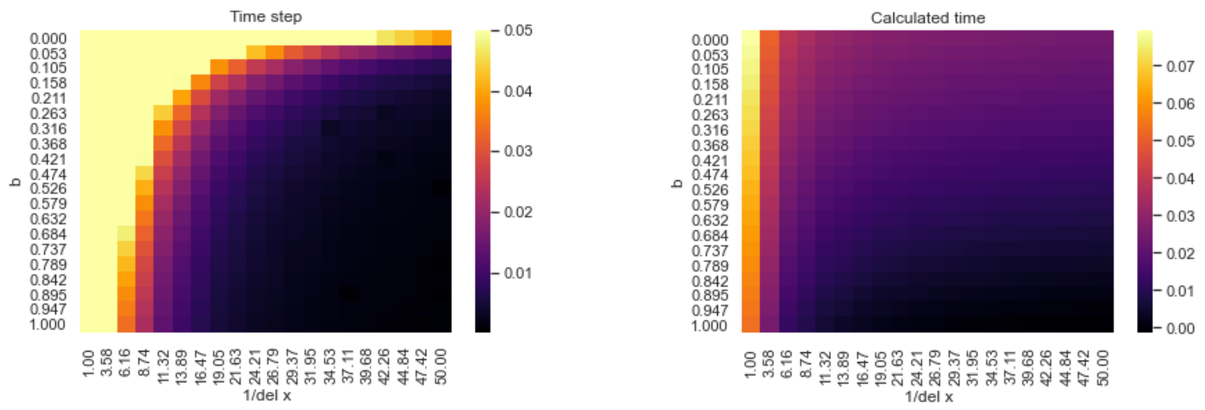


(b) Loss function with converging values

Figure 5.6: Loss function for the calculated method parameters found through the linear relation

Figure 5.6 (a) displays the values for the loss function when using the method parameters that were calculated through the linear relation. In figure 5.6 (b) all diverging results, that is values over 1, were set to 0 in order see clearly for which parameters the method converges. It is quite clear that although the linear relation could be used to produce some good parameters, the results are very bad in most cases.

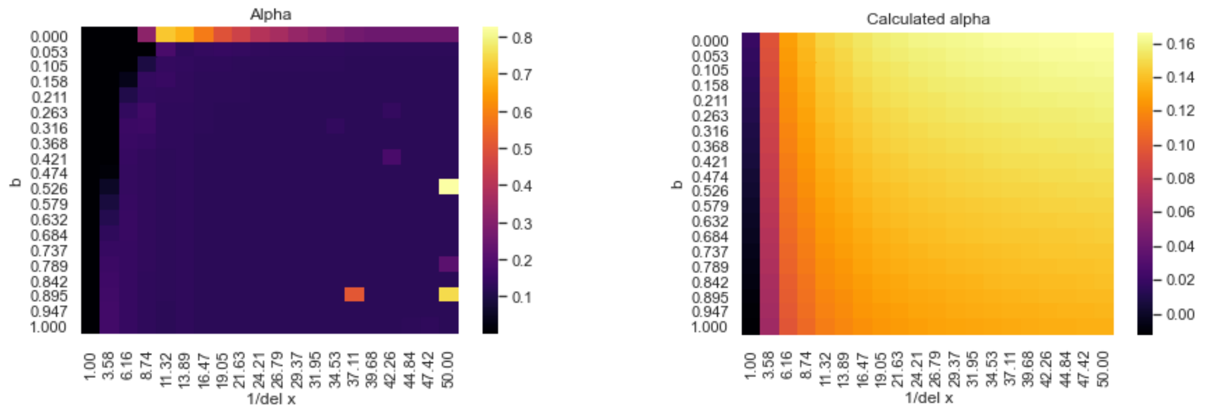
Instead of a linear relation, next an affine relation was assumed. Equation (4.4) will now be solved, again with the intention of comparing the optimized method parameters with the ones found through the relation. Again, we are looking for similarity between the two.



(a) Optimized Δt (b) Calculated Δt

Figure 5.7: Comparison between the time step generated by optimization and by the affine relation function.

With the affine relation we see an improvement from the linear. There is a correct decrease in the size of the time step both with decreasing Δx and increasing b.



(a) Optimized α (b) Calculated α

Figure 5.8: Comparison between the α generated by optimization and by the affine relation function.

For α the overall behaviour looks correct, with small values for the largest value of Δx

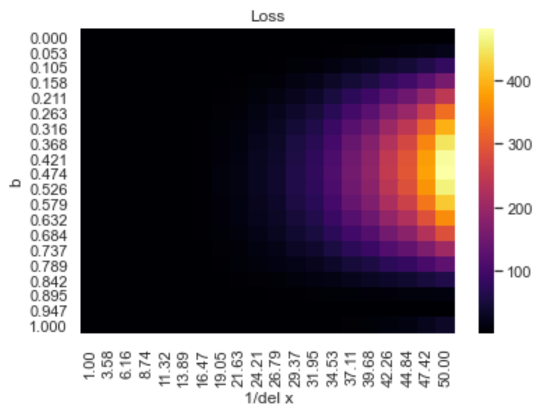
and almost constant value for the rest of the input parameters. However, the calculated α looks shifted, so that the majority of the values are too high. The vector that relates the input parameters to the method parameters for Δt is

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix} = \begin{pmatrix} 0.11398704 \\ -0.02578615 \\ 0.02231511 \end{pmatrix}$$

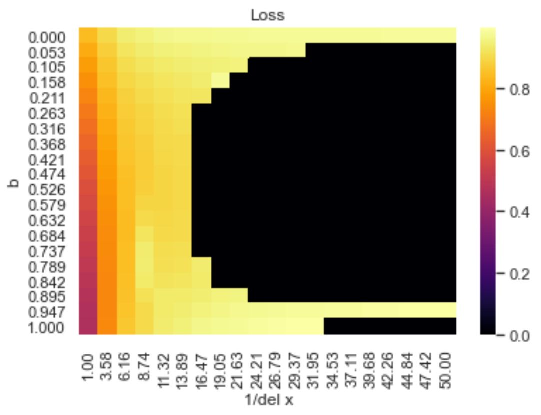
which had the residual $r_{\Delta t} = 0.05783049$ and the vector that related the input parameters to α is

$$\begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \end{pmatrix} = \begin{pmatrix} -0.30632281 \\ -0.03147297 \\ 0.17218164 \end{pmatrix}$$

with a residual of $r_{alpha} = 2.59469461$. We see that the residuals are lower for this relation than for the linear. To test if these method parameters work, again we display the loss function after 10 iterations to see if the method converges for the original problem.



(a) Loss function with all values

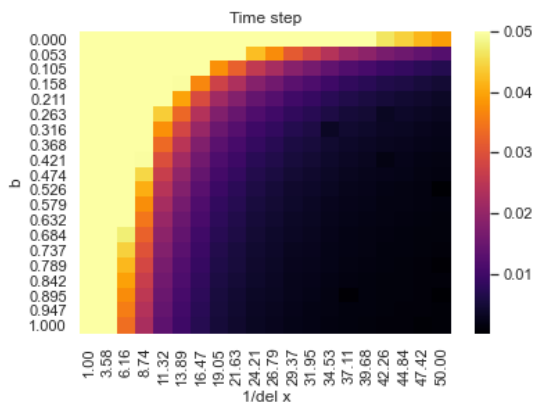


(b) Loss function with converging values

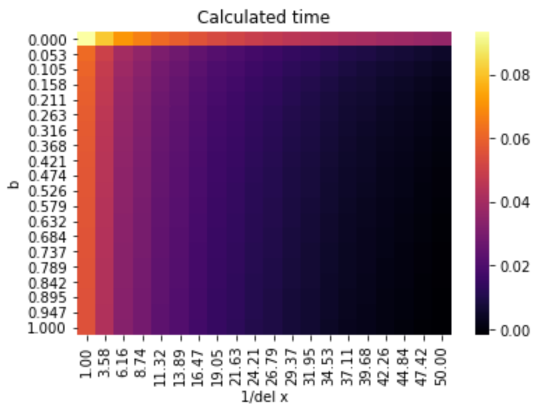
Figure 5.9: Loss function for the calculated method parameters found through the affine relation

Comparing to the linear relation, we notice that the overall highest value in the loss function is lower and occurs for lower values of b. We also see that there are more values for which the method converges.

As a final attempt, the same exact procedure will now be repeated for the logarithmic relation, equation (4.5).

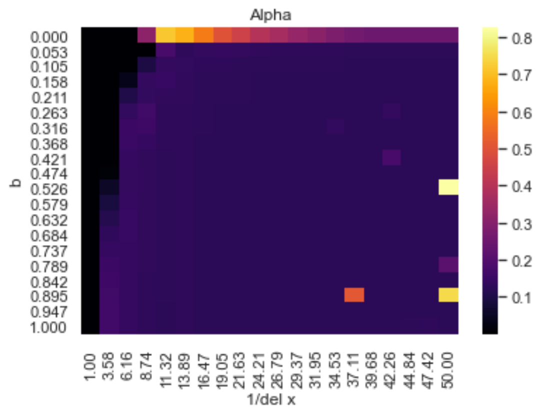


(a) Optimized Δt

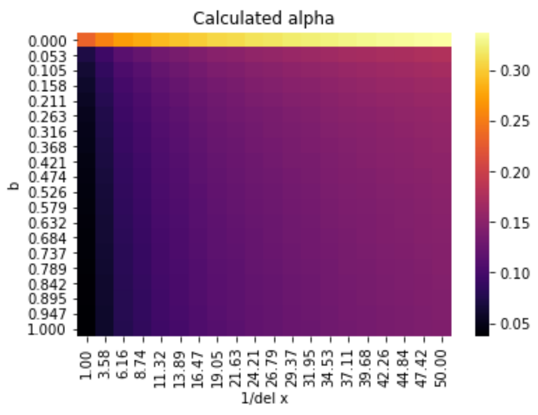


(b) Calculated Δt

Figure 5.10: Comparison between the time step generated by optimization and by the logarithmic relation function.



(a) Optimized α



(b) Calculated α

Figure 5.11: Comparison between the α generated by optimization and by the logarithmic relation function.

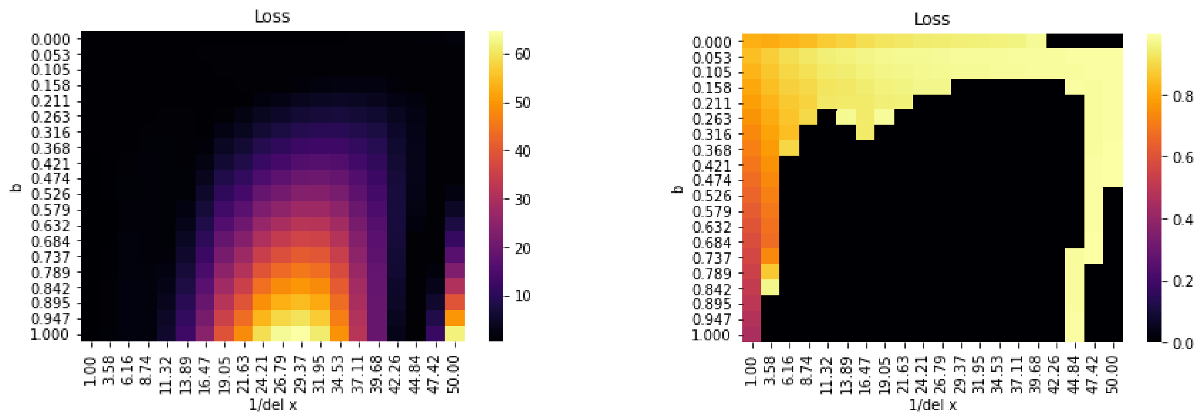
In both figures we see that the calculated parameters behave very similarly to the optimized one. As opposed to the affine relation, the logarithmic seems to describe α better and qualitatively, this looks like the best solution out of the three relations that have been tested so far. The vector that relates the input parameters to the method parameters for Δt is

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix} = \begin{pmatrix} 0.04059826 \\ -0.00547204 \\ 0.06757667 \end{pmatrix}$$

which had the residual $r_{\Delta t} = 0.02800722$ and the vector that related the input parameters to α is

$$\begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \end{pmatrix} = \begin{pmatrix} -0.07490579 \\ -0.0278398 \\ 0.01441217 \end{pmatrix}$$

with a residual of $r_{alpha} = 2.09705821$. Again, the residuals have improved from the affine case. Once more we will study the loss heat map to determine whether this logarithmic relation improves the overall convergence for the original problem.



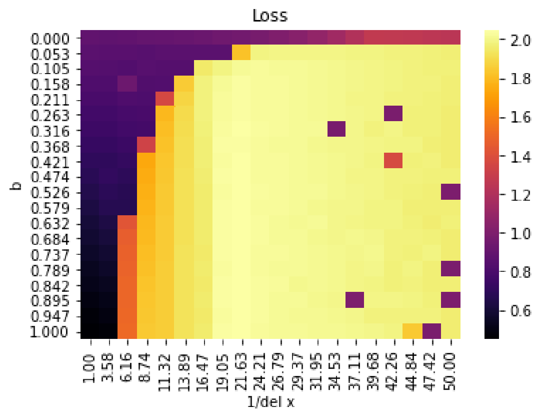
(a) Loss function with all values

(b) Loss function with converging values

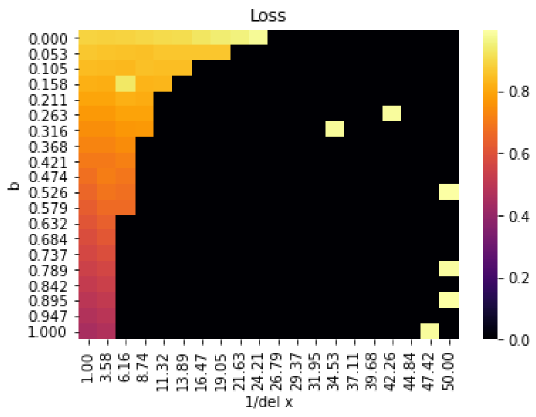
Figure 5.12: Loss function for the calculated method parameters found through the logarithmic relation

The improvement of the residuals is not reflected in the loss functions as seen above, even though the calculated parameters looked similar to the optimized ones. One explanation for the failure of this method can be that the input parameters and method parameters do not have a linear, affine nor logarithmic relation, and are perhaps described by some other function. More functions beside those included in the thesis were tested, and none of them yielded interesting results.

Another possible explanation is that either of the method parameters Δt or α are very sensitive, and/or that the system as a whole is not very robust. This can be assessed through the so called sensitivity analysis and robustness analysis. Sensitivity analysis tests how the studied system responds to small variations in the optimal parameters. The parameters are varied one by one with a small quantity, oftentimes either by increasing or decreasing them by 10% to see how it affects the result. Robustness analysis, on the other hand, measures the impact on the model when all parameters are changed at once, which again is oftentimes done with 10%. For this model sensitivity and robustness analyses can be done by varying the Δt and α and investigating how it effects the loss function. We can start by increasing the time step Δt by 10% and leaving α optimized.



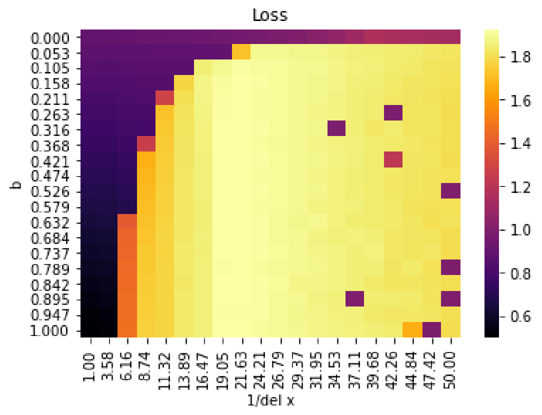
(a) Loss function with all values



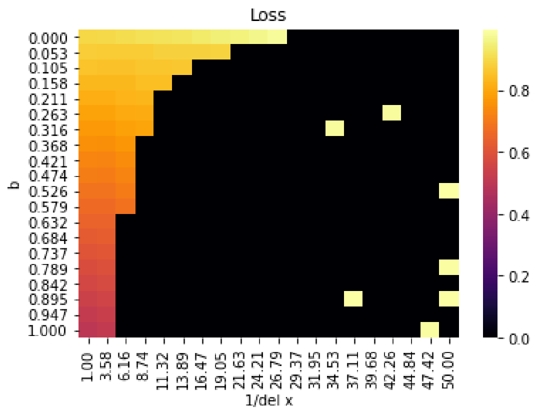
(b) Loss function with converging values

Figure 5.13: Loss function with increased Δt.

In figure 5.13 (a) we see all values in the loss function, while 5.13 (b) displays only the converging values, while all diverging are set to 0 (colored black). The highest value is 2.05 and the lowest 0.45. It is quite clear that changing the time step with just 10% makes a big difference for the convergence- now a vast majority of the method parameters cannot solve the original problem. We see that the method parameters for lower values of b, but primarily for high Δx still perform fairly well. There are also a couple of outliers scattered where the method converges. We can now perform a similar sensitivity analysis for α to see how it affects the loss function.



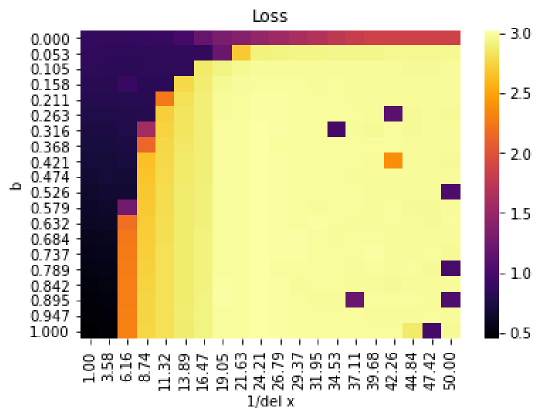
(a) Loss function with all values



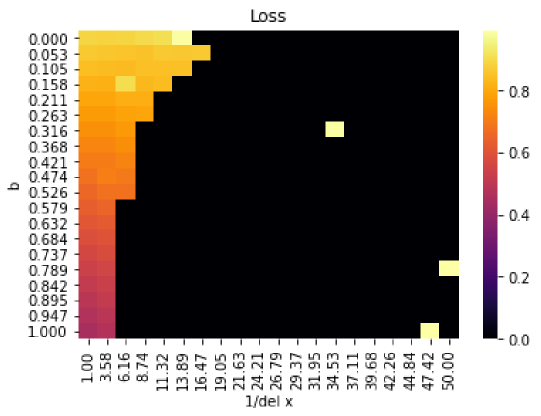
(b) Loss function with converging values

Figure 5.14: Loss function with increased α.

Again, the fitness function has values over one for most of the method parameters and one can conclude that both Δt and α are sensitive to variations. Here the highest loss value is 1.92 and lowest 0.50. Finally a robustness analysis can be performed with both parameters increased by 10%.



(a) Loss function with all values



(b) Loss function with converging values

Figure 5.15: Loss function with increased Δt and α.

Again, the robustness analysis only confirms what the sensitivity analyses have shown- a majority of the heat map diverges. It becomes very clear that even small variations to the parameters prevents the Runge-Kutta method from converging to the right solution.

This is a probable explanation for the method investigated in this thesis not working. Most likely, there is no function that can relate the input parameters and the method parameters accurately enough to obtain the optimal method parameters. This means that whatever calculated method parameters, α_r and Δt_r the relation vectors produce will be an approximation. Since it has been shown that the model is very sensitive to small variations, α_r and Δt_r will not be satisfactory for the method. The behaviour found from the sensitivity and robustness analyses agree with the loss functions found in the initial investigation in figure 5.1. There we saw that for small b and Δx the loss function is below one for essentially all values. Moving slightly in the parameter space did therefore not matter for the convergence of the method. For high b and small Δx, however, there was only a narrow region where the method converged, so small perturbations were expected to have a larger effect on the quality of the method. In order for this procedure of finding method parameters to be effective, one would have to find a function which is very accurate in the sensitive regions, even if it comes at the cost of being less precise in the other regions. Another approach could be to divide the parameter space into subsections and describe them with different relations, which would be particularly useful if it happens that the regions cannot effectively be described by one singular function. It is also a possibility that there are two different models relating α and Δt respectively to the input parameters.

Looking back at the primary heat maps in figure 5.1, it seems obvious that different areas of input parameters would require different search spaces for the method parameters. With that being said, the technique investigated in this thesis seems to work well for larger Δx, even if may not be optimal for that region, and there are definitely more approaches to be explored for finding a successful relation between the input and method parameters that could solve the advection-diffusion equation.

Chapter 6

Summary and conclusions

In this thesis the advection-diffusion equation was converted to a linear equation system using the finite difference method. The linear equation system can be solved with pseudo transient continuation using the 2-step Runge-Kutta method, which has two method parameters. The convergence of the Runge-Kutta method toward the solution depends on the right choice of method parameters, given a pair of input parameters from the advection-diffusion problem. The method parameters which resulted in the fastest convergence of the Runge-Kutta method given a pair of input parameters were found using particle swarm optimization. It was found that the method converges for all problems with the optimized method parameters. Next, a search for a relation between parameters in the Runge-Kutta method and parameters in the advection-diffusion equation was conducted. Linear, affine and logarithmic models were tested, but none consistently produced method parameters for which the Runge-Kutta method converges. Sensitivity and robustness analyses were conducted, which showed that the Runge-Kutta model is very sensitive, and even small perturbations from the optimal method parameters lead to divergence. It was concluded that none of the relations explored in the thesis could explain the relation between the two method parameters and the input parameters. In the future, one could try two different models for the method parameters separately, or test different models for different ranges of input parameters.

Bibliography

[1] Suli, E., Meyers, D. *An Introduction to Numerical Analysis*, Cambridge: Cambridge University Press; 2003

[2] Sauer, T. *Numerical Analysis*, Harlow: Pearson; 2014

[3] Birken, P. *Numerical methods for stiff problems*, Lund; 2022

[4] Bertsekas, D. P. *Nonlinear Programming: Concepts, Algorithms, and Applications*, Belmont: Athena Scientific; 2016

[5] Nobile, M. S., Cazzaniga, P. et al. *Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization*, Swarm and Evolutionary Computation, Volume 39, Pages 70-85 (2018)

[6] Trefethen, L. N., Bau III D. *Numerical Linear Algebra*, Philadelphia: SIAM; 1997

[7] Kelley, C. T., Keyes, D. E. *Convergence Analysis of Pseudo-Transient Continuation*, SIAM Journal on Numerical Analysis, Volume 35, Issue 2 (1998)

[8] Stocker, T. *Introduction to Climate Modeling*, Berlin: Springer; 2011

[9] Koirala, R. P., Dawanse, S., Pantha, N. *Diffusion of glucose in water: A molecular dynamics study*, Journal of Molecular Liquids, Volume 345, article id. 117826 (2022)

[10] Haynes, W. M., Dymond, A. E. *Diffusion Coefficients in Gases*, Journal of Physical and Chemical Reference Data, Volume 34, Pages 1139-1191 (2005)

Appendix

FST PSO

In FST PSO the cognitive term c_1^i, social term c_2^i, inertia w^i and the maximum and minimum velocity are determined for all individual particles, i, with each iteration based on two functions- the distance of the particle to the global best, δ, and the improvement of the fitness compared to the previous iteration ϕ. The maximum velocity is defined as $v_{max}^i = \mathfrak{U} \cdot (P_{max}^i - P_{min}^i)$ and minimum velocity as $v_{min}^i = \mathfrak{L} \cdot (P_{max}^i - P_{min}^i)$, where $[P_{min}^i, P_{max}^i]$ is the interval of the search space for particle i and the so called clamping values must fulfill $\mathfrak{U} \in (0, 1]$, $\mathfrak{L} < \mathfrak{U}$ and $\mathfrak{L} \in (0, 1]$. The 15 rules that govern the update of the terms $c_1^i(t)$, $c_2^i(t)$, $w^i(t)$, $\mathfrak{U}^i(t)$ and $\mathfrak{L}^i(t)$ are summarised in the table below.

Number	**Rule definition**
1	if ϕ worse or δ same $\rightarrow w$ low
2	if ϕ same or δ near $\rightarrow w$ medium
3	if ϕ better or δ far $\rightarrow w$ high
4	if ϕ better or δ near $\rightarrow c_2$ low
5	if ϕ same or δ same $\rightarrow c_2$ medium
6	if ϕ worse or δ far $\rightarrow c_2$ high
7	if δ far $\rightarrow c_1$ low
8	if ϕ worse or ϕ same or δ same or δ near $\rightarrow c_1$ medium
9	if ϕ better $\rightarrow c_1$ high
10	if ϕ same or ϕ better or δ far $\rightarrow \mathfrak{L}$ low
11	if δ same or δ near $\rightarrow \mathfrak{L}$ medium
12	if ϕ worse $\rightarrow \mathfrak{L}$ high
13	if δ same $\rightarrow \mathfrak{U}$ low
14	if ϕ same or ϕ better or δ near $\rightarrow \mathfrak{U}$ medium
15	if ϕ worse or δ far $\rightarrow \mathfrak{U}$ high

Table 6.1: Rules for the updates of the different terms

Low, medium and high values for the respective terms are summarized in the table below.

Variable	**Low**	**Medium**	**High**
w	0.3	0.5	1.0
c_2	1.0	2.0	3.0
c_1	0.1	1.5	3.0
$\mathfrak{L}$	0.0	0.001	0.01
$\mathfrak{U}$	0.1	0.15	0.2

Table 6.2: Examples of values for the variables

[5]

Bachelor’s Theses in Mathematical Sciences 2023:K12
ISSN 1654-6229

LUNFNA-4046-2023

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/