The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Psuedo-Spectral Methods for Multi-Scale Problems

by

Andrew Boyd

Dissertation Presented for the Degree of
MSc in Computational Applied Mathematics

August 2020

Supervised by
Dr Benjamin Goddard

Abstract

Chebyshev pseudo-spectral methods can be used in PDEs using very low amounts of data points, due to their extremely fast convergence rates. The low number of data points becomes problematic when a function is extremely steep in some places, but flat in others. In order to combat this, mappings are used to transform functions into forms that are more readily approximated by low order polynomials, allowing for much higher accuracy with far fewer points. This method is developed, along with methods for determining optimal mappings, and then applied to several PDEs, resulting in much improved convergence, at a cost of more computation. This is shown to be worth it in some of the 1D applications considered here, and would be expected to be far more compelling in higher dimensional problems. This is implemented with code written in MATLAB.

Acknowledgments

I would like to thank Dr. Benjamin Goddard for all his time, expertise and advice over the summer.

I would also like to thank Isabell, Chantal and my mother, Sarah, for their help in proof-reading the following.

Own Work Declaration

I declare that the following work is my own except where otherwise noted.

Contents

1 Introduction **1**

2 Background **3**
2.1 Fundamentals of Spectral Methods . . . 3
2.2 Pseudo-Spectral Groundings & Chebyshev Justification . . . 5
2.3 Chebyshev Differentiation Matrix . . . 7
2.4 Integration . . . 9

3 Mappings **10**
3.1 Maps . . . 10
3.2 Calculating the First and Second Derivative . . . 10
3.2.1 First Derivative . . . 11
3.2.2 Second Derivative . . . 11
3.3 Integration . . . 12

4 Interpolation **14**
4.1 Implementation . . . 15

5 Stationary Functions **15**

6 Methods for Updating the Grid **17**
6.1 Sobolev Norm . . . 17
6.1.1 Normalisation Constants . . . 17
6.2 Updating the grid . . . 19
6.2.1 Alpha Fixed . . . 19
6.2.2 Beta Fixed . . . 20
6.2.3 Both Alpha and Beta Vary . . . 20

7 Moving Exact Solutions **20**
7.1 Moving Tanh . . . 21
7.1.1 Initial Alpha Value . . . 21
7.1.2 Tolerance . . . 21
7.1.3 Range of C . . . 21
7.1.4 Results . . . 21
7.2 Spreading out Gaussian . . . 23
7.2.1 Initial Values . . . 24
7.2.2 Results . . . 24

8 Applications in PDEs **24**
8.1 DAEs . . . 24
8.2 Implementation with MATLAB solvers . . . 26
8.3 Advection Equation with Steep Initial Condition . . . 26
8.3.1 Implementation . . . 27
8.3.2 Parameters . . . 27
8.3.3 Results . . . 27
8.4 Diffusion Equation . . . 29
8.4.1 Parameters . . . 30
8.4.2 Results . . . 30
8.5 Opinion Dynamics Equation . . . 31
8.5.1 Convolutions . . . 33
8.5.2 Parameters . . . 33
8.5.3 Results . . . 34
8.6 Pedestrian Dynamics Equation . . . 36

9 Conclusion **38**

List of Tables

1 A comparison of error, interpolated error, and computational cost of Adaptive and Fixed methods when initial condition has $A = 20$. Adaptive entries for $N > 100$ are not included as no increases in accuracy fail to occur. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time. 29

2 A comparison of the error, interpolated error, and computational time required when the adaptive and fixed grid methods are applied to solve the diffusion equation as defined above, with initial condition $t_0 = 0.001$. Computations for $N > 100$ are not carried out using the adaptive method. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time. 31

3 A comparison of adaptive and fixed methods in calculating the solution of (8.6). Computations for $N > 100$ are not carried out using the adaptive method. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time. 36

List of Figures

2 An exhibition of *spectral convergence*, the errors decrease rapidly until rounding errors dominate. 4

3 The error in spectral derivative for the hat function. Since this function has discontinuities in the derivative, the decay rate is much worse. (Note the y-axis in comparison to Figure 2.) The weighted 2-norm is used since the derivative fails to converge at all in the standard 1-norm used in this project. 4

4 Convergence rates for varyingly smooth functions 5

5 An interpolation of the function $f(x) = \frac{1}{1+16x^2}$ using an equi-spaced grid and a Chebyshev distributed grid. 7

6 A demonstration of the effect of the *tanmap* on a grid of 30 Chebyshev points. . 11

7 The effect of introducing a *tanmap* centred around 0 on the approximation of the spectral derivative of $\tanh(Ax)$. 12

8 The effect of introducing a *tanmap* centered around 0 on the approximation of the second derivative of $\tanh(Ax)$. 13

9 A comparison of the two methods of calculating the second derivative. 13

10 The effect on the error of the first derivative of $tanh(Ax)$. The error given in the colour bar is a $\log_{10}$ scale. 16

11 Error changing as alpha changes and number of points varies. The error given in the colour bar is a $\log_{10}$ scale. 17

12 Error changing in derivative of $e^{-\left(\frac{x}{\sigma}\right)^2}$ as σ and the concentration α are varied. The error given in the colour bar is a $\log_{10}$ scale. 18

13 The effect on accuracy of the *tanmap* when the value of β is offset from the true location of the inflection point. 18

14 A comparison of a solution of the advection equation (see Section 8.3.) With the incorrectly normalised constants, the norm becomes too sensitive, leading to excessive regridding, and thus the solution fails. All other parameters are kept constant. 19

15 Examples of the error of interpolation with correctly and incorrectly tuned tolerance. 22

16 The effect on accuracy of the interpolation of the moving tanh wave when one of the three parameters of N, A and tol are fixed. 23

17 Example error rates of interpolation of a spreading out Gaussian curve. 25

18 The effect on accuracy of varying several sets of parameters. 25

19 Examples of the solution and its changing error over time for fixed and adaptive grids. 26
20 The effect on accuracy of the interpolated solution of the advection equation, and the time taken to calculate this solution, when one of the three parameters of N, A and tol are fixed. Occasional NaN results in inconsistent places seem to be due to numerical coincidences resulting in gridding issues. Error is measured as the maximum of the 1-norm of the error over time. 28
21 A comparison of the solution of the diffusion equation when the premultiplier is used and is not used. 29
22 An example solution of the diffusion equation using the adaptive method and the fixed grid alternative. 30
23 The varying of several parameters and their effect on the accuracy of the interpolated solution of the diffusion equation, as well as the time taken to calculate that solution. Error is measured as the maximum of the 1-norm of the error over time. 32
24 Example solutions of equation (8.6) illustrating the two possible regimes. The 'exact' solution is a fixed grid solution calculated with $N = 500$ points. Note the differing y-axes. 34
25 A comparison of the effects of various parameters on the accuracy of the solution of the Opinion Dynamics equation (8.6). Error is measured as the maximum of the 1-norm of the error over time. 35
26 Examples of the steady states of the FP equation in 1D for $\sigma = 0.5, 0.1, 0.05$ and (left) the influx limited phase - $a = 0.2, b = 0.4$, (center) the outflux limited phase - $a = 0.4, b = 0.2$, and (right) the maximal current phase - $a = 0.9, b = 0.975$. . . 37
27 Solution of (8.11), with $\sigma = 0.05$, $a = 0.5$, $b = 0.6$. The x coordinate of the black dot represents the value β the grid is currently using. 37

1 Introduction

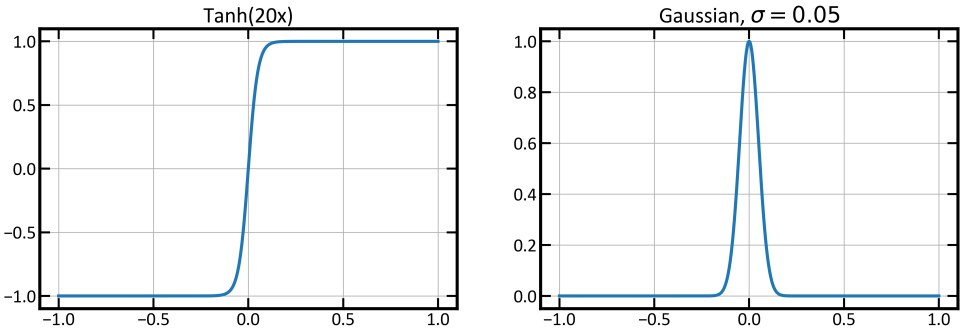
Partial Differential Equations are used to model a large array of physical (and other) phenomena. Spectral Methods are a class of numerical methods for solving differential equations. They are used in the application of Partial Differential Equations (PDEs) frequently due to their extremely fast convergence rates, which is because they exhibit a behaviour known as spectral accuracy. This means that the method is $O(h^{-m})$ for every m, provided that the solution is infinitely differentiable, and is in fact of $O(Ce^{-h})$ if the solution is suitably analytic[1]. They will be applied in this project to calculate spatial derivatives for PDEs, along with a MATLAB ODE solver to advance forward in time. They will be used in conjunction with mappings in order to transform functions that exhibit typically poor performance for spectral methods in order to greatly improve these rates of convergence[2]. The reason spectral methods can exhibit poor performance for very steep functions is due to an effect known as the Runge effect. This is similar to the Gibbs phenomenon observed in Fourier series, and appears as a persistent 'overshoot' in the interpolation that occurs at very quick changes in the derivative. Examples of such functions are shown in Figure 1a, and their transformations are shown in Figure 1b These mappings are used in conjunction with 'regridding' techniques in order to calculate optimal grids that adapt as the solution of a PDE changes, in order to maintain a much improved accuracy level over the traditional fixed grid method. Challenges involved in this project include accurately calculating when to 'regrid', and calculating suitable new grids whenever a 'regrid' is required. This must be carried out without any exact solution, and so the methods involve minimisation of a functional of the solution as it changes over time.

Sections 2 and 2.2 detail important background on the mechanics of spectral methods, and their effectiveness. Section 3 details the different mappings that can be used, and then follows on with the way these mappings can be incorporated into the spectral methods when applied to PDEs. Section 4 details the form of interpolation used in this paper, known as barycentric interpolation, as well as notes on its implementation within this project. Within Section 5, the effectiveness of the mappings introduced in Section 3 are examined on stationary functions, in terms of their accuracies in computing derivatives of steep functions. In Section 6, the method that is used in order to determine when the grid must be updated is laid out, as well as methods that can be used in order to calculate the optimal grid at the update time. This is first tested in Section 7, in which the sensitivity and accuracy of the regridding method are analysed, and various parameters are introduced in order to improve consistency/accuracy. Finally, the developed method is applied to four PDEs in Section 8, and its utility is compared to the classical fixed grid method.

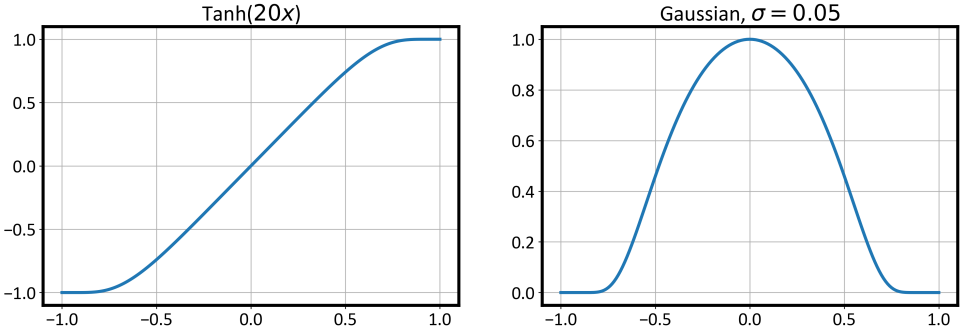
All code used is available at `https://github.com/adfboyd/Psuedo-Spectral_Methods.git`.

[1] Here the traditional definition that $O(f(h))$ means $\leq Cf(h)$ for some C is used.

[2] As will be examined in Section 5, without the mappings, the convergence is still spectral, it merely has a very large constant multiplying the exponential decay.



(a) Functions with extremely steep gradients in a small part of the region considered, and very flat areas in others.



(b) The same functions transformed with appropriate maps into versions that can be more readily interpolated.

2 Background

The following is a relatively brief introduction to spectral methods, largely based on the work found in [18]. If more detail is required, it can be found in [7, 8, 11].

2.1 Fundamentals of Spectral Methods

The simplest application of spectral methods is in the calculation of the derivative of a function, given a set number of points. In numerical methods, this is done by constructing a differentiation matrix. Consider a grid of points x_j, with a function evaluated there as points $u_j = f(x_j)$. The simplest version of this can be obtained by doing a standard second-order finite difference approximation of the derivative:

$$w_j = \frac{u_{j+1} - u_{j-1}}{2h} \tag{2.1}$$

where $w_j \approx f'(x_j)$. In the periodic case, this leads to the matrix vector equation:

$$\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ \vdots \\ w_n \end{pmatrix} = \frac{1}{2h} \times \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & -1 \\ -1 & 0 & 1 & \dots & 0 & 0 \\ 0 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & -1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ \vdots \\ w_n \end{pmatrix} \tag{2.2}$$

This matrix can also be derived through interpolants and derivatives:
For $j = 1, \dots, n$:

- Let p_j be the unique polynomial of degree ≤ 2, such that $p_j(x_j) = u_j$, $p_j(x_{j+1}) = u_{j+1}$, and $p_j(x_{j-1}) = u_{j-1}$.
- Set $w_j = p_j'(x_j)$.

This matrix has order of convergence 2, that is, it is order $O(h^2)$.

This approach can be generalised to polynomials of higher degree that interpolate more points. For example the interpolation resulting from using a polynomial of degree 4, and the 5 points in the neighbourhood of x_j for each j, results in an order of convergence of 4. Spectral methods arrive from asking the question: what happens when we increase this number of points interpolated to its limit? In other words, what happens when we interpolate on a polynomial of degree n for every point?

In this case, the method simplifies to:

- Find p: a function that satisfies $u_j = p(x_j)$, for all j
- Set $w_j = p'(x_j)$

The exact nature of p depends on the problem we are considering. Spectral methods are those that arise when we consider a periodic grid, in which case it makes sense to make p a combination of sines and cosines. On a non-periodic grid, combinations of algebraic polynomials on irregular grids are the natural choice to be used. These are known as Psuedo-Spectral methods, and this is the technique that this project will focus on. When this is done, one finds that the approximations arrived at exhibit a phenomenon known as *spectral accuracy*. An example of the effects of spectral differentiation on two functions, one analytic, and one with discontinuities in the derivative, are shown in Figures 2 and 3. A wider comparison, this time in the accuracies of integration of varyingly smooth functions, is illustrated in Figure 4.

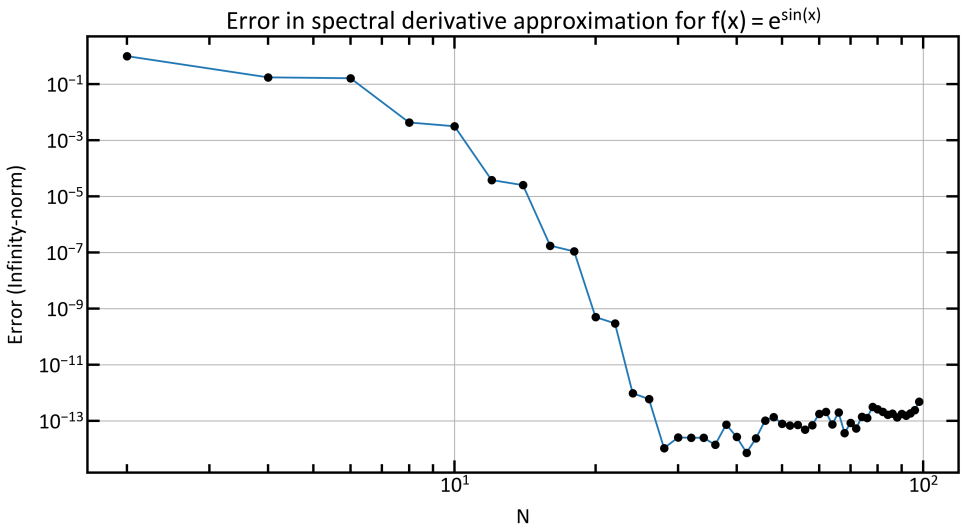


Figure 2: An exhibition of *spectral convergence*, the errors decrease rapidly until rounding errors dominate.

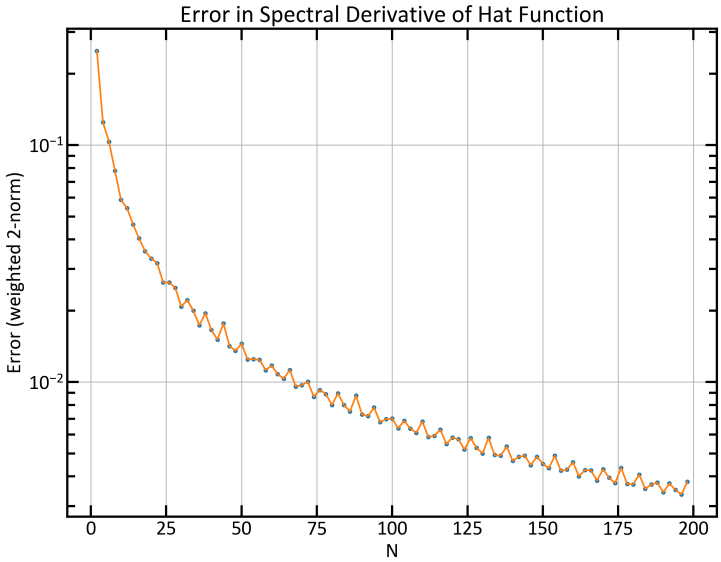


Figure 3: The error in spectral derivative for the hat function. Since this function has discontinuities in the derivative, the decay rate is much worse. (Note the y-axis in comparison to Figure 2.) The weighted 2-norm is used since the derivative fails to converge at all in the standard 1-norm used in this project.

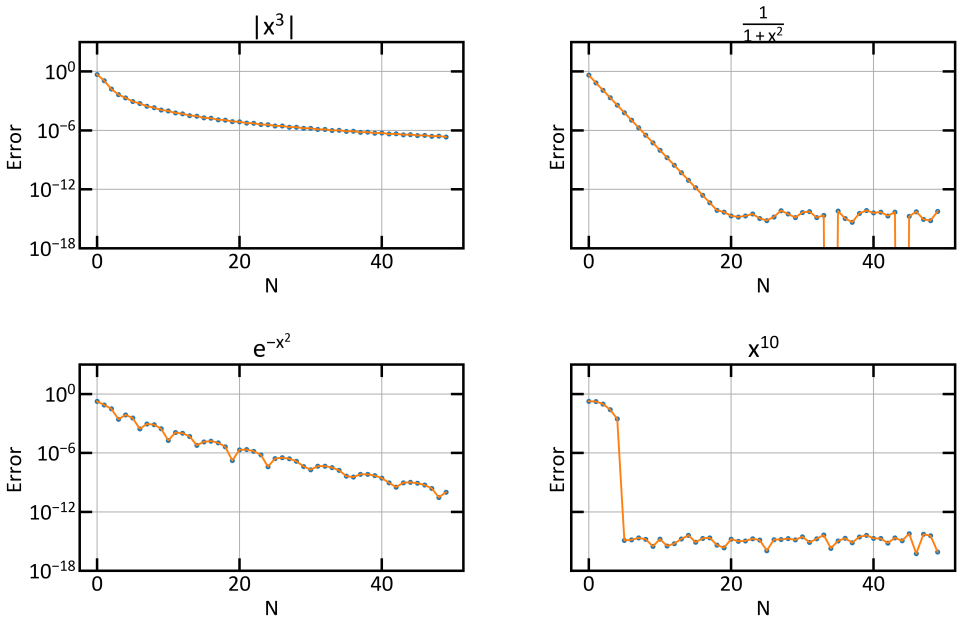


Figure 4: Convergence rates for varyingly smooth functions

2.2 Pseudo-Spectral Groundings & Chebyshev Justification

Based on the procedure laid out above, the goal then is to find the best interpolation that can be used to obtain an accurate representation of a given function.

The case of algebraic interpolation on the interval [-1, 1] is then considered. The goal is to find a way of interpolating a given function such that as we take more points, the error in the interpolation tends to 0. The obvious start is to attempt to interpolate a function by taking n equally spaced points and then interpolating this to a degree $n-1$ polynomial. This, however, is a bad idea, and not only does the error not tend to 0 as $n \to \infty$, the error increases rapidly, at a rate that can be as bad as 2^n. The solution to this is to not pick points that are equally spaced, and instead to pick points which have the following distribution:

$$\lim_{N\to\infty} \text{density} \approx \frac{N}{\pi(1-x^2)}. \tag{2.3}$$

There are several sets of points that satisfy Equation (2.3), the simplest of which are the Chebyshev Points:

$$x_j = -\cos\left(\frac{j\pi}{N}\right) \quad j = 0, 1, \ldots, N, \tag{2.4}$$

where the minus sign is included so that $x_0 = -1$ and $x_N = 1$.

The difference in the accuracy of the two different point sampling methods can be seen in Figure 5.

An explanation for this behaviour is as follows: [18] Suppose we have a monic polynomial p of degree N. This can be written as:

$$p(z) = \prod_{k=1}^{N} (z - z_k). \tag{2.5}$$

Taking the absolute value, this implies:

$$|p(z)| = |\prod_{k=1}^{N} (z - z_k)|. \tag{2.6}$$

And so:

$$\log|p(z)| = \sum_{k=1}^{N} \log|z - z_k|. \tag{2.7}$$

Consider:

$$\phi_N(z) = N^{-1} \sum_{k=1}^{N} \log|z - z_k|. \tag{2.8}$$

Interestingly, ϕ_N now has an interpretation as an electrostatic potential: that is, it represents the potential at z due to charges at z_k which all have potential $N^{-1}\log|z - z_k|$.

Also, we can see that the size of $p(z)$ is related to the size of $\phi_N(z)$ as follows:

$$|p(z)| = e^{N\phi_N(z)}. \tag{2.9}$$

So the size of $p(z)$ is exponentially dependent on the value of $\phi_N(z)$. And further: if $\phi_N(z)$ is approximately constant along [-1,1], then $p(z)$ will be as well. However, if $\phi_N(z)$ varies along [-1, 1], then the variations in $|p(z)|$ will be exponentially bigger and will grow with N. In this framework, it makes sense to take the limit $N \to \infty$, and think in terms of a density function $\rho(x)$, such that $\int_{-1}^{1} \rho(x)dx = 1$. Then the number of grid points in an interval is given by the integral:

$$N \int_a^b \rho(x)dx. \tag{2.10}$$

For finite N, ρ must of course be the sum of Dirac delta functions, but we shall take the limit $N \to \infty$. For equispaced points, this limit is

$$\rho(x) = \frac{1}{2}, \quad x \in [-1, 1]. \tag{2.11}$$

The corresponding potential ϕ is then given by the integral

$$\phi(z) = \int_{-1}^{1} \rho(x) \log|z - x|dx. \tag{2.12}$$

(Note that this is a continuous version of (2.8).) It can be shown from this, that the potential for equispaced points in the limit $N \to \infty$ is:

$$\phi(z) = -1 + \mathrm{Re}\left(\frac{z+1}{2}\log(z+1) + \frac{z-1}{2}\log(z-1)\right), \tag{2.13}$$

where Re(.) denotes the real part. Particularly, $\phi(\pm 1) = -1 + \log 2$, and $\phi(0) = -1$. Therefore, using 2.9, it can be deduced that if a polynomial has roots equally spaced in [-1, 1], then it will take values about 2^N times larger at points near the boundary than at points around the origin.

$$|p(z)| \approx e^{N\phi(x)} = \begin{cases} (2/e)^N & \text{near } x = \pm 1, \\ (1/e)^N & \text{near } x = 0. \end{cases} \tag{2.14}$$

Alternatively, consider the continuous distribution

$$\rho(x) = \frac{1}{\pi\sqrt{1 - x^2}}, \quad x \in [-1, 1]. \tag{2.15}$$

This results in the potential

$$\phi(z) = \log\frac{|z - \sqrt{z^2 - 1}|}{2}. \tag{2.16}$$

The level curves of this potential correspond to ellipses with foci at ± 1, and thus the axes [-1,1], which is a degenerate ellipse (as its minor axis has length 0), is a level curve, with value $-\log 2$.

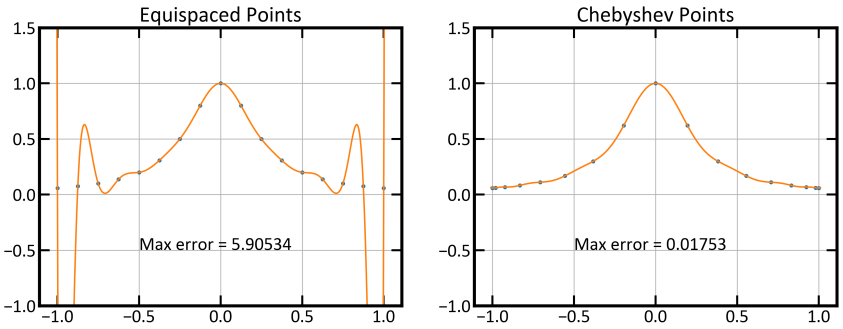


Figure 5: An interpolation of the function $f(x) = \frac{1}{1+16x^2}$ using an equi-spaced grid and a Chebyshev distributed grid.

It can then be concluded that if a monic polynomial p has N roots spaced according to the Chebyshev distribution in [-1, 1], then it will oscillate between values of comparable size on the order of 2^{-N} throughout [-1,1]:

$$|p(x)| \approx e^{N\phi(x)}, \quad x \in [-1, 1]. \tag{2.17}$$

2.3 Chebyshev Differentiation Matrix

Suppose the method outlined in Section 2.1 is now used on a Chebyshev grid. That is, given a function f defined on a set of Chebyshev points of size $N+1$, obtain the discrete derivative of f as follows:

- Let p be the unique polynomial of degree $\leq N$ such that $p(x_j) = f_j$ for $j = 0, \ldots N$.
- Set $w_j = p'(x_j)$.

This operation is linear, and thus it can be represented by a linear operator, i.e. an $(N+1) \times (N+1)$ matrix. Let this be represented by D_N. So:

$$\mathbf{w} = D_N \mathbf{f}. \tag{2.18}$$

Before the general case, consider the cases $N = 1$ and $N = 2$ first. If $N = 1$, then the interpolating points are $x = \pm 1$, and the interpolating polynomial is:

$$p(x) = \frac{1}{2}(1-x)f_0 + \frac{1}{2}(1+x)f_1. \tag{2.19}$$

Taking the derivative gives:

$$p'(x) = -\frac{1}{2}f_0 + \frac{1}{2}f_1. \tag{2.20}$$

This is independent of x, and so the columns of D_N are the same, and we can write

$$D_1 = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}. \tag{2.21}$$

If $N = 2$, then the interpolating points are $x = \pm 1, 0$, and the interpolating polynomial is:

$$p(x) = \frac{1}{2}x(x-1)f_0 + (1-x)(1+x)f_1 + \frac{1}{2}x(1+x)f_2. \tag{2.22}$$

Differentiating gives:

$$p'(x) = \left(x - \frac{1}{2}\right) f_0 - 2x f_1 + \left(x + \frac{1}{2}\right) f_2. \tag{2.23}$$

The differentiation matrix can therefore be obtained by evaluating this function at each of the interpolation points, each respective value of $x = -1, 0, 1$ corresponding to a different column. Thus:

$$D_2 = \begin{pmatrix} -\frac{3}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & -2 & \frac{3}{2} \end{pmatrix}. \tag{2.24}$$

The general case is as follows, and was first published in [10]: For each $N \geq 1$, let the rows of the $(N+1) \times (N+1)$ Chebyshev differentiation matrix D_N be indexed from 0 to N. Also, let

$$p_j(x) = \frac{1}{a_j} \prod_{\substack{k=0, \\ k \neq j}}^{k=N} (x - x_k), \qquad a_j = \prod_{\substack{k=0, \\ k \neq j}}^{k=N} (x_j - x_k). \tag{2.25}$$

This means that p_j represents a function that has values $p(x_j) = 1$, and $p(x_i) = 0$, for all $i \neq j$. Then the entries $(D_N)_{ij}$ correspond to the value of $p_j'(x_i)$. To obtain p_j', first take the log of (2.25), and take the derivative, giving

$$p'(x) = p(x) \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x - x_k}. \tag{2.26}$$

If $i = j$, then

$$p_j'(x_i) = p_j'(x_j) \tag{2.27}$$

$$= p_j(x_j) \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x_j - x_k} \tag{2.28}$$

$$= \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x_j} = D_{jj}. \tag{2.29}$$

If $i \neq j$, then

$$p_j'(x_i) = p_j(x_i) \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x_i - x_k} \tag{2.30}$$

$$= \frac{1}{a_j} \prod_{\substack{k=0, \\ k \neq j}}^{k=N} (x_i - x_k) \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x_i - x_k} \tag{2.31}$$

$$= \frac{1}{a_j} \left(\prod_{\substack{k=0, \\ k \neq i,j}}^{k=N} (x_i - x_k) \right) (x_i - x_i) \sum_{\substack{k=0, \\ k \neq j}}^{k=N} \frac{1}{x_i - x_k}. \tag{2.32}$$

Now, the product term is distributed across each of the terms within the sum, and for all but the term within the sum in which $i = j$, the multiplication by $(x_i - x_i)$ results in the term being

0. However, one part of the sum has denominator $x_i - x_i$ too, and these cancel, leading to:

$$D_{ij} = p_j'(x_i) \tag{2.33}$$

$$= \frac{1}{a_j} \prod_{\substack{k=0,\\ k\neq i,j}}^{k=N} (x_i - x_k) \tag{2.34}$$

$$= \frac{a_i}{a_j(x_i - x_j)}. \tag{2.35}$$

It has been shown in [10] that these results are equivalent to the following:

$$\begin{aligned} (D_N)_{00} &= -\frac{2N^2+1}{6}, \quad (D_N)_{NN} = \frac{2N^2+1}{6}, \\ (D_N)_{jj} &= \frac{-x_j}{2(1-x_j^2)}, \quad j = 1, \ldots, N-1, \\ (D_N)_{ij} &= \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(x_i - x_j)}, \quad i \neq j, \quad i,j = 0, \ldots, N, \end{aligned} \tag{2.36}$$

where:

$$c_i = \begin{cases} 2 & i = 0 \text{ or } N, \\ 1 & \text{otherwise.} \end{cases}$$

2.4 Integration

Calculating the integral of a function can also be done in a spectral way. Let $f(x)$ be a function evaluated on a grid with values f_x. Suppose the integral of this function is desired, i.e.:

$$I = \int_{-1}^{1} f(x)dx. \tag{2.37}$$

This can be thought of as a particular case of ODE, namely:

$$u'(x) = f(x), \quad u(-1) = 0, x > -1. \tag{2.38}$$

This can therefore be set up as a spectral method on [-1,1] with a Chebyshev grid. In order to impose the boundary condition, simply strip off the first row and column of the differentiation matrix D, and let the resulting matrix be called $\tilde{D}$. Then the resulting system of linear equations is:

$$\tilde{D}\mathbf{u} = \mathbf{f}, \tag{2.39}$$

where $\mathbf{f} = (f(x_0), f(x_1), \ldots f(x_{N-1}))$. In fact, the vector $\mathbf{u}$ gives the integral of the function $f(x)$ at each respective grid point, and all that is required is the last component, u_{N-1}. This means that only the first row of $\tilde{D}$, let this be called w, need be calculated. Then an equivalent equation is

$$I = u_{N-1} = w^T\mathbf{f}. \tag{2.40}$$

This is the method that was implemented whenever numerical integration was called for in this project, and is specifically used for calculating the Sobolev Norms explained in Section 6.1. It should be noted that in practice, the vector w^T was appended with an extra entry of value 0, simply so that the calculation of the integral could be done using the entire vector of $\mathbf{f}$.

3 Mappings

While Chebyshev methods do guarantee spectral accuracy for all analytic functions, the coefficient of this accuracy can be extremely high in certain cases, resulting in still very poor convergence rates, and large numbers of points being required. This happens specifically in cases where the function being studied exhibits a rapid change in a small area, such as a steep wave or Gaussian. (See Figure 1a).

One way to deal with this challenge is to use maps to alter the positions of the gridpoints, so that they are clustered around the region of interest. This alternatively can be thought of as transforming the function so that it can be more readily approximated by lower order polynomials [2, 3].

3.1 Maps

The maps from this section are based on the corresponding sections within [4] and [15]. Much greater detail on possible mappings can be found within [13].

There are several maps that can be used to transform from the interval [-1,1] to an arbitrary interval $[a, b]$.

The simplest one is simply a linear map:

$$f(s, a, b) = \frac{b-a}{2}s + \frac{a+b}{2} = x. \tag{3.1}$$

This can be composed with the following maps, which map from [-1,1] to [-1,1], to give the same arbitrary interval.

To map points closer to the center of the interval, a *center* map can be used. This has the form:

$$f(s, \gamma) = (1-\gamma)s^3 + \gamma s = x, \tag{3.2}$$

where $0 < \gamma < 1$ indicates how strongly the points are concentrated around 0.

The standard Chebyshev distribution clusters points around the boundary of the interval. To cancel this out, the following map can be used:

$$f(s, \alpha) = \frac{\sin^{-1}(\alpha s)}{\sin^{-1}(\alpha)}, \tag{3.3}$$

where $0 < \alpha < 1$ varies the amount of spreading out, and as $\alpha \to 0$, the map approaches a linear map.

For the purposes of this project, a more flexible mapping is required. Thus we introduce the *tanmap* [4]:

$$f(s, \alpha, \beta) = \beta + \frac{\tan(\lambda(x - s_0))}{c}, \tag{3.4}$$

where $\lambda = \frac{\arctan(\alpha(1-\beta))}{1-s_0}$, $s_0 = \frac{k-1}{k+1}$, and $k = \frac{\arctan(\alpha(1+\beta))}{\arctan(\alpha(1-\beta))}$. Here $\alpha > 0$ represents the concentration of the points around the value $\beta \in [-1, 1]$. (As $\alpha \to 0$, the *tanmap* approaches the identity.)

An illustration of the effect of the *tanmap* on 30 Chebyshev points is illustrated in Figure 6. This mapping, in conjunction with a linear map, will be used extensively throughout the remainder of the project due to its ease of adaptability and flexibility.

3.2 Calculating the First and Second Derivative

These maps can be used in conjunction with Chebyshev spectral methods in the following way:

- Transform the function and coordinates according to the mapping used.

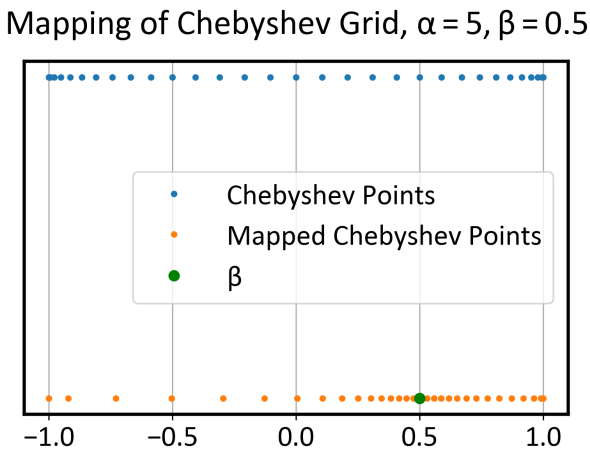


Figure 6: A demonstration of the effect of the *tanmap* on a grid of 30 Chebyshev points.

- Calculate the appropriate (transformed) differentiation matrix, $\mathbf{D}_{map}$, for the mapping used.
- Calculate the derivative, $\mathbf{D}_{map}f(\mathbf{x})$.

The first step is accomplished by taking the mapping of Chebyshev points, then evaluating these points on the function. The next step requires a composition of the original Chebyshev Differentiation matrix, $\mathbf{D}$, and the derivatives of the map being used. This method can of course be extended for higher order derivatives, but this was not necessary for this project.

3.2.1 First Derivative

Applying the matrix D takes the derivative of the function evaluated at the transformed Chebyshev grid. If f is the function we wish to evaluate, with values f_j on the grid x_j, and our mapping is h. Then $Df(\mathbf{x})$ represents:

$$\frac{d(f(h(x))}{dx} = f'(h(x)) \cdot h'(x), \tag{3.5}$$

where the right hand side follows from the chain rule. Rearranging, we get:

$$f'(h(x)) = \frac{d(f(h(x))}{dx} \cdot \big(h'(x)\big)^{-1}. \tag{3.6}$$

So, for the first derivative,

$$D_{map} = D \cdot \mathrm{diag}(1/Dx), \tag{3.7}$$

where $\mathrm{diag}(1/Dx)$ is a diagonal matrix with entries corresponding to the inverse of the derivative of the mapping function at each of the points x_j. Note that diagonal matrices commute with symmetric matrices such as $\mathbf{B}$, so this equation can equally be written as

$$D_{map} = \mathrm{diag}(1/Dx) \cdot D. \tag{3.8}$$

3.2.2 Second Derivative

In the case of the second derivative, we can either simply multiply by the matrix found in Section 3.2.1 twice, or we can apply the method used above again. In this case, we find that:

$$\frac{d^2(f(h(x))}{dx^2} = \big(f''(h(x)) \cdot h'(x)\big) \cdot h'(x) + f'(h(x)) \cdot h''(x), \tag{3.9}$$

using the product rule and the chain rule.

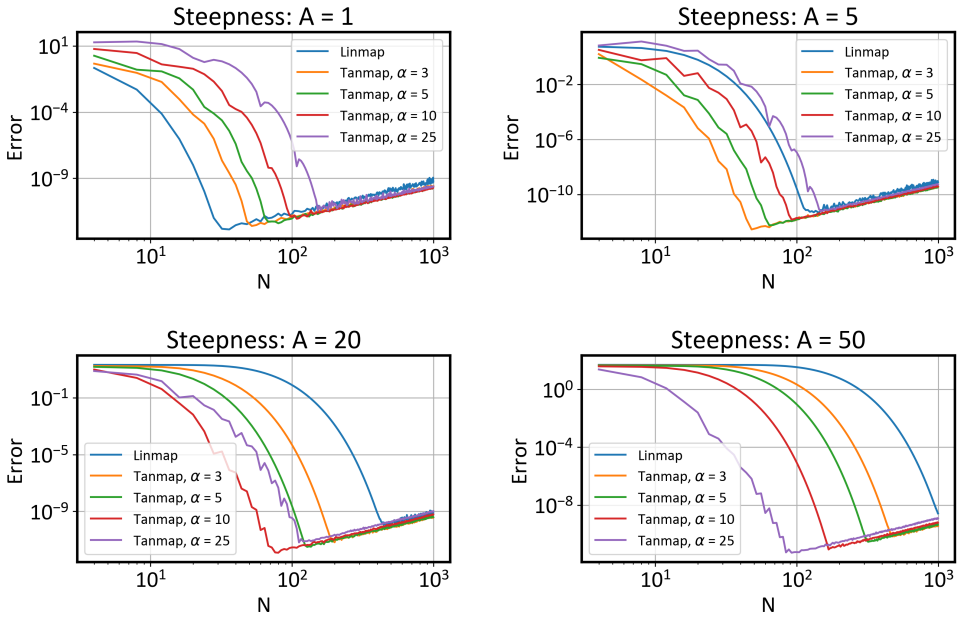


Figure 7: The effect of introducing a *tanmap* centred around 0 on the approximation of the spectral derivative of $\tanh(Ax)$.

This rearranges to:

$$f''(g(x)) = \frac{\frac{d^2(f(h(x))}{dx^2} - f'(h(x)) \cdot h''(x)}{h'(x)^2}. \tag{3.10}$$

Using Equation (3.6), this gives:

$$f''(h(x)) = \frac{\frac{d^2(f(h(x))}{dx^2} - \left(\frac{d(f(h(x))}{dx} \cdot (h'(x))^{-1}\right) \cdot h''(x)}{h'(x)^2}. \tag{3.11}$$

This implies that, for the second derivative:

$$\mathbf{D}_{map} = \left(\mathbf{D}^2 - (\mathbf{D} \cdot \mathrm{diag}(1/Dx) \cdot \mathrm{diag}(DDx)\right) \cdot \mathrm{diag}(1/Dx^2), \tag{3.12}$$

where $\mathrm{diag}(1/Dx)$ is as above, and $\mathrm{diag}(DDx)$ is a diagonal matrix with elements corresponding to the second derivative of the mapping at the grid points x_j.

(As shown in Figure 9, the full method is marginally more reliable, though the reasons for this will not be dwelt on.)

3.3 Integration

To calculate the integral in conjunction with Chebyshev spectral methods, one simply needs to use substitution. If a function f has values f_j when evaluated at points x_j, where x_j is a Chebyshev grid transformed under a mapping h, then the numerical integration performed by multiplying by the integration weights w_j is equivalent to the following:

$$w^T f(h(x)) \approx \int_{-1}^{1} f(h(x))dx. \tag{3.13}$$

Letting $y = h(x)$ and defining $y_0 = h(x_0 = -1)$ and $y_N = h(x_N = 1)$, this becomes:

$$w^T f(y) \approx \int_{y_0}^{y_N} f(y)dy = \int_{y_0}^{y_N} f(y)\frac{dy}{dx}dx. \tag{3.14}$$

Therefore, in order to calculate the integral between y_0 and y_N of a function which has been

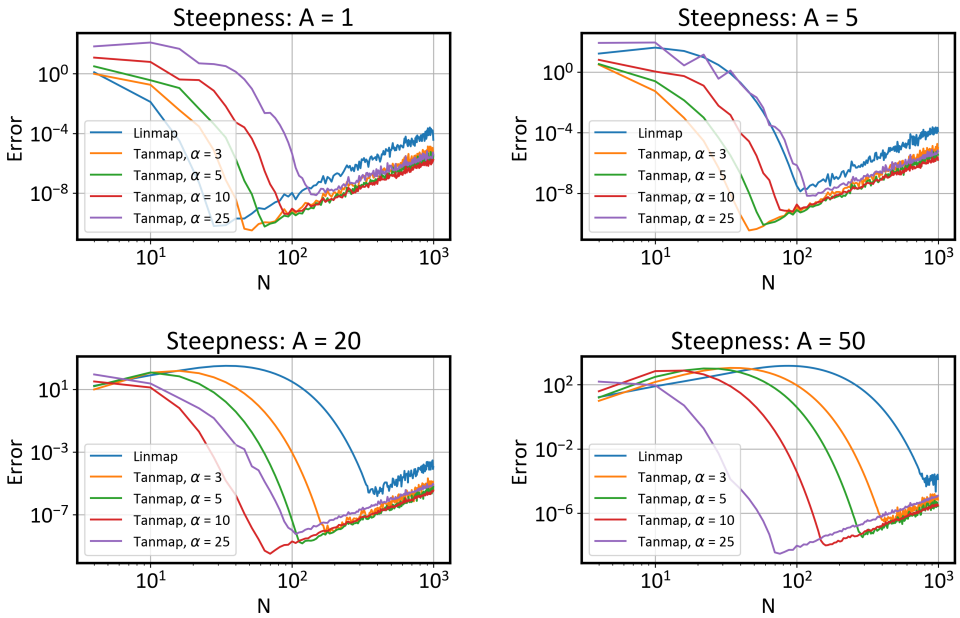


Figure 8: The effect of introducing a *tanmap* centered around 0 on the approximation of the second derivative of $\tanh(Ax)$.

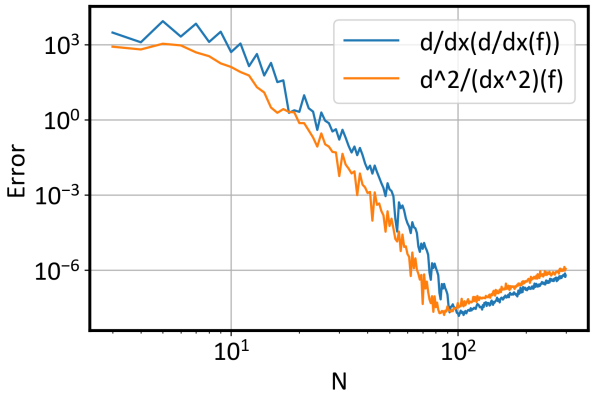


Figure 9: A comparison of the two methods of calculating the second derivative.

sampled on the transformed grid, one must first multiply the transformed points by the derivative of the mapping used. This is implemented simply by multiplying the integration weights w_j point-wise with the derivative of the map at each point x_j.

4 Interpolation

As has been shown, extremely accurate solutions can be found with relatively few points when Pseudo-spectral methods are used. This however presents a separate problem, in that if our solution only includes 20-50 points, it may look very jagged when plotted with standard linear interpolation, and we may want to know the solution at a point that is not near our chosen grid. The solution to this problem is interpolation, which is a way of extending a set of fixed outputs to a continuous function. The method used throughout this paper is known as *Barycentric* interpolation, based on the work found in [6].

Barycentric interpolation is a specific case of Lagrange Interpolation. This is the general case where the polynomial of degree $n-1$ of a set of n points x_i for $i = 1 \dots n$, with points $f(x_i) = f_i$ is given by the formula:

$$p(x) = \sum_{j=1}^{n} f_j l_j(x), \quad l_j(x) = \frac{\prod_{i=1, i\neq j}^{n}(x - x_i)}{\prod_{i=1, i\neq j}^{n}(x_j - x_i)}. \tag{4.1}$$

This is constructed since the polynomial $l_j(x)$ has the property that

$$l_j(x_i) = \begin{cases} 1, j = i, \\ 0, j \neq i. \end{cases} \tag{4.2}$$

This method in general has a few shortcomings, namely:

1. Each evaluation of $p(x)$ requires $O(n^2)$ additions and multiplications[3].
2. Adding a new data pair (x_{n+1}, f_{n+1}) requires a new computation from scratch.
3. The computation is numerically unstable.

These can be overcome in part as will be shown.

The first observation to make is that for each l_j, the numerator can be written simply as $l(x) = (x - x_1(x - x_2) \dots (x - x_n)$ divided by $(x - x_j)$ (provided that $x \neq x_j$). Also, let us define the barycentric weights, w_j like so:

$$w_j = \frac{1}{\prod_{i\neq j}(x_j - x_i)}. \tag{4.3}$$

We can then write:

$$l_j(x) = l(x)\frac{w_j}{x - x_j}. \tag{4.4}$$

This factor of $l(x)$ appears in every $l_j(x)$ and does not depend on j, so it can be pulled outside the sum of (4.1), yielding

$$p(x) = l(x)\sum_{j=1}^{n}\frac{w_j}{x - x_j}f_j, \tag{4.5}$$

and thus $p(x)$ can be calculated using $O(n^2)$ operations to calculate some fixed quantity independent of $f(x)$, and then $O(n)$ operations to evaluate p once these quantities are known. This

[3] This can also be made more efficient through the use of Horner's rule, but this is beyond the scope of the project.

puts it on par with the other main form of interpolation used, known as Newton Interpolation, which is found in many standard Numerical Analysis texts.

In order to update the interpolation when a new grid point x_{n+1} is added, one must carry out two calculations:

- Divide each w_j by $x_j - x_{n+1}$. This costs $O(n)$ operations.
- Compute w_{n+1} using (4.3). This costs $O(n)$ operations.

And thus, the method can be updated using only $O(n)$ operations too. Equation (4.5) can be modified further to arrive at a more elegant version, which is a version that will be used in this project. Let us interpolate, along with the data f_j, the constant function 1. Then we get

$$\sum_{j=1}^{n} l_j(x) = l(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j}. \tag{4.6}$$

Dividing equation (4.5) by equation (4.6), we arrive at the standard barycentric formula:

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x-x_j} f_j}{\sum_{j=0}^{n} \frac{w_j}{x-x_j}}. \tag{4.7}$$

This formula is now almost useful, with the only remaining thing to do being to calculate the weights w_j. When the points x_j are arranged according to a Chebyshev distribution, as is the case for the methods considered in this paper[4], the weights w_j are known to be simply [14]

$$w_j = (-1)^j \delta_j, \quad \delta_j = \begin{cases} \frac{1}{2}, & j = 0, n, \\ 1, & \text{otherwise.} \end{cases} \tag{4.8}$$

4.1 Implementation

A code based on the work from Dr Goddard has been used in this paper, *BaryM.m*. This adapts the principals outlined in 4 and uses it to generate a matrix $\mathbf{B}$ that will interpolate a function from one grid to another. (Say, for example, from a Chebyshev distributed grid onto a fine equi-spaced grid.) This can be achieved by noting that, for any individual x, the formulae given in Equation (4.7) can be thought of as a linear combination of the outputs on the grid x_j, $j = 1 \dots n$, (these outputs being f_j). So for any given grid of points, one can construct a matrix such that for every desired output point y, a row of the matrix is constructed with entries according to the formula given in Equation (4.7). The interpolations are then calculated at these points as $\mathbf{f_{interp}} = \mathbf{Bf}$.

One issue that can occur with this method is the possibility of the interpolating point x being equal to a grid point x_j. In this case, the formula calls for dividing by zero, which presents problems. Thankfully, there is an easy fix here, which is just to replace any rows where this occurs with a row of zeros and a single 1 corresponding to the problematic grid point. This is of course because if $x = x_j$, then of course $f(x) = f(x_j) = f_j$.

The calculations are done in this way as it saves a considerable amount of computation. If the grids we want are fixed, (as they will be for at least some amount of time), then the matrix does not need to be calculated again for different sets of outputs f_j.

5 Stationary Functions

The effect of using the *tanmap* on the case of differentiating the function $f(x) = \tanh(Ax)$ where A is large is illustrated in Figure 7. As this shows, the linear map does eventually show spectral

[4] In the case where adaptive mappings are applied, the interpolation will be applied before the inverse mapping is applied, and thus will always be used on a Chebyshev grid.

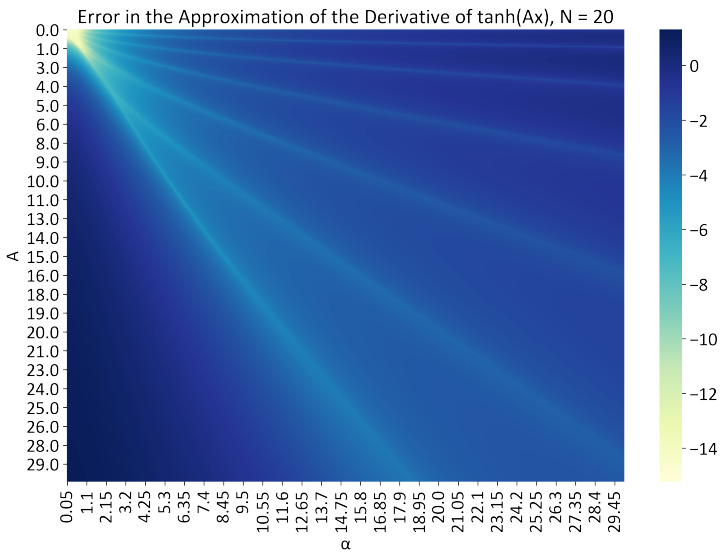


Figure 10: The effect on the error of the first derivative of $tanh(Ax)$. The error given in the colour bar is a $\log_{10}$ scale.

convergence, even in the extremely steep case of $A = 50$, but this takes a much larger value of N, ten times as many points as using a roughly fitted tanmap. It also shows that as the steepness of the function increases, the optimal argument for the tanmap changes: when $A = 5$, the optimal argument is $\alpha \approx 3$, but when $A = 50$, optimal argument is $\alpha \approx 25$. These differences are even more prominent in the case of the second derivative, as can be seen in Figure 8. An important point to note is that, even with the optimal mapping, when the function gets very steep, the error achieved is still larger, and due to the fact that more points are required to reach the level of rounding errors within the computation system, in fact the minimum error available is also higher. This is illustrated by the 'optimal' line of the values for α for each A in Figure 10.

Also indicated by Figure 10, is that in general the optimal value of α for a given steepness, A, is very closely approximated by $\alpha = \frac{A}{2}$, with the optimal α getting slightly above this when A rises in value above $A = 16$. Another feature shown most clearly in Figure 10, but also present in Figure 11 is the presence of secondary optimal lines separate from the main branch representing the optimal α value. While the cause of these will not be investigated further - other than to say they seem to be some sort of harmonic frequency of the base note, in that they somewhat coincidentally happen to place grid points at important points of interest - this however is noteworthy, as it illustrates a potential local minima that an optimisation routine used in Section 6 might fall into.

Figure 11 shows the different accuracies of a *tanmap* on the derivative of tanh(20x) for a range of parameters α and N. As expected, the ideal value of α is roughly constant for each N, and correctly tuning α can result in several extra digits of accuracy. Only even values of N are shown on this grid, as in the case where N is odd, a significantly lower accuracy is displayed in comparison to a similar number of even points. This is due to the fact that when N is even, the actual number of points is odd (as there are always $N + 1$ points). This means that there is always a point exactly in the middle of the region [-1,1], which is the most important part of the domain as it is right in the middle of the area that the mapping is attempting to concentrate on. This is a specific phenomena that only occurs when the wave is exactly in the middle of the domain, but similar behaviour is possible if mappings with large α values are used to concentrate the points around a different area, although since the centre will not go exactly to the point β, the effect will be much less prominent, if at all observable.

Figure 13 shows the effect of 'missing' a function by a set amount. As can be seen, even if the estimate for β is significantly off, causing a considerable loss of accuracy when compared to its optimal value, it can still produce a significant improvement in accuracy when compared to a fixed grid. This implies that in the case of a moving wave type problem, we expect to obtain

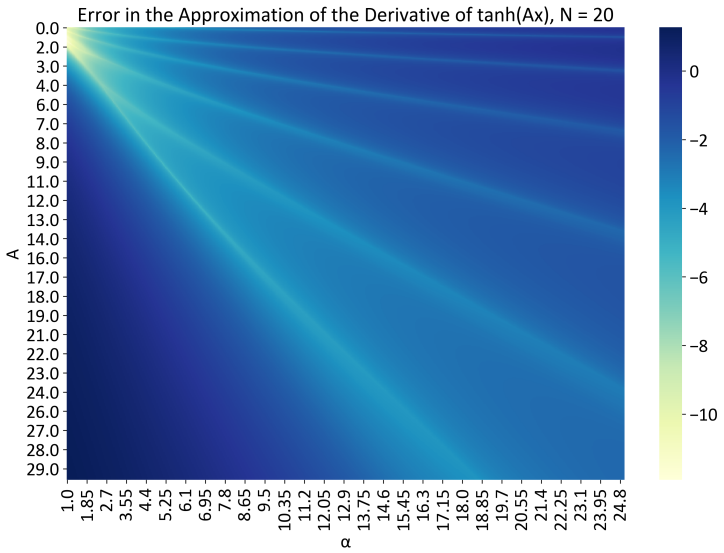


Figure 11: Error changing as alpha changes and number of points varies. The error given in the colour bar is a $\log_{10}$ scale.

helpful results even when the tracking of the wave is not extremely accurate.

6 Methods for Updating the Grid

6.1 Sobolev Norm

As seen in papers [3, 2, 17, 1], in the absence of an exact solution, as is the case in all the applications one may want to use this technique, another measure to determine whether an appropriate grid is being used must be found. Several methods are available, the most common of which being to find a grid which minimises some functional $I(f)$ for f being an interpolation in some grid. (For an alternative technique, see [17, 5].) The functional to be used in this paper is a version of a weighted Sobolev norm:

$$I(f) = \int_D a|f|^2 + b\left|\frac{df}{dx}\right|^2 + c\left|\frac{d^2f}{dx^2}\right|^2 dx. \tag{6.1}$$

This is similar to that used in [3], chosen with a constant weighting, instead of their preferred method of a Chebyshev weight function $w(x) = (1-x^2)^{-1/2}$. It can be seen that in some sense that in attempting to minimise this value for a function, it should avoid any unnecessary 'unsmoothness' or other undesirable features, most specifically, the avoidance of any occurance of the Runge effect is sought.

6.1.1 Normalisation Constants

Within Equation (6.1), it can be seen that there are three normalisation constants, a, b and c. In this project, it was decided that these constants should be chosen so that the three components of the integral within (6.1) all have similar sizes. This approach, is, as far as the author of this project is aware, novel. This requires knowing some information about the function to be studied before use, if more precise tunings are desired. In the case of a function of the form $f(x) = \tanh(Ax)$, its derivative has maximum size of order A, and its second derivative has maximum size of order A^2. Therefore, the second and third terms of (6.1) vary according to A^2 and A^4 respectively. Due to this, the most 'balanced' setting for the values a, b, and c are given by $b = aA^{-2}$ and $c = aA^{-4}$, where there is an arbitrary choice for the value of a as the scaling is irrelevant.

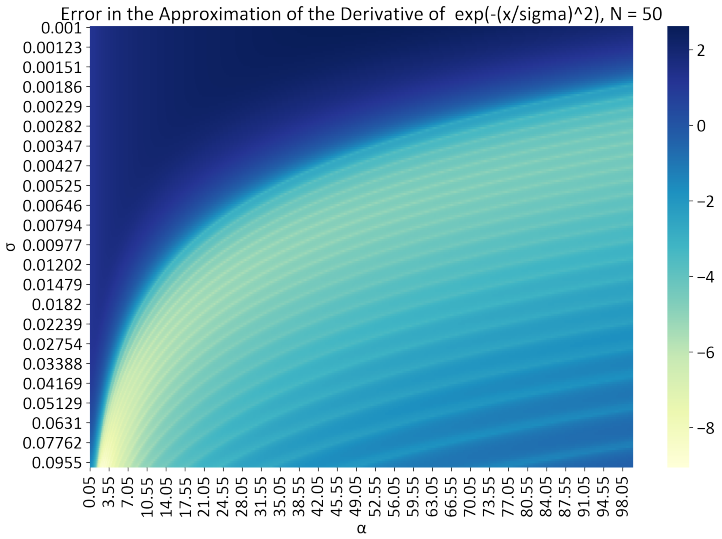


Figure 12: Error changing in derivative of $e^{-\left(\frac{x}{\sigma}\right)^2}$ as σ and the concentration α are varied. The error given in the colour bar is a $\log_{10}$ scale.

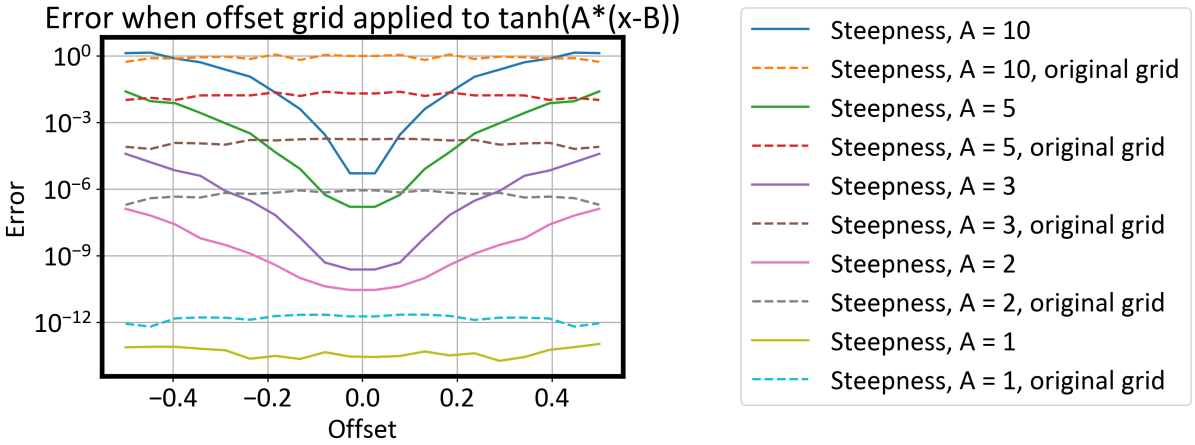
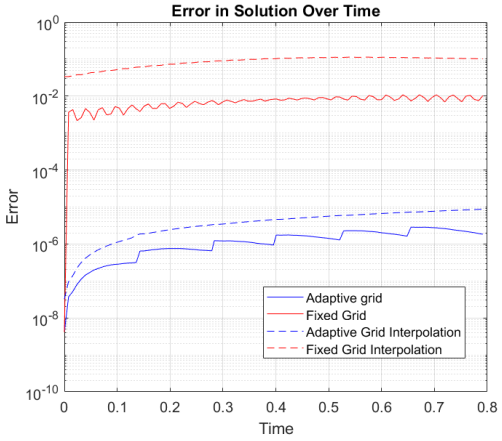
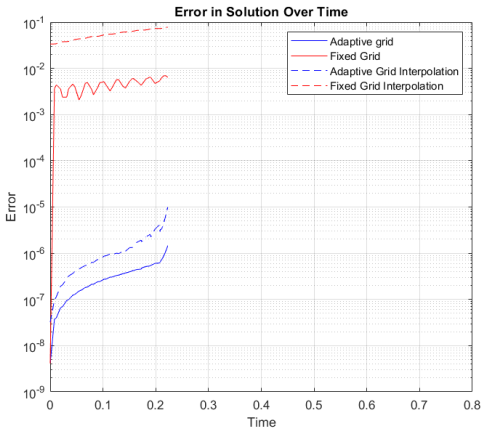


Figure 13: The effect on accuracy of the *tanmap* when the value of β is offset from the true location of the inflection point.



(a) Correctly normalised constant in Sobolev norm.



(b) Unadjusted, $a = b = c = 1$.

Figure 14: A comparison of a solution of the advection equation (see Section 8.3.) With the incorrectly normalised constants, the norm becomes too sensitive, leading to excessive regridding, and thus the solution fails. All other parameters are kept constant.

In the case of a function of the form $f(x) = e^{-\frac{x^2}{\sigma}}$, similarly the sizes of the first and second derivatives of $f(x)$ vary according to $\frac{1}{\sigma}$ and $\frac{1}{\sigma^2}$ respectively, and thus the second and third terms of (6.1) vary according to $\frac{1}{\sigma^2}$ and $\frac{1}{\sigma^4}$ respectively. Thus the optimal choices for a, b and c are: $b = a\sigma^2$, and $c = a\sigma^4$. Due to the nature of the problems involving this sort of function, the grid updating methods will use an input of the current time in PDE problems to roughly estimate the value of σ or its equivalent when possible. This method for calculating the Sobolev norm does seem to produce more stable results. A brief example of this is demonstrated in Figure 14.

6.2 Updating the grid

In order to update the grid, several MATLAB functions were developed for different cases. They all rely on using the *tanmap* transformation, and altering the first and second arguments α, β that it takes in.

6.2.1 Alpha Fixed

In the case where α, which in some sense represents the degree of concentration of the grid points, is fixed, the only remaining task is to determine β. The following procedure was implemented:

- Using the given grid, calculate the derivative and interpolate onto a fine grid.
- Find the maximum absolute value of this interpolated derivative.
- Set this as the initial guess for β.
- Construct a function *findA* as follows:
  - For any input α, construct a grid according to the *tanmap* using this input as well as the previously given β and boundary values.
  - Calculate an interpolating function given the previous grid and outputs.
  - Evaluate the interpolating function on the new grid points.
  - Calculate the Sobolev Norm of this interpolation.
- Run the MATLAB built in optimiser *fminsearch* on a function *findB*, with the initial guess as determined.

The first three steps were implemented due to the algorithm *fminsearch* apparently being extremely sensitive to the initial guess used. This is because there are several local minima in the function *findB*, which *fminsearch* will get stuck in.

This case is useful in situations where a traveling wave occurs, for example an advection equation or the pedestrian dynamics equation (8.11) discussed further on. In these situations, the steepness of the solution stays roughly constant over time, and hence α can be kept constant too. In contrast, the centre of the steepness moves, and thus β must be updated frequently.

6.2.2 Beta Fixed

In the case where β, the point of focus of the *tanmap* is kept constant, all that must be updated is α, the concentration. This is done as follows:

- Construct a function *findA* as follows:
  - For any input α, construct a grid according to the *tanmap* using this input as well as the previously given β and boundary values.
  - Calculate an interpolating function given the previous grid and outputs.
  - Evaluate the interpolating function on the new grid points.
  - Calculate the Sobolev Norm of this interpolation.
- Implement the built-in MATLAB function *fminsearch* on the function *findA* with the initial guess of the previous α.

This is for use in a function with a single peak or point of interest, which doesn't move, but does vary in terms of steepness. Examples of this include the moving Gaussian, and the diffusion equation.

6.2.3 Both Alpha and Beta Vary

In this case, a combination of the above two methods can be used. It is still advisable to update the initial guess for β by calculating the maximum point of the derivative, as the sensitivity of the optimisation programs to initial guesses is still very important. This may be necessary in certain situations, but it should be noted that in the cases tried here, it was not advisable to use when one variable could be fixed, as it was extremely prone to finding local minima which were non-optimal, and then being unable to correct itself.

7 Moving Exact Solutions

Before implementing the above procedure on full PDEs, the method was tested on some moving examples.

In this section the first experiments with a changing grid are explored. Here they are used to interpolate a steep, moving tanh wave, and a spreading out Gaussian.

The procedure used in each case is as follows:

- Calculate the Sobolev norm (see section 6.1) of the function in some initial (good) grid.
- Advance the function forward in time.
- Sample the new function on the old grid, and interpolate.
- Calculate the Sobolev norm of the interpolation of the function.
- If the ratio of the previous norm to the initial norm is sufficiently different, update the grid, and hence the initial norm.
- Repeat.

The definition of "sufficiently different" (based on the work in [3]) shall be defined as follows: Update the grid if:

$$c_1 \leq \frac{I(f)}{I(f_0)} \leq c_2, \tag{7.1}$$

where $I(f_0)$ is the norm of the function evaluated at the last time the grid was updated, is no longer true. The precise values for c_1 and c_2 depend greatly on the problem being considered, as is discussed below. They will always take the form $c_1 = 1 - tol, c_2 = 1 + tol$ in this project.

7.1 Moving Tanh

Here the method for updating the grid used was the Alpha fixed version, as the tanh wave will have a consistent steepness, but a moving location. The tanh wave in question is as follows:

$$f(x) = \tanh(A(x - C)), \quad C \in [-0.25, 0.25]. \tag{7.2}$$

(This was implemented in *tanhmove.m.*)

In the case of a fixed steepness but moving wave, it would be expected that an exact solution would have a very consistent norm. This is one reason why very fine tolerances must be chosen in determining c_1 and c_2, where the tolerance, T, is used such that $c_1 = 1 - T$, and $c_2 = 1 + T$.

The exact optimal value depends on the number of points used (N), and the steepness of the tanhwave (A).

An in depth look at the case when the steepness, A, is equal to 40, will be shown, although this choice of A is mostly arbitrary, and largely picked for illustrative purposes.

As this is a rather steep example, about 100 points are required to get down to machine precision, even with the appropriate mapping.

7.1.1 Initial Alpha Value

In general for a tanh wave, the optimal value for the concentration α was determined to be very near to the steepness, A, divided by 2. This agrees fairly well with the optimal arguments expected based on the preliminary explorations with stationary tanh waves, although it is slightly lower. This seems to be because it allows for slightly more flexibility in the selection of β. This also makes the 'optimal' range for β a little wider when regridding is required, and thus increases the chance that the routine will find the correct minima. This exact tuning is specific to the range considered here ([-1,1]), and it should be noted that for an equally steep wave on a larger range, a higher value of alpha must be used, as the effect of rescaling causes the function to appear steeper.

7.1.2 Tolerance

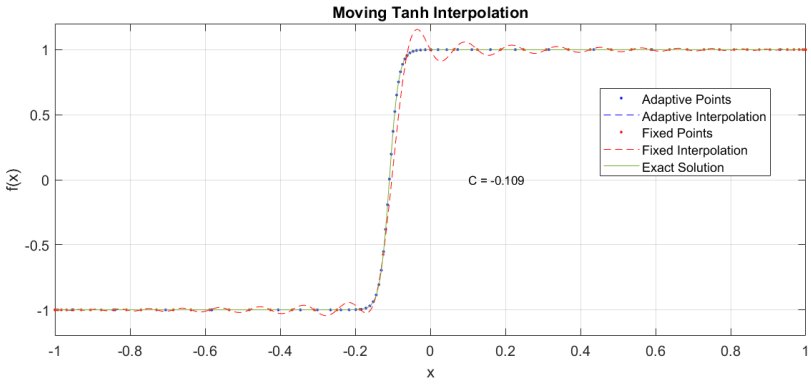
For this type of problem, when the exact variance in the Norm of the function is very low, it is necessary to pick very low values for the tolerance, and in fact the lowest possible tolerance before the routine starts finding false minima. Here a value of 10^{-8} was used. This value must be increased for steeper functions and decreased if more points are used. An example of the results obtained when the tolerance is too low (10^{-9}) is illustrated in Figure 15c.

7.1.3 Range of C

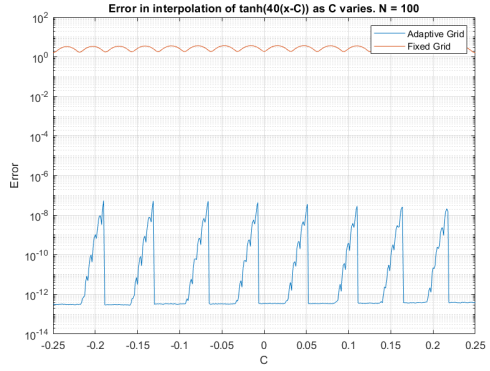
A range of C going from [-0.25, 0.25] was used in these experiments, as it was found that waves nearer the boundary mean that the exact norm of the solution changes slightly, and so this was avoided when very precise tolerances were used.

7.1.4 Results

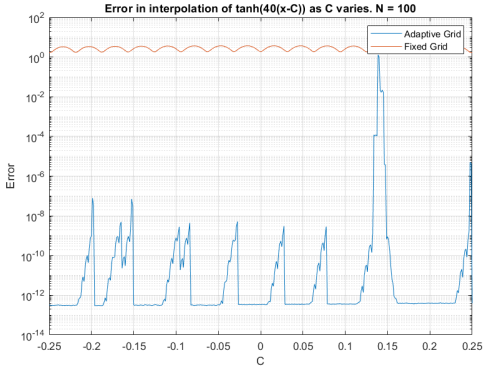
In all following discussion, error is calculated in the 1-norm, and any reference of maximum error refers to the maximum recorded 1-norm reading over all times, or steps, in a simulation.



(a) A screenshot of the moving tanh and its adaptive and fixed interpolations.

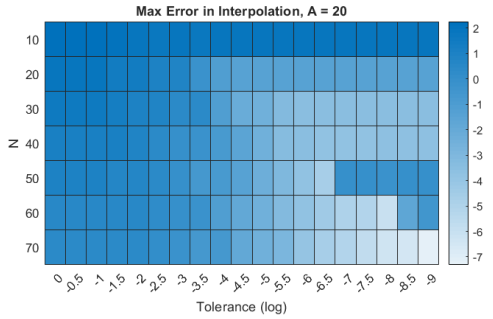


(b) The error in the interpolation of the function $\tanh(40(x - C))$ as C varies.

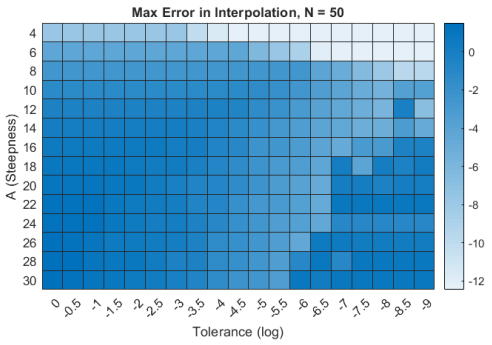


(c) An example of the tolerance being too low (in this case set to 10^{-9} for this problem, resulting in a gridding error around $C = 0.15$.

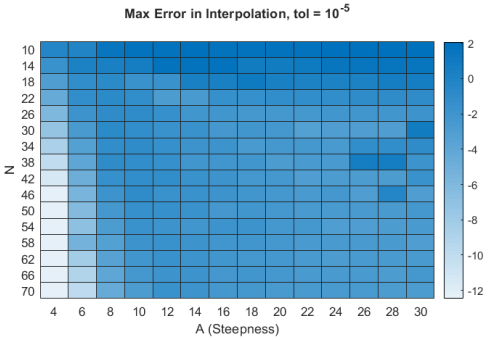
Figure 15: Examples of the error of interpolation with correctly and incorrectly tuned tolerance.



(a) The effect on accuracy when function steepness is kept constant.



(b) The effect on accuracy when number of grid points is kept constant.



(c) The effect on accuracy when tolerance of the regridding function is kept constant.

Figure 16: The effect on accuracy of the interpolation of the moving tanh wave when one of the three parameters of N, A and tol are fixed.

With correctly tuned α, T, and C, the results obtained in this procedure are found in Figure 15b. The use of a correct mapping enables an increase in accuracy of several digits. The effect of the regridding can be seen by the sudden gains in accuracy that occur after a 'spike'.

For a more in depth demonstration of the effects of changing around the values for the tolerance, N, and A values, see the heatmaps in Figure 16.

As shown by Figure 16a, when the number of points is increased, the optimal tolerance used must decrease. This is due to the fact that the increased number of points allows for a higher accuracy, and therefore a more sensitive value for the tolerance must be used to detect when the function is moving.

Figure 16b shows that, when the number of grid points is kept constant, as steepness increases, the optimal tolerance value must increase, as otherwise events such as seen in Figure 15c will occur.

Figure 16c demonstrates that without adjusting the tolerance correctly with the number of points, and tolerance levels, there is little to no gain made from increasing the number of points, as one would usually expect from a spectral method.

7.2 Spreading out Gaussian

Here the method of choice for updating the grid is the Beta fixed version, as the peak of the Gaussian spreads out, requiring a change in α, but the location of the peak remains constant. (This was implemented in *gaussmove.m.*) The function to be studied is:

$$g(x) = e^{-\frac{x^2}{t^2}}, \quad t \in [t_0, 1]. \tag{7.3}$$

The exact function itself has a changing norm over time, so we would expect even extremely accurate interpolations to have a changing norm. This means that the tolerance for the adaptive

system must be adjusted so that it does not change too quickly. Here this is reflected in the choice of tolerance, which is significantly larger than the tolerance value in the case of the *tanh* wave.

7.2.1 Initial Values

In this case, the choice for the initial β value is clear, and should be placed at the peak of the Gaussian, which in this case was simply at $\beta = 0$. The initial choice for α is less obvious, and varies significantly depending on the initial steepness of the function. In this case, for an initial variance of $\sigma = 0.01$, the value $\alpha = 30$ was determined to be sufficient.

It should also be noted that the argfinder routine does not seem to be very good at finding optimal α values. In order to combat this, the estimates for new arguments at points where the grid adapts were altered slightly. In this case, since it is expected that the solution spreads out, (similarly to a Diffusion equation), before inputting the previous estimates for the correct arguments into the argfinder routine, the estimate for α was multiplied by $mult$, a new parameter that was introduced within this project. The ideal value for $mult$ depends on the task at hand, but the comparison in the accuracy of both methods is shown in Figure 17a and Figure 17b.

7.2.2 Results

As shown in Figure 17a, at times when the Gaussian function is steep, the adapting grid provides several digits of increased accuracy. It should be noted that the function eventually flattens out to the point that the fixed grid is sufficient, and past this point a range of values for α will give a solution that is accurate to machine precision, thus making the adaptive routine obsolete. Figure 17b demonstrates the necessity of the premultiplying value used in this type of problem.

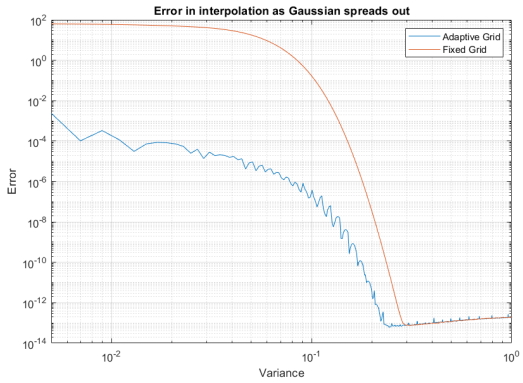
In all the results shown in Figure 18, unless the parameter is one of the ones being varied in each individual subfigure, the parameters take the following values: $N = 50$, $t_0 = 0.001$, $\alpha_0 = 30$, $tol = 10^{-1}$, $mult = 0.7$. The relationship between optimal multiplier values for a given tolerance is shown in Figure 18a. When the tolerance is set to $10^0 = 1$, then the function does not regrid at all, explaining why the premultiplying value does not affect the results. Otherwise, in general one can see that when tolerance is decreased, the premultiplier value must be increased. This is due to the fact that as the tolerance decreases, the number of regrids that occur increases, and thus the reduction in concentration does not need to be so severe at each individual event. Figure 18b shows the effects of varying the tolerance and the number of points, N. As one would expect, whatever value of tolerance is used, increasing N results in lower errors. By comparing rows, it can be seen that, at least for the given settings of t_0, α_0 and $mult$, the optimal value for the tolerance is 10^{-1}. Finally, Figure 18c shows the effects of varying the initial time, and the initial α_0 guess. As might be expected, when the initial time is lower, the optimal value for α_0 increases. This is because the initial shape of the gaussian is more steep, and thus the initial grid needs to be more focused around its peak to accurately interpolate it. A less expected behaviour is the much higher minimum of the error when the initial time is especially low. This is due to a combination of two factors. The first is that, even with appropriate mappings, the accuracies of interpolations of extremely steep functions is lower than interpolations of relatively 'flatter' ones. This was noted in Section 5. The second is that, as α_0 is increased to very large numbers, it may be necessary to decrease the value of $mult$, so that it may still reach the appropriately much lower values required when the function spreads out.

8 Applications in PDEs

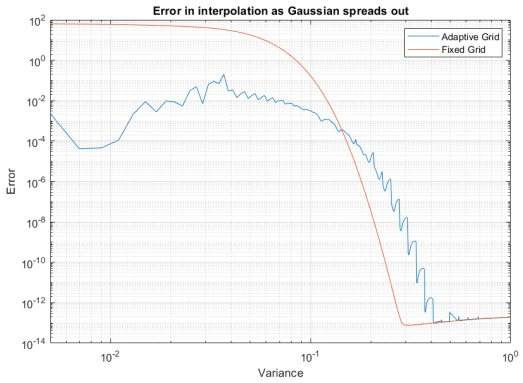
8.1 DAEs

Standard ODE solvers solve systems of the form:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t). \tag{8.1}$$

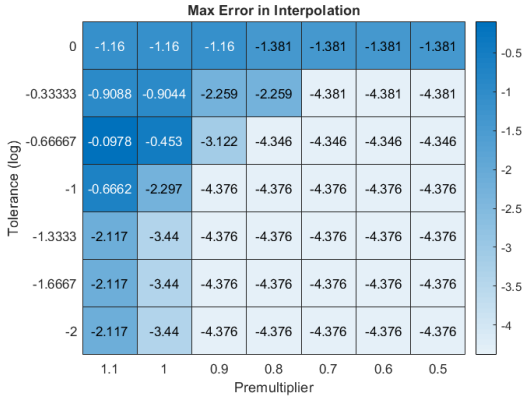


(a) The error in the interpolation of a Gaussian distribution as the variance increases.

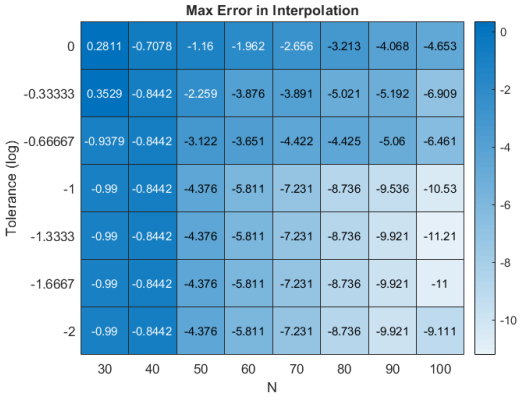


(b) An example of the effect of failure to precondition the estimates of the grid points.

Figure 17: Example error rates of interpolation of a spreading out Gaussian curve.



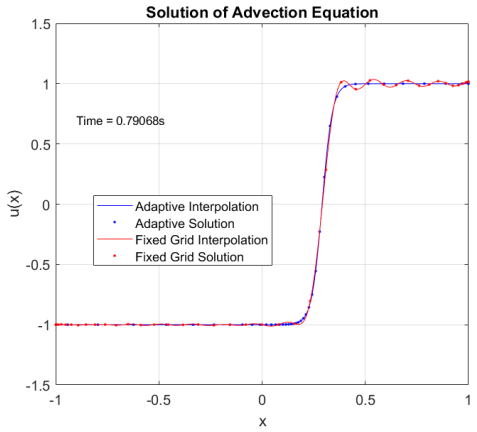
(a) Varying tolerance and multiplier values.



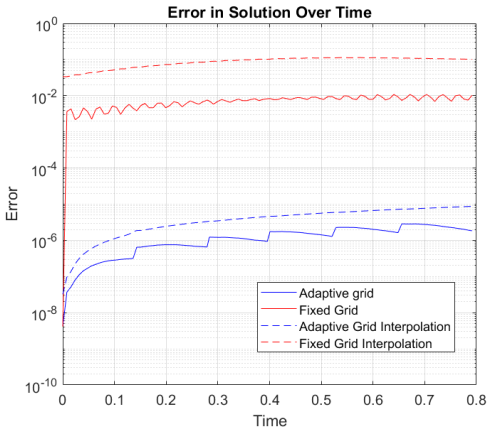
(b) Varying tolerance and number of points.

(c) Varying initial time, and initial alpha guess.

Figure 18: The effect on accuracy of varying several sets of parameters.



(a) An example of the solution and its interpolation for $N = 40$ points with the fixed and adaptive grids.



(b) The error in solution of the advection equation, with $N = 100$ points for the moving grid, and the fixed grid. The interpolation is done onto a uniform grid of 1000 points.

Figure 19: Examples of the solution and its changing error over time for fixed and adaptive grids.

Or, more generally:

$$\mathbf{M}\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t). \tag{8.2}$$

These problems are very similar in the case where $\mathbf{M}$ is non-singular, as the linear system can simply be rearranged. A more difficult situation arises when $\mathbf{M}$ is singular. In this case the system is known as a Differential Algebraic Equation, or DAE. These are harder to solve, and require different solvers computationally. In this paper, the MATLAB solver *ODE15s* has been used.

8.2 Implementation with MATLAB solvers

In order to implement the checks used to adapt the grid as done for the moving functions demonstrated above, the MATLAB ODE solvers *events* system was used.

This was implemented as follows:

- For every step, calculate the Sobolev Norm of the solution.
- Calculate the ratio of this norm to the norm of the function at the last time the grid was updated.
- If this ratio falls out of the specified tolerance of the function, update the grid.
- Update the values for the most recent grid Norm.

8.3 Advection Equation with Steep Initial Condition

Here the solution of the advection equation with steep initial conditions was studied. The initial conditions chosen were a steep tanh wave of the form used in section 7.1. It is expected that this should produce very similar results to the program in that section, and the results can be compared to the exact solution of the 1D advection equation, shown here. For the 1D advection equation:

$$\frac{d\mathbf{u}}{dt} = c\frac{d\mathbf{u}}{dx}, \quad \mathbf{u}(0, x) = \mathbf{f}(x), \tag{8.3}$$

the solution is:

$$\mathbf{u}(t, x) = \mathbf{f}(x - ct). \tag{8.4}$$

It should be noted that the solution of the advection equation has the exact same solution as the moving tanh wave from 7.1, so the only expected differences between the results found here and in that section should be that, instead of returning accuracy down to the almost exact levels found by resampling the initial condition, instead continuing on from the solution found at the time of the regridding. It should be noted that the advection equation is in general numerically unstable, so it could be argued that this is not the fairest test of the methods developed here. Nevertheless it is included due to it having exact solutions, thus allowing easy error comparisons, as well as its similarity to Section 7.1.

8.3.1 Implementation

As demonstrated in Section 7.1, when tolerances are set too low for this sort of problem, it results in extremely large errors. These very large errors have been found to result in solutions that vary drastically from the exact solution, and also result in the regridding being triggered much more often than it should be, requiring much longer computation times. In order to address this, a line in the implementation code was written such that if the time between regridding is sufficiently different from the time taken to get to the first regridding, the program will stop, and a NaN is given as output for the error.

8.3.2 Parameters

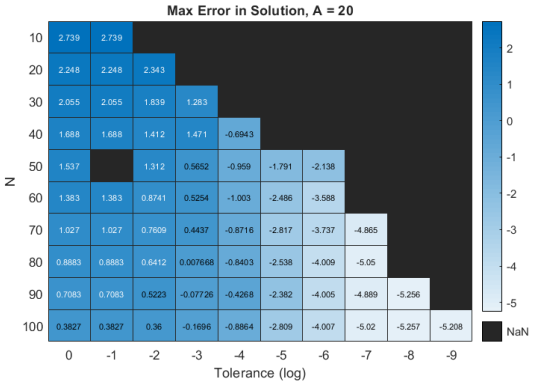
Here the chosen initial condition was chosen to be $\tanh(A(x+0.5))$, and the initial mapping was chosen such that $\alpha = \frac{A}{2}$, and $\beta = -0.5$. The wave speed, c, was set to 1. The parameters that will be studied in the results section are the effects of varying: the steepness (A), the number of grid points (N), and the tolerance of the regridding function (tol).

8.3.3 Results

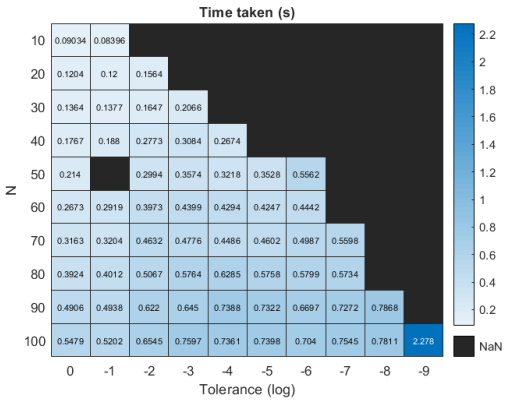
The results in the specific case of the parameters laid out above is shown in Figure 19. The results are compared to a traditional fixed grid with the same number of points, and it can be seen that the results are vastly improved, both in error of the solution points, and error of the interpolated points.

A more general view of the accuracy of the method when different parameters are varied is shown in Figure 20. This closely mirrors the results found in Figure 16, the most significant difference being that when the tolerance is too low for the function (in the sense that the regrid steps stop being regular), the PDE fails to solve. This is shown with the NaN values in the heat maps.

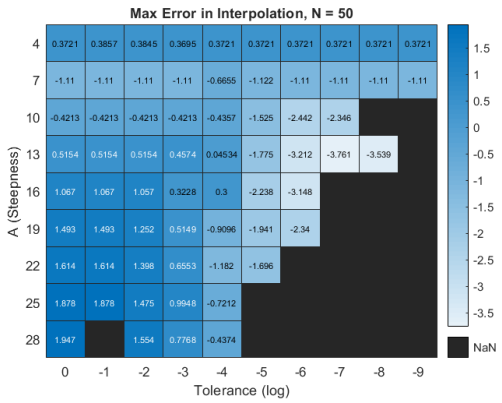
As shown in Table 1, the performance of the Adaptive method vastly outperforms that of the fixed grid when compared in terms of number of grid points used. Indeed, using just 40 points with the adaptive grid is more accurate than 100 points with the fixed grid. This effect is even more pronounced when looking at the interpolated error, however the computational costs, here measured by computational time, are much larger for the adaptive grid when the number of grid points is compared. The important distinction is comparisons in methods that lead to similar accuracies. If only a relatively small level of accuracy is required, say about 10^{-3}, then this is much faster with the fixed method, as although around 100 - 150 points are required, this is still an order of magnitude faster to compute. On the other hand, if more precise results are required, say to at least 10^{-5} in interpolation, then the adaptive method can get very close with around half the time, and with 16% of the points of a fixed grid method. These differences would be more or less pronounced as the initial condition is made more and less steep. The eventual plateauing of the error in both the adaptive and fixed methods occurs due to rounding errors, and the inherent instability of the advection equation. Illustrated in the last few rows of Table 1 is the rate that the calculation time increases as larger numbers of points are used. This can be seen in that although $N = 500$ points results in comparable times in the fixed grid method to the adaptive versions, when this is increased to $N = 1000$, the calculation time takes around 20 times longer. Although this kind of scale is unnecessary in almost all 1D applications, the



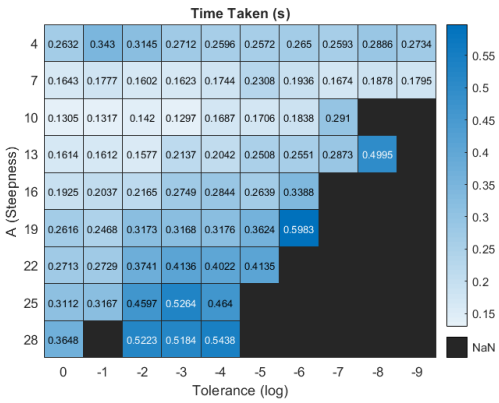
(a) The effect on accuracy when function steepness is kept constant.



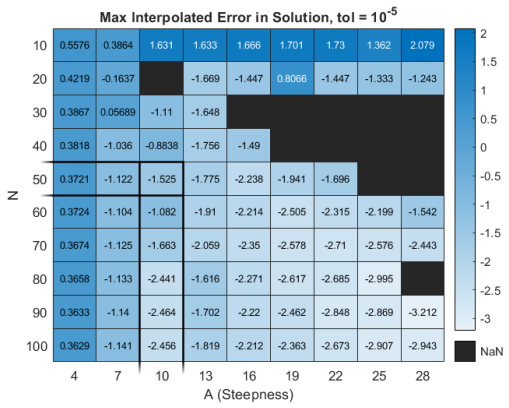
(b) The calculation time when function steepness is kept constant.



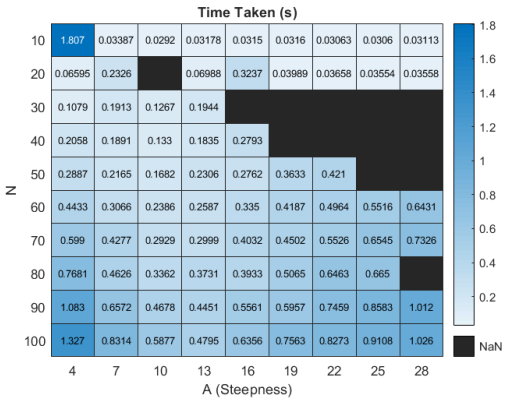
(c) The effect on accuracy when number of grid points is kept constant.



(d) The calculation time when number of grid points is kept constant.



(e) The effect on accuracy when tolerance of the regridding function is kept constant.

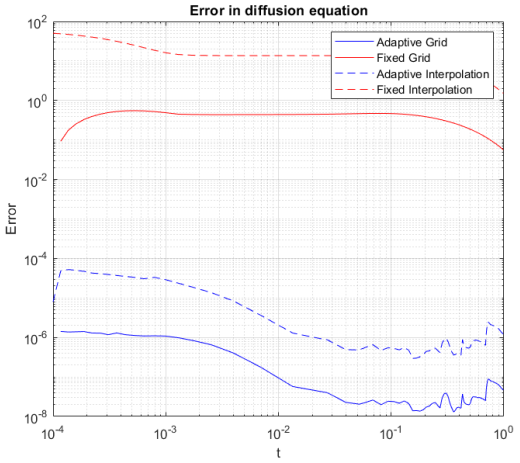


(f) The calculation time when tolerance of the regridding function is kept constant.

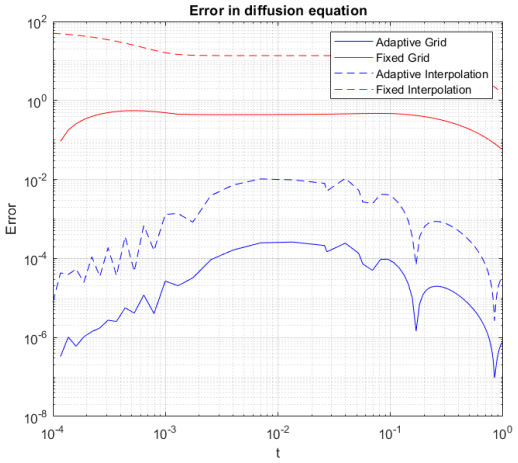
Figure 20: The effect on accuracy of the interpolated solution of the advection equation, and the time taken to calculate this solution, when one of the three parameters of N, A and tol are fixed. Occasional NaN results in inconsistent places seem to be due to numerical coincidences resulting in gridding issues. Error is measured as the maximum of the 1-norm of the error over time.

N	Optimal Tolerance	Adaptive Error (log10)	Fixed Error (log10)	Int. Adaptive Error (log10)	Int. Fixed Error (log10)	Adaptive Time (s)	Fixed Time (s)	Regrids
30	10^{-3}	-0.3496	-0.2618	1.2827	1.2758	0.24	0.03	4
40	$10^{-4.5}$	-2.9714	-0.3508	-1.5385	1.0946	0.32	0.04	5
50	$10^{-5.5}$	-3.2533	-0.6338	-1.7976	0.7278	0.41	0.05	7
60	10^{-6}	-4.8022	-0.9644	-3.5885	0.3009	0.46	0.07	6
70	$10^{-7.5}$	-5.7717	-1.1609	-4.8915	0.0684	0.59	0.07	7
80	10^{-8}	-5.8111	-1.4485	-5.1972	-0.2906	0.61	0.07	6
90	10^{-8}	-5.7378	-1.7868	-5.2556	-0.6837	0.70	0.11	5
100	10^{-9}	-5.7517	-2.0261	-5.2077	-0.9537	2.07	0.11	24
150	N/A	N/A	-3.5940	N/A	-2.6930	N/A	0.17	N/A
200	N/A	N/A	-5.1423	N/A	-4.2845	N/A	0.48	N/A
250	N/A	N/A	-5.6082	N/A	-4.8201	N/A	0.75	N/A
300	N/A	N/A	-5.4895	N/A	-4.7772	N/A	0.63	N/A
500	N/A	N/A	-5.9315	N/A	-5.4499	N/A	6.58	N/A
1000	N/A	N/A	-6.5067	N/A	-6.6591	N/A	123.10	N/A

Table 1: A comparison of error, interpolated error, and computational cost of Adaptive and Fixed methods when initial condition has $A = 20$. Adaptive entries for $N > 100$ are not included as no increases in accuracy fail to occur. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time.



(a) An example of the adaptive method against the fixed grid method, with $N = 50$ and $t_0 = 0.0001$, and $mult = 0.8$.



(b) An example of the adaptive method without the multiplier, and compared against the same fixed grid solution.

Figure 21: A comparison of the solution of the diffusion equation when the premultiplier is used and is not used.

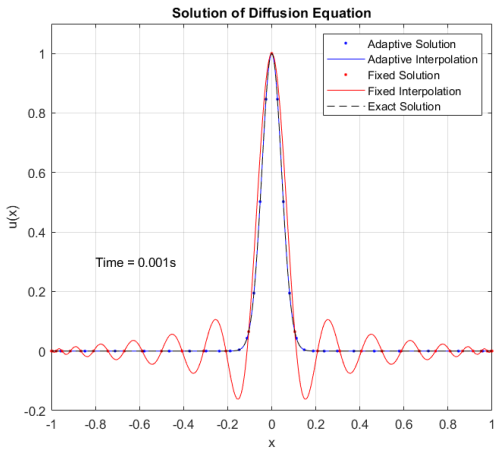
importance of keeping the number of points used down is clear, and of much greater relevance when dealing with higher dimensional PDEs.

8.4 Diffusion Equation

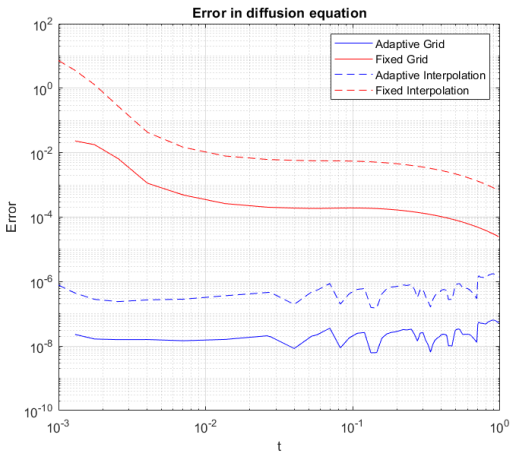
Here the solution of the Diffusion equation, with boundary conditions such that it matched the exact solution on an infinite domain, was solved, with steep initial conditions (corresponding to small initial times, t_0). That is:

$$\frac{d\mathbf{u}}{dt} = D\frac{d^2\mathbf{u}}{dx^2}, \quad \mathbf{u}(t_0, x) = e^{-\frac{x^2}{4Dt_0}}, \quad \mathbf{u}(t > t_0, \pm 1) = \sqrt{\frac{t_0}{t}} e^{\frac{-1}{4Dt}}. \tag{8.5}$$

The errors obtained in the specific case where the initial time was set to $t_0 = 0.001$, $N = 50$ grid points where used, and the tolerance was set to 10^{-1} is shown in Figure 21a. This is also using a preconditioning multiplier, *mult*, of the alpha argument of 0.8 for every regridding (as used in Section 7.2.) The effect without this multiplier can be seen in Figure 21b.



(a) A comparison of the adaptive and fixed methods and their solutions of the diffusion equation at an early time, using $N = 30$ points. The exact solution is indistinguishable from the adaptive interpolation at this scale.



(b) A comparison of the error rates of the adaptive and fixed grid methods and their interpolations, using $N = 50$.

Figure 22: An example solution of the diffusion equation using the adaptive method and the fixed grid alternative.

8.4.1 Parameters

The parameters that will be varied in this equation are as follows: t_0 (the initial time), N (the number of grid points), *tol* (the regridding tolerance), *mult* (the premultiplying value for the concentration α), and α_0 (the initial mapping concentration).

There are two reasons that there are more parameters to study here than in Section 8.3. The first is that the necessity for the premultiplying value, *mult*, and thus it's varying possible values. The second is that the optimal initial value for the concentration is less clear here, and certainly does not have such a simple linear relationship that was observed in the case of Section 8.3. In the case where this parameter is not explicitly varied, the results found in Section 5 will be used - specifically Figure 12, which for example shows that for an initial time of $t_0 = 0.001$, (equivalent to a $\sigma = (0.001)^{\frac{1}{2}} = 0.031$, the optimal value for α should be around $\alpha = 10$. Based on this, in the following results section, whatever parameters are not being varied will be at the following values: $N = 50$, $t_0 = 0.001$, $\alpha_0 = 10$, $tol = 10^{-0.5}$, $mult = 0.8$.

8.4.2 Results

Figure 23 shows a comparison of various parameter combinations and examines the effects on the maximum error of the solution of the diffusion equation as stated above, and the time taken to calculate that solution.

Figure 23a demonstrates the effect of varying the initial time and the initial α. As expected, for smaller initial times, larger values of α perform better. It is important to note that still, even when the mapping used is correct, a less accurate solution will be reached than if the mapping is performed on a 'flatter' function. It is interesting to note that the optimal value for the initial α value is significantly lower than predicted based on the results of Section 6. For example, when $t_0 = 0.001$, it seems that the optimal value is $\alpha_0 \approx 5$, instead of the value of 10 that was expected. This is because, in a similar fashion to the behaviour of the optimal α for the moving *tanh* and advection equation, a slightly lower value of α allows for a greater accuracy over the entire function for a range of times that it will be applied during. In fact $\alpha = 5$ is sufficiently low that it gives a reasonably accurate interpolation and derivative even when $t = 1$, meaning the grid need not adapt at all. It is noted however that this is not true when very low numbers of points are used, as illustrated in Figures 23c and 23d, and then the optimal α value is higher,

N	Optimal Tolerance, Multiplier & α_0.	Adaptive Error (log10)	Fixed Error (log10)	Adaptive Int. Error (log10)	Fixed Int. Error (log10)	Adaptive Time (s)	Fixed Time (s)	Regrids
20	$10^{-0.8}$, 0.8, 10	-1.3981	-0.3947	0.3506	1.9291	0.30	0.02	29
30	$10^{-0.8}$, 0.9, 5	-3.7859	-0.8249	-2.3069	1.6390	0.51	0.05	29
40	$10^{-0.5}$, 0.8, 5	-5.7821	-1.1570	-4.2825	1.2884	0.38	0.06	13
50	$10^{-0.5}$, 0.8, 5	-7.1924	-1.6355	-5.7583	0.8582	0.45	0.06	13
	$10^{-0.8}$, 0.8, 10	-5.6172	-1.6361	-4.2421	0.8582	0.78	0.06	29
60	$10^{-0.5}$, 0.8, 5	-7.1233	-2.1664	-5.7672	0.3465	0.55	0.06	13
	$10^{-0.5}$, 0.8, 10	-6.9995	-2.1609	-5.6288	0.3465	1.00	0.06	29
80	$10^{-0.5}$, 0.8, 5	-6.9976	-3.6271	-5.7671	-0.9229	0.73	0.07	13
100	$10^{-0.5}$, 0.8, 5	-6.9010	-5.5499	-5.7682	-2.5165	1.01	0.09	13
150	N/A	N/A	-6.9621	N/A	-6.0175	N/A	0.15	N/A
200	N/A	N/A	-6.8421	N/A	-6.0149	N/A	0.21	N/A
250	N/A	N/A	-6.7515	N/A	-6.0093	N/A	0.24	N/A
500	N/A	N/A	-6.4501	N/A	-6.0057	N/A	5.43	N/A
800	N/A	N/A	-6.2193	N/A	-6.0318	N/A	66.27	N/A

Table 2: A comparison of the error, interpolated error, and computational time required when the adaptive and fixed grid methods are applied to solve the diffusion equation as defined above, with initial condition $t_0 = 0.001$. Computations for $N > 100$ are not carried out using the adaptive method. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time.

as the significantly lower than optimal value results in too great an inaccuracy at the first time step. Thus, the default settings for the parameters being studied were changed, so that $\alpha_0 = 5$.

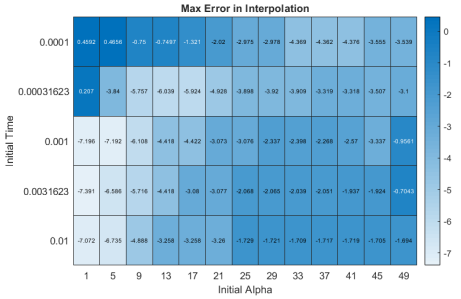
Figure 23g shows the effect of varying N and the regridding tolerance, *tol* on the solution. As one would expect, for any given value of *tol*, increasing N results in a more accurate solution. However it can be seen that the optimal value for *tol*, at least given the preset value for *mult*. Also illustrated in Figure 23h is the calculation time for each of these solutions. When the value for *tol* is decreased, it results in more regridding needing to be done. This increases computation time considerably, as can be seen by the large increases in time as one moves down the heat map. It is also clear that once machine precision is reached (appears to be at around 10^{-7} for this problem), increasing N is a waste of resources. Figure 23e shows the effect of varying *tol* and the premultiplier, *mult*. It shows that as the tolerance value is reduced, the optimal premultiplier approaches closer to 1. This is because when *tol* is reduced, it results in more regriddings being taken, and thus the necessitated reduction in α is less severe. Figure 23f demonstrates the effect of changing the *tol* value, and thus the computational cost of the increasing regridding required as *tol* is reduced.

A comparison of correctly 'tuned' parameters for an adaptive method and a fixed grid method with equal numbers of points is included in Table 2. It can be seen that, once a sufficient value of N is reached, the method will fewer regrids is actually more accurate. This is due to the increased error that results from the interpolation that must be done at each regridding step, so it makes sense that minimising the number of times this must be carried out is important. In a comparison of equal points, once again it is clear that the adaptive method does significantly better in both error at the grid points, and overall interpolated error. Indeed, for low numbers of points, for example when $N \leq 50$, the interpolated error for the adaptive grid is on the order of 5 digits. While it is true that eventually this level of accuracy is reached by the fixed grid method, the number of points needed to get to, for example 3 digits of interpolated accuracy, is roughly 3 times as much (requiring $N \approx$ 100-150 as opposed to $N = 40$). This is still quicker than the adaptive method, however the difference is not huge, and, as mentioned in Section 8.3, if applied to higher dimensional problems, this requirement for fewer points may result in genuine and significant computational savings using the adaptive method.

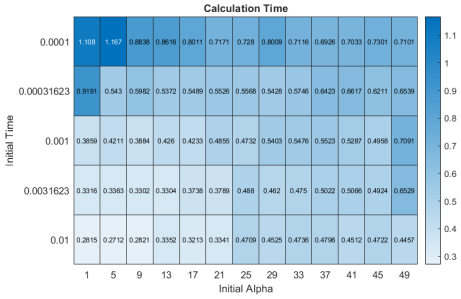
8.5 Opinion Dynamics Equation

Here the equation found in paper [19] was implemented as a good example for the use of the systems developed in this project. This equation is as follows:

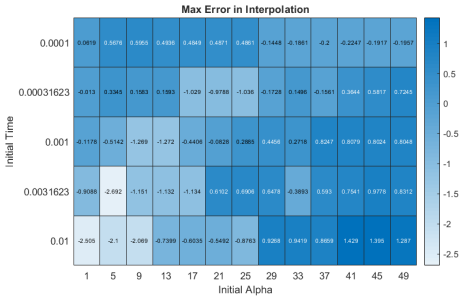
$$\rho_t(x,t) = \left(\rho(x,t)\int (x-y)\rho(y,t)\mathbf{1}_{|y-x|\leq R}dy\right)_x + \frac{\sigma^2}{2}\rho_{xx}(x,t). \tag{8.6}$$



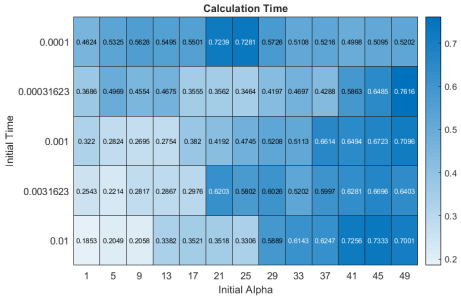
(a) The effect on the error of the interpolated solution when initial time and initial concentration are varied.



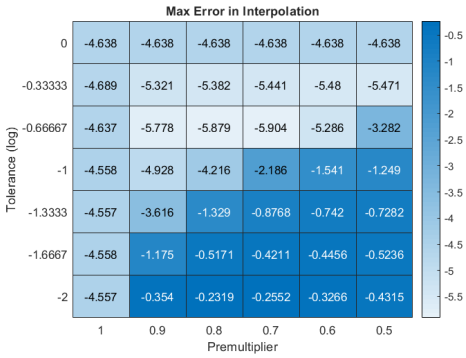
(b) The effect on the calculation time when initial time and initial concentration are varied.



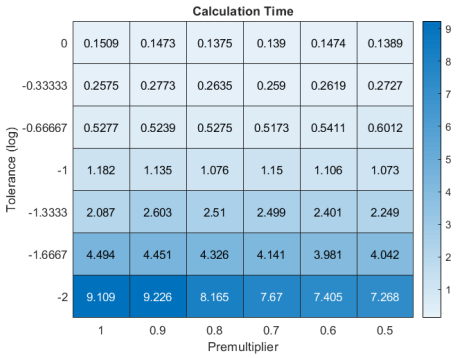
(c) The effect on the error of the interpolated solution when initial time and initial concentration are varied, using only $N = 20$ points.



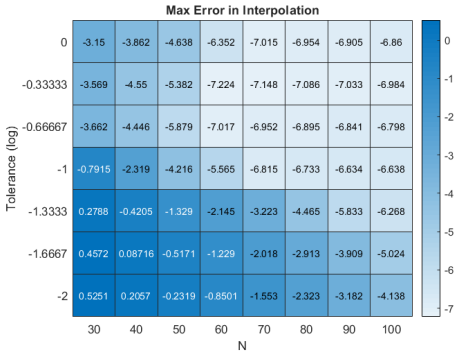
(d) The effect on the calculation time when initial time, and initial concentration are varied, using only $N = 20$ points



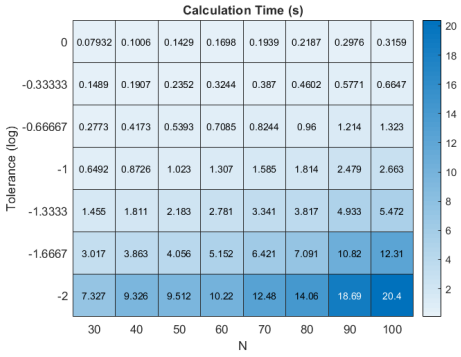
(e) The effect on the error of the interpolated solution when tolerance value and the premultiplier value are varied.



(f) The effect on the calculation time of the solution when tolerance value and the premultiplier value are varied.



(g) The effect on the error of the interpolated solution when number of points, N, and tolerance value, tol, are varied.



(h) The effect on calculation time of the solution when number of points, N, and tolerance value, tol, are varied.

Figure 23: The varying of several parameters and their effect on the accuracy of the interpolated solution of the diffusion equation, as well as the time taken to calculate that solution. Error is measured as the maximum of the 1-norm of the error over time.

The initial condition is chosen to be of the form $\rho(x,0) = Ce^{-20(x-0.5)^2}$, with C chosen such that the function is normalised to have mass (area) equal to 1.

This equation is used to model opinion dynamics within a group. It represents a set of agents on the real line, each of which are instructed to move towards the center of mass within a certain radius, R, of their position. Intuitively, this models people's opinions, which will change according to what people around you think, but you will care more about the people who think similarly to you.

The expected behaviour of this solution is for the initial peak to either spread out or sharpen, depending on the strength of the diffusion term, which is governed by the size of σ.

It should be noted that there was an error within [19] that means all of the results in the paper are only correct when rescaled[5]. These rescalings have been carried out within the code so that the results can be more readily compared to the paper.

8.5.1 Convolutions

Within (8.6) there is a convolution multiplied by an indicator function. This operation can all be done with a single matrix multiplication on ρ, in the steps as follows. Consider first a convolution without the indicator function:

$$\int_{[0,1]} (x-y)\rho(y,t)dy \approx \sum_{j=0}^{N} (y_i - y_j)\rho(y_j,t)w_j, \tag{8.7}$$

$$= \mathbf{w} * \left[(y_i - \mathbf{y}) \odot \rho\right], \tag{8.8}$$

where w_j are the appropriate integration weights discussed in Section 2, and $\odot$ represents element-wise multiplication. This is equivalent to the following action on ρ:

$$\begin{pmatrix} (\mathbf{w}).*(y_0 - \mathbf{y}) \\ \vdots \\ \vdots \\ (\mathbf{w}).*(y_N - \mathbf{y}) \end{pmatrix} \begin{pmatrix} \rho_1 \\ \vdots \\ \vdots \\ \rho_N \end{pmatrix} \approx \int_{[0,1]} (x-y)\rho(y,t)dy. \tag{8.9}$$

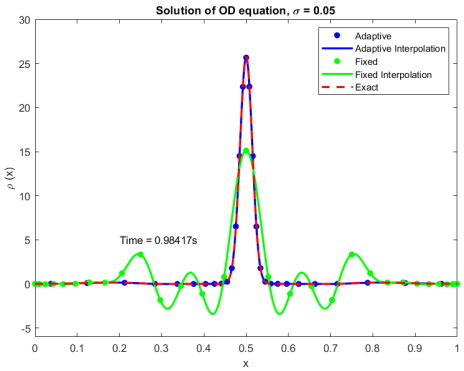
The key point here is that the full convolution and integration can be performed by multiplying ρ by a matrix which does not depend on either ρ itself or t. Now let us consider the full case, with indicator function. In this case the only difference is that, before carrying out the convolution, to interpolate from the full grid on [0,1] to a subgrid on whatever points are within the range R of each point y_i, let this subgrid have points that comprise the vector $\mathbf{y}^{(i)}$. The range of points in the subgrid is as follows: $[\max(0, y_i - R), \min(1, y_i + R)]$. However, this step can also be accomplished by matrix multiplication using the methods described in Section 4. Let the resulting matrix for a given subgrid $\mathbf{y}^{(i)}$ be called $B^{(i)}$. The integration weights $\mathbf{w}$ are calculated on $\mathbf{y}^{(i)}$, resulting in the vector $\mathbf{w}^{(i)}$, and the vector $\mathbf{y}$ in (8.9) is replaced by $\mathbf{y}^{(i)}$. This means the entire operation of calculating the convolution with the indicator function can be accomplished by multiplying ρ by a single matrix, one which is independent of ρ and t. This means the matrix need only be calculated once for each grid it is used with, saving large amounts of computation. Each row of the matrix is thus determined as follows:

$$M_{i,:} = \left(\mathbf{w}^{(i)} \odot (y_i - \mathbf{y}^{(i)})\right) * B^{(i)}. \tag{8.10}$$

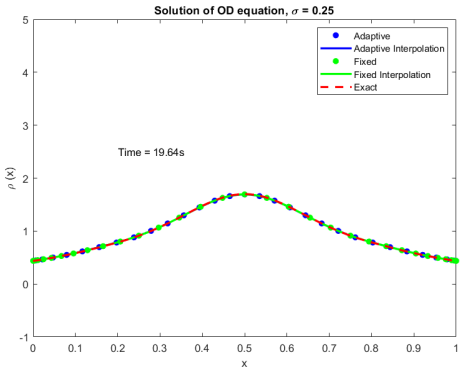
8.5.2 Parameters

This equation is expected to behave similarly to the diffusion equation, except instead of slowly spreading out a sharp peak, a sharp peak will materialise from a relatively flat initial condition.

[5] These scalings are: rescale time by multiplying by 2π, and rescale space by multiplying by $(2\pi)^{-1/2}$.



(a) An example of the solution of equation (8.6) with $\sigma = 0.05$, using a fixed and adaptive grid, each for $N = 30$ points.



(b) An example of the solution of equation (8.6) with $\sigma = 0.25$, using a fixed and adaptive grid, each for $N = 30$ points.

Figure 24: Example solutions of equation (8.6) illustrating the two possible regimes. The 'exact' solution is a fixed grid solution calculated with $N = 500$ points. Note the differing y-axes.

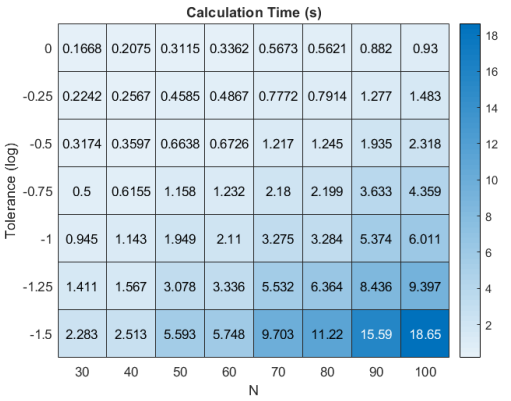
Therefore, it will include all the parameters introduced in Section 8.4, a similar range for the values of *tol* will be explored, and an inverse range for the values of *mult*. An exception is that instead of t_0, which controlled the initial steepness of the problem in Section 8.4, this equation has σ, which works in a similar way, in that reducing σ towards 0 results in increasingly steep final distributions. The default settings for parameters in the following sections, unless explicitly mentioned otherwise, is as follows: $N = 50$, $\sigma = 0.1$, $tol = 10^{-0.5}$, $\alpha_0 = 2$, $mult = 1.1$

8.5.3 Results

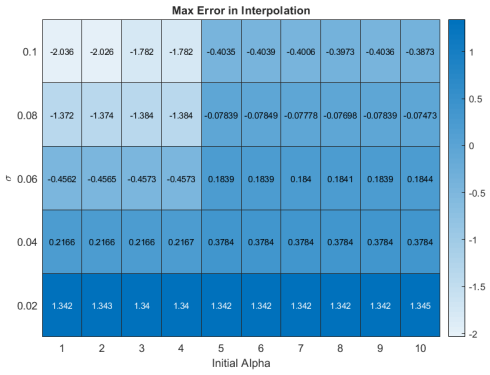
Examples of the solution for $\sigma = 0.05$ and $\sigma = 0.25$ (rescaled), are shown in Figure 24. As can be seen, when σ is large, the solution remains relatively flat, and there is little to be gained by using an adaptive method. When σ is very low however, the solution becomes a very steep spike, and here the adaptive method performs much better than an equivalent number of fixed points. This is clearly shown in Figure 24a. Therefore, the remainder of the results will be discussed on values of σ that result in the steep regime. In order to compare accuracies of the adaptive method, it was compared against a fixed spectral method solution of (8.6) using $N = 500$. In the absence of an analytic solution, this is the best alternative, and given the results in the previous sections, it seems reasonable to expect this to be accurate within rounding errors. Thus, all methods were interpolated onto an equi-spaced grid of 1000 points, and then compared with the interpolation of the $N = 500$ solution using the fixed grid. It is important to note that due to the extra steps involved with interpolating a non-exact solution for comparison, the errors found seem to be much larger than in previous experiments. However, they still show when the adaptive method shows a good approximation for the solution especially in comparison to the fixed grid method, as well as when it fails. The results obtained when several different parameters were varied is shown in Figure 25.

The first figures show the effect that varying N and *tol* has on the accuracy and calculation time. A useful observation is that *tol* seems to have very little effect on the computed error. Since *tol* still has a very large effect on the calculation time, this implies very low tolerances are preferable. Another interesting affect is the 'striping' of the results in Figure 25a. This seems to be due to a regridding error that occurs when N is an even multiple of 10, although the reasons for this are unclear. Figures 25c and 25d show the effect of varying σ and α_0. It clearly demonstrates that the optimal α_0 is less than 3, and in the following discussions, we have continued to take it as $\alpha_0 = 2$, reasoning that the higher end of what is suitable for the initial condition allows for increased accuracy as the function becomes steeper. It also demonstrates that as σ decreases, the resulting error increases. This is because it results in a steeper function, which naturally is harder to calculate accurately, even with appropriate mappings. Figures 25e and 25f show the

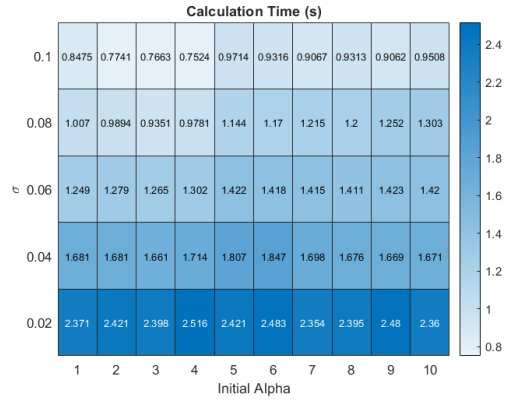
(a) A comparison of resulting error of the solution when N and tol are varied.



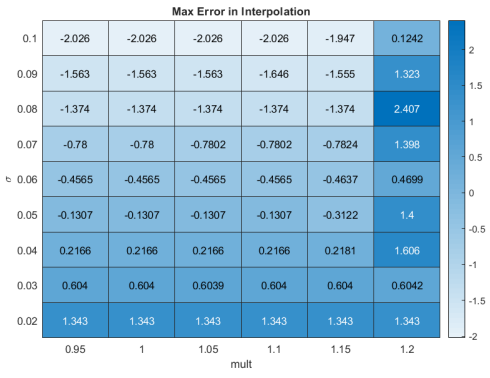
(b) A comparison of resulting error of the solution when N and tol are varied.



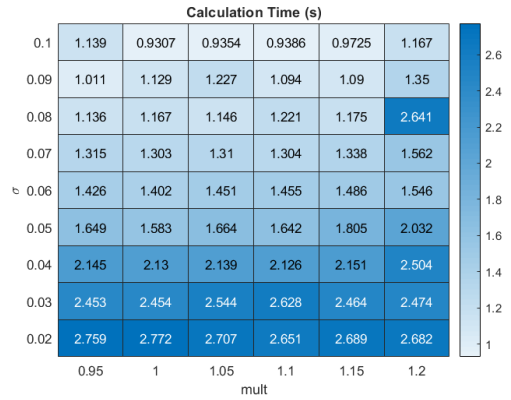
(c) A comparison of resulting error of the solution when σ and α_0 are varied.



(d) A comparison of computation time of the solution when σ and α_0 are varied.



(e) A comparison of resulting error of the solution when σ and $mult$ are varied.



(f) A comparison of computation time of the solution when σ and $mult$ are varied.

Figure 25: A comparison of the effects of various parameters on the accuracy of the solution of the Opinion Dynamics equation (8.6). Error is measured as the maximum of the 1-norm of the error over time.

N	Optimal tol & mult	Adaptive Error (log10)	Fixed Error (log10)	Adaptive Time (s)	Fixed Time (s)	Regrids
20	$10^{-0.5}$, 1.3	1.6076	2.8190	0.26	0.01	14
30	$10^{-0.7}$, 1.05	-0.2929	2.4233	0.32	0.01	14
40	$10^{-0.5}$, 1.05	-1.1431	1.8671	0.46	0.02	14
50	$10^{-0.7}$, 1.05	-2.0258	1.1694	0.61	0.03	14
60	$10^{-0.5}$, 1.05	-2.2433	0.3479	0.86	0.04	14
70	$10^{-0.7}$, 1.05	-2.3965	-0.6066	1.11	0.05	14
80	$10^{-0.7}$, 1.05	-2.7466	-1.6982	1.92	0.06	21
90	$10^{-0.7}$, 1.05	-2.8225	-2.5670	2.57	0.07	21
100	$10^{-0.5}$, 1.05	-2.7561	-2.7300	2.01	0.09	14
150	$10^{-0.5}$, 1.05	-2.8769	-3.5119	4.37	0.21	14
200	$10^{-0.5}$, 1.05	-3.2275	-3.0822	7.32	0.38	14
300	N/A	N/A	-3.8662	N/A	1.00	N/A
500	N/A	N/A	-7.3196	N/A	5.66	N/A
800	N/A	N/A	-4.0563	N/A	28.62	N/A

Table 3: A comparison of adaptive and fixed methods in calculating the solution of (8.6). Computations for $N > 100$ are not carried out using the adaptive method. Interpolation was done onto a uniform grid of 1000 points. Error is measured as the maximum of the 1-norm of the error over time.

effects of varying *mult* and σ. As can be seen, when σ is increased, an increasing value of *mult* can be used. However, it is interesting to note that in this case, it seems the addition of the parameter *mult* appears to have little effect, as the built in optimisation step of the regridding seems to produce appropriately increasing values of α, even in the case when a value of $mult < 1$ is used. Based on all of the study carried out above, the optimal parameters were decided to be $tol = 10^{-0.5}$, $mult = 1.1$, and $\alpha_0 = 2$. A table detailing the accuracies of the solution in comparison to fixed grids of the same size is shown in Table 3.

This table demonstrates the advantages and disadvantages of using an adaptive method in order to solve (8.6). As can be seen, for any given number of points N, the adaptive solution clearly outperforms the fixed solution, by around 2 decimal places until rounding errors begin to take over.

8.6 Pedestrian Dynamics Equation

The PDE to be studied in this subsection is used with [9] to model pedestrian movement in a corridor. It is a Fokker-Planck equation derived within [9] from an SDE (Stochastic Differential Equation), and presents some behaviour that it would be useful to test this method on. The full equation is:

$$\partial_t \tilde{\rho} = \operatorname{div}\left(\Sigma \nabla \tilde{\rho} - \tilde{\rho}\tilde{F}(\tilde{\rho})\right), \tag{8.11}$$

(the $\tilde{}$ is due to a normalisation carried out on ρ) where

$$\tilde{F}(\tilde{\rho}) = \tilde{f}(\tilde{\rho})e_1, \quad \tilde{f}(\tilde{\rho}) = \nu_{\max}(1-\tilde{\rho}). \tag{8.12}$$

This has boundary conditions:

$$\tilde{j}\dot{n} = -a(1-\tilde{\rho}) \quad (x_1, x_2) \in \Gamma_{in}, \tag{8.13}$$

$$\tilde{j}\dot{n} = b\tilde{\rho} \quad (x_1, x_2) \in \Gamma_{out}, \tag{8.14}$$

$$\tilde{j}\dot{n} = 0 \quad (x_1, x_2) \in \Gamma_N. \tag{8.15}$$

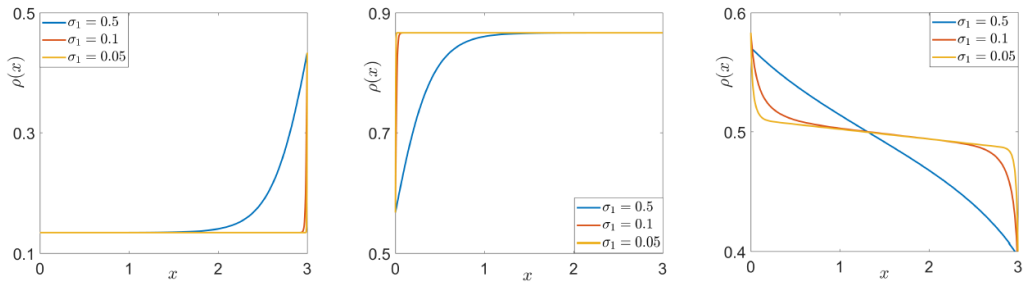


Figure 26: Examples of the steady states of the FP equation in 1D for $\sigma = 0.5, 0.1, 0.05$ and (left) the influx limited phase - $a = 0.2, b = 0.4$, (center) the outflux limited phase - $a = 0.4, b = 0.2$, and (right) the maximal current phase - $a = 0.9, b = 0.975$.

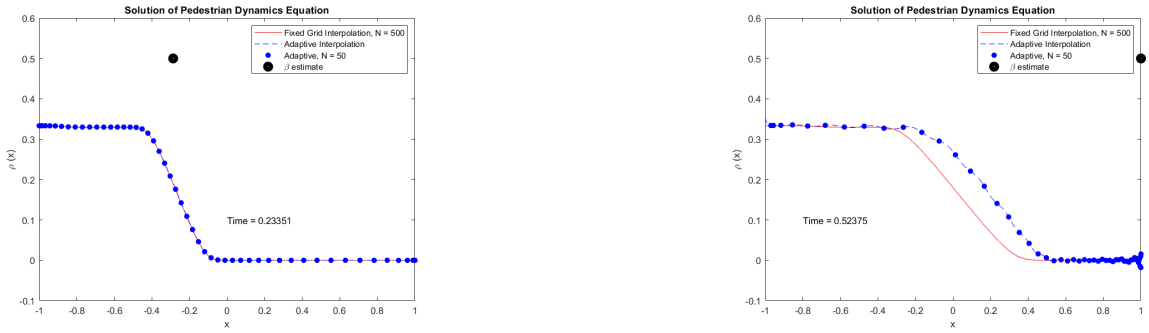
The 1D version of this equation shall be studied here, that is:

$$\partial_t \rho = \partial_x \left(\sigma \partial_x \rho - \nu_{\max} \rho (1 - \rho) \right). \tag{8.16}$$

This equation is non-linear, and is expected to produce very fine boundary layers at the left and right sides of the boundary, as the solution continues, and depending on the values of a and b.

The expected solutions are shown within [9], and are as shown in Figure 26.

The adaptive method was applied to this equation in the influx limited phase (when $a < b$), and a snapshot of the solutions is shown in Figure 27. As can be seen, the adaptive solution performs well initially, but after time $t \approx 0.5s$, the method fails entirely to produce something accurate. This is because at this time, the true solution begins to 'build up' at the boundary $x = 1$, leading to a very steep solution, or 'boundary layer' at this point. This means that the grid finding routine now finds this point as being the optimal place to center around, leading to very inaccurate results. As will be discussed later, this is a problem that cannot be overcome with the current method being used, though there are more sophisticated techniques that may be applicable ([1]).



(a) Solution of (8.11) at time $t = 0.233$.

(b) Solution of (8.11) at time $t = 0.524$.

Figure 27: Solution of (8.11), with $\sigma = 0.05$, $a = 0.5$, $b = 0.6$. The x coordinate of the black dot represents the value β the grid is currently using.

9 Conclusion

In Section 7, two demonstration problems were used in order to study a more basic problem before moving onto the more complex problem of PDEs. From this, the expected range of certain parameters for different types of problem was learned, and the introduction of a new parameter, *mult* was introduced, which in certain circumstances, greatly improved performance. As has been shown throughout Section 8, the use of an adaptive method to solve PDEs with steep solutions has proven to present considerable gains in accuracy when compared to similar numbers of points in a non-adapting grid. Although the additional computation required means that the extra effort is not necessarily worth it in all of the applications studied here, the potential for computational savings is clear. Firstly, the algorithm used here to calculate new grids was not optimised in any particularly stringent way, so it is not unreasonable to expect that large computational savings could be made if more time was spent optimising this step. The second, clearer way that this method will prove more efficient is when it is applied to higher dimensional PDEs. As demonstrated several times, when N gets very large, the computational times increase rapidly. This problem becomes much more salient in higher dimensions. If a given solution requires 10 times the amount of points to reach a given accuracy in 1D using a fixed solution, then it would take 100 times as many points in 2D, 1000 times as many in 3D, and so on. In this case, the adaptive method should produce considerable advantages. There are some drawbacks of this method. As demonstrated by several of the cases within Section 8, there is a requirement of knowing some behaviour about the expected solution of the PDE that the adaptive method is being applied to, at least in order to properly tune the method. This is not always possible, and the hassle this involves may again mean that it is not worth the time. This could possibly be addressed by incorporating more information about the current solution in order to work out the appropriate mapping at regrids. (For example, the maximum of second derivative might be utilised to provide some sort of estimate for α.)

Another clear problem is identified in Section 8.6, where the method fails due to there being more than one region of 'steepness'. This could be addressed by splitting the domain into separate subdomains [1].

Future Work As noted above, this method is of most use when applied to higher order PDEs, and so an obvious next step is to apply it to these problems. Additionally, the use of several different subdomains might be used in order to still produce accurate results when there is more than one steep region, or 'layer', could be utilised for problems such as (8.11).

References

[1] A. Bayliss, M. Garbey, and B. J. Matkowsky. Adaptive pseudo-spectral domain decomposition and the approximation of multiple layers. *J. Comput. Phys.*, 119(1):132–141, 1995.

[2] A. Bayliss, D. Gottlieb, B. J. Matkowsky, and M. Minkoff. An adaptive pseudo-spectral method for reaction diffusion problems. *J. Comput. Phys*, 81(421), 1989.

[3] A. Bayliss and B. J. Matkowsky. Fronts, relaxation oscillations, and period doubling in solid fuel combustion. *J. Comput. Phys*, 71(147), 1987.

[4] A. Bayliss and E. Turkel. Mappings and accuracy for chebyshev pseudo-spectral approximations. *Institute for Computational Mechanics in Propulsion*, 2000.

[5] J. P. Berrut, R. Baltensperger, and H. D. Mittelmann. Recent developments in barycentric rational interpolation. *Internat. Ser. Numer. Math.*, 151:27–51, 2005.

[6] J.-P. Berrut and L. N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 2004.

[7] J. P. Boyd. Chebyshev & fourier spectral methods. *Lecture Notes in Engineering*, 49, 1989.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, New York, 1989.

[9] S. N. Gomes, A. M. Stuart, and M.-T. Wolfram. Parameter estimation for macroscopic pedestrian dynamics models from microscopic data. *J. Appl. Math.*, 79(4):1475–1500, 2019.

[10] D. Gottlieb, M. Y. Hussaini, and S. A. Orszag. *Introduction: Theory and Applications of Spectral Methods*. SIAM, 1984.

[11] D. Gottlieb and S. A. Orszag. Numerical analysis of spectral methods. *C.B.M.S.-N.S.F. Conference Series in Applied Mathematics*, 1977.

[12] N. Gröwe-Kuska and W. Römisch. *Stochastic unit commitment in hydro-thermal power production planning*. Preprints aus dem Institut für Mathematik. Humboldt-Universität zu Berlin, Institut für Mathematik, 2001.

[13] V. D. Leseikin. *Grid Generation Methods*. Springer, 2017.

[14] H. E. Salzer. Lagrangian interpolation at the Chebyshev points $x_{n,\nu} = \cos(\nu\pi/n), \nu = O(1)n$; some unnoted advantages. *Comput. J.*, 15:156–159, 1972.

[15] S. Sarra. Chebyshev pseudospectral methods for conservation laws with source terms and applications to multiphase flow. *Graduate Thesis, Dissertations, and Problem Reports*, 1642, 2002.

[16] T. Shiina and J. R. Birge. Stochastic unit commitment problem. *International Transactions in Operational Research*, 11(1):19–32, 2004.

[17] T. W. Tee and L. N. Trefethen. A rational spectral collocation method with adaptively transformed chebyshev grid points. *J. Sci. Comput.*, 28(5):1798–1811, 2006.

[18] L. N. Trefethen. *Spectral Methods in MATLAB*. SIAM, 2000.

[19] C. Wang, Q. Li, W. E, and B. Chazelle. Noisy hegselmann-krause systems: Phase transition and the 2r-conjecture. *Journal of Statistical Physics*, 2017.