Australian Government
Department of Defence
Science and Technology

Rapid solution of the Schrödinger equation: Towards a study of the utility of the Bohm filter

Daniel L. Whittenbury[1 2], *Ayse Kizilersu*[1 2],
Anthony W. Thomas[1 2] *and Samuel P. Drake*[3]

[1] **School of Physical Sciences, University of Adelaide**

[2] **Centre for the Subatomic Structure of Matter (CSSM)**

[3] **Cyber and Electronic Warfare Division**

Defence Science and Technology Group

DST-Group–TR–3513

ABSTRACT

The second project report for the Efficient Generation and Evolution of Probability Density Maps project is reproduced as a Defence Science and Technology Group technical report. Here we focus on solving the Schrödinger equation numerically for several simple potentials using Fourier and Chebyshev pseudo-spectral methods. The report has been written in such way to be more pedagogical rather than complete.

Published by

Cyber and Electronic Warfare Division
Defence Science and Technology Group
PO Box 1500
Edinburgh, South Australia 5111, Australia

Telephone: 1300 333 362

AR-017-243
July, 2018

APPROVED FOR PUBLIC RELEASE

Rapid solution of the Schrödinger equation: Towards a study of the utility of the Bohm filter

Executive Summary

The Bohm filter as proposed by Drake [1] is an attempt to confront some of the shortcomings of the well known Kalman filter and its variants. The underlying idea is to use a Schrödinger equation to model the motion of an object we wish to track. This report is a first step towards developing a filter based on such an approach. Analytic solutions to the Schrödinger equation are only known for a few well known potentials. As a prerequisite to constructing a filter based on this approach we will require accurate and efficient methods to solve the second order differential equation for potentials relevant to tracking problems. This aspect of implementing a Bohm filter is the focus of this report. Here we concentrate on finding numerical solutions to the Schrödinger equation for several time-independent potentials using Fourier and Chebyshev pseudo-spectral methods.

The report begins with a brief introduction to several relevant topics such as the reduction of the time-dependent Schrödinger equation to the time-independent equation for time-independent potentials, collocation interpolation and numerical quadrature. We then proceed to introduce Fourier and Chebyshev pseudo-spectral methods. We then apply them to several time-independent Schrödinger equations. We then conclude and discuss future research directions.

The report has been written in a pedagogical style with many illustrative examples given. For this reason it may also be of use to a wider audience interested in solving other unrelated differential equations. In the appendices we have provided a few additional pieces of information. In particular, we have written several simple MATLAB® scripts which illustrate various ideas presented in the report. These scripts are stored in Govdex and are available upon request. All figures found herein can be obtained using these scripts with either no or only minimal modifications required.

THIS PAGE IS INTENTIONALLY BLANK

Authors

Daniel L. Whittenbury
School of Physical Sciences, University of Adelaide

Daniel L. Whittenbury recently obtained his PhD in theoretical physics from the University of Adelaide in 2015. His dissertation research focussed on a topic at the border of nuclear structure and nuclear astrophysics. The underlying theme of which was an investigation of the equation of state of strongly interacting matter and the modelling of cold neutron stars. In addition to this he also holds a Bachelor of Science from Flinders University, a Graduate Diploma of Mathematical Science and a Master of Science in theoretical physics both from the University of Adelaide. Currently he works as a post-doctoral researcher in the Physics department of the University of Adelaide.

Anthony W. Thomas
School of Physical Sciences, University of Adelaide

Anthony W. Thomas is a theoretical physicist who holds the Elder Chair of Physics at the University of Adelaide. He serves as Associate Director of the Australian Research Council (ARC) Centre of Excellence for Particle Physics at the Terascale (CoEPP), as well as Director of the Adelaide node. Within the University of Adelaide he is Director of the University Research Centre for Complex Systems and the Structure of Matter (CSSM). From 2009-15 he held an ARC Australian Laureate Fellowship. During the 6 year period from 2004-2009, Professor Thomas served as Chief Scientist and Associate Director for Theoretical and Computational Physics at the US Department of Energy's Thomas Jefferson National Accelerator Facility. He is currently the Vice-Chair of the Asian Nuclear Physics Association and past-Chair of the IUPAP Working Group on International Co-operation in Nuclear Physics, having served as its inaugural chair for the first 6 years.

Ayse Kizilersu
School of Physical Sciences, University of Adelaide

Ayse Kizilersu is a senior researcher in the physics department at Adelaide University. She was the winner of the best student in both her final undergraduate year and for her masters in the physics department at Istanbul University. She was awarded a British Council Research Scholarship and obtained her PhD in theoretical particle physics from the Centre for Particle Theory, University of Durham, England. After her PhD she completed her habilitation and was award the title of Associate Professor. Her research has spanned a range from topics from theoretical nuclear physics to stock market analysis. She is currently working on accurate probability modelling of rare events with incomplete data.

Samuel P. Drake
Defence Science and Technology Group

Sam Drake is a senior research scientist in the Defence Science and Technology (DST) group and an adjunct lecturer in the physics department at Adelaide University. He obtained a Bachelor of Science with first class honours majoring in physics from the University of Melbourne and completed his PhD in General Relativity at the University of Adelaide. After doing post-doctoral research in the Dipartmento di Fisica Galileo Galilei at the University of Padova, Italy he began working as a research scientist in DST group. The range of work he has been involved in ranges from global satellite navigation to the autonomous control of unmanned aerial vehicles (UAVs). He is currently working on geolocation algorithms and applications of relativity to satellite dynamics and signal processing.

Contents

List of Figures

List of Tables

Glossary

SVM	Stochastic variational method
TDSE	Time-dependent Schrödinger equation
TISE	Time-independent Schrödinger equation
UAV	Unmanned aerial vehicle

Notation

A_n	The nth coefficient in a wave function expansion.
χ_j	The jth test function.
C_j	The jth cardinal basis function.
$D_N^{(\ell)}$	The ℓth order differentiation matrix.
$\boldsymbol{e}_{n+1}$	Unit vector.
ϵ	Quartic coupling.
E_n	Energy eigenvalue.
g	Acceleration due to gravity.
$\hat{f}_j$	The jth coefficient of the function $f(x)$.
H	Hamiltonian.
$H_n(x)$	The nth Hermite polynomial.
$I_n(f)$	The approximation the integral of $f(x)$ with an n-point quadrature rule.
$\hbar$	Planck's reduced constant.
J	Jacobi matrix.
k_N	Leading coefficient of an nth degree polynomial.
$\ell(x)$	Node polynomial.
$\ell_j(x)$	The jth Lagrange basis polynomial.
L	Linear differential operator.
$L^2([a,b])$	The second Lebesgue space.

Notation

$L^2_w([a,b])$	The second weighted Lebesgue space.
λ_n	The nth eigenvalue.
$L_n(x)$	The nth Laguerre polynomial.
m	Mass.
μ	Mean.
μ_0	Zeroth moment.
$p_N(x)$	An algebraic polynomial of degree N.
$P_n(x)$	The nth Legendre polynomial.
P_N	Projection operator.
$\phi_j(x)$	The jth degree polynomial of a set of orthogonal polynomials.
$\boldsymbol{\phi}$	Vector of orthogonal polynomials.
$\Psi(x,t)$	Time-dependent wave function.
$\psi(x)$	Time-independent wave function.
$q(x)$	Quotient polynomial.
$r(x)$	Runge's function in Sec. 5.4 and the remainder polynomial in Sec. 6.
R	The residual.
$R_n(f)$	The residual from approximating the integral of $f(x)$ with an n-point quadrature rule.
σ	Standard deviation.
T	Tridiagonal matrix.
$T_n(x)$	The nth Chebyshev polynomial.
$u_n(x)$	The nth eigenfunction.
$V(x,t)$	Time-dependent potential.
$V(x)$	Time-independent potential.
$V_{AHO}(x)$	Anharmonic or quartic potential
$V_{\text{HO}}(x)$	Harmonic oscillator potential.
$V_L(x)$	Linear potential.
V	Vandermonde matrix.
ω	Angular frequency.
$w(x)$	Weight function.
x_j	The jth grid point.

1 Introduction

Noisy measurements are prevalent throughout modern engineering and experimental science, complicating the accurate determination of a system's state or an object's location and velocity. Filtering noise is routinely performed using an assortment of filters, among them the most commonplace is the Kalman filter and its variants. These filters are used in very diverse applications which range from following a finger moving on a laptop's track pad to locating an unmanned aerial vehicle (UAV). The Kalman filter has been successfully applied in many varied situations, in part due to its simplicity both conceptually and computationally. However, it is somewhat limited by its assumptions of a linear motion model and measurement errors which follow a Gaussian distribution. In real-world applications, these conditions are not always met as motion is rarely linear nor are the measurement errors typically Gaussian.

In this report we aim to develop Sam Drake's original idea, the Bohm filter, as communicated in his private notes [1]. Our intent for this project is an investigation into the feasibility of the Bohm filter from conception, through to application to realistic tracking problems. In this phase of the project we focus on the solution of the Schrödinger equation. The Bohm filter was suggested to address some of the inadequacies of the Kalman filter. It is formulated as a Bayesian filter (making use of Bayes' rule), just like the Kalman filter, but the fundamental idea is to use Schrödinger's equation for the motion model where Planck's constant can be varied based on the experimental uncertainty of the measured location and velocity of the object being tracked.

Connecting classical and quantum mechanics is a long standing problem in the foundations of quantum theory and has very profound implications for our understanding of the nature of reality and how we should interpret quantum theory. Thus, the rigorous justification of using quantum mechanics to describe a classical system is conceptually difficult. However, it has been demonstrated in a series of works, see Refs. [2–8], that a classical system with noise can be described by an equation, which takes exactly the same mathematical form as the Schrödinger equation, the equation which underlies non-relativistic quantum mechanics. This comes under the name of Stochastic Variational Method (SVM) and takes the form of a generalised variational principle in order to unify classical and quantum mechanics in the same mathematical framework. We take this unified treatment as a justification to examine the usefulness of the Bohm filter and leave further scrutiny of the quantum–classical connection to later exploration.

As we are taking a more utilitarian view, we will focus on the practical implementation of the Bohm filter and with the aim of extending the current investigation to examine the usefulness of the Bohm filter to tracking problems in the next phase of the project. Obviously, each tracking problem will be described by a different potential and in general they will not be exactly solvable or time-independent. An undeniable advantage of the Kalman filter over the present filter is that much of the calculation for the Kalman filter can be simplified analytically because of the assumptions of linear motion and Gaussian measurement errors. Whereas, for the current filter much of the calculation will need to be performed numerically. Hence, the Bohm filter will necessarily be much more demanding than the Kalman filter computationally. To reduce the additional computational overhead of the Bohm filter we will need to use numerical methods which minimise the number of floating point operations and the amount of memory used for a given accuracy when solving the Schrödinger equation. The smooth nature of wave

functions leads us naturally to consider a class of methods called spectral methods, in particular the subclass of pseudo-spectral (collocation) methods. This will be the most important aspect of implementing the Bohm filter so we will develop this in immense detail.

In fact, the majority of this report will be concerned with introducing the ideas behind one of the most useful and powerful approaches to solving differential equations when the unknown function is anticipated to be smooth. We will not attempt a full review of the current state of the art regarding this class of methods, but we will rather opt for a more pedagogical and illustrative approach with a number of explicit examples which should be understandable to a wider audience.

We will consider the Schrödinger equation with several potentials known to be analytically solvable with the view of gauging the efficiency and accuracy of these methods. These potentials would not necessarily be of any use in any real-world tracking problem, but rather provide us with some useful models. The possibility of extending to more realistic potentials, which may be used in real-world tracking problems, is reserved for future research.

The outline of this report is as follows. In section 2 we introduce the time-dependent Schrödinger equation and its reduction to the time-independent Schrödinger equation for static potentials. The material of the following sections is all very well known and can be found in a number of good textbooks on approximation theory, orthogonal polynomials, analysis, numerical methods and spectral methods. In section 3 we summarise some of the important details of spectral methods and in particular motivate the use of the so-called pseudo-spectral methods. Section 4 is devoted to introducing the orthogonal polynomials and some of their important properties. In Section 5 we will introduce some basic concepts such as the idea of a collocation interpolation and define various approaches to polynomial interpolation. Section 6 discusses numerical quadrature. Sections 7 and 8 introduce the Fourier and Chebyshev pseudo-spectral methods and their application to a few simple potentials. The report then finishes with a discussion and conclusions in Section 9.

Throughout this report all numerical calculations were performed using MATLAB® [9]. Functions implemented in MATLAB® to investigate the pseudo-spectral methods can be found in the cited literature. However, various example scripts have been written which can produce the figures in this report. They are stored in Govdex and are available upon request. The appendices also contain useful tid bits of information.

There is a vast literature dedicated to spectral methods, but through the course of this research project we have found the following resources particularly useful:

- The two most useful references were Ref. [10], which is focussed entirely on pseudo-spectral methods and Ref. [11] on approximation theory.
- There are a number of classic books on spectral methods, in particular, we draw the readers attention to Boyd [12], Funaro [13], Canuto *et al* [14, 15], Fornberg [16] and also Shen *et al* [17].
- Also the very useful mathematical handbook [18], particularly chapters 22 and 25 were incredibly helpful.

2 Solution of the TDSE

We are interested in developing Drake's Bohm filter to be used to filter noisy measurements and track an unknown moving object. The idea of the Bohm filter (or tracker in this context) is an alternative to the Kalman filter. To build a Bayesian tracker we need a motion model and a sensor (or error) model. The Bohm filter uses Schrödinger's equation for the motion model, where Planck's constant can be varied based on the experimental uncertainty of the measured location and velocity of the object being tracked. Each tracking problem will be described by a different potential and in general will not be exactly solvable or time-independent. To initiate an investigation into how we may implement such a tracker in practice, we first need a fast and accurate numerical method to solve the time-dependent Schrödinger equation (TDSE). This will be the most important part of the Bohm tracker.

The TDSE is

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{x},t) = \left(-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{x},t)\right)\Psi(\mathbf{x},t) \equiv H\Psi(\mathbf{x},t)\ , \tag{1}$$

where all the symbols have their usual meaning, i.e., $\Psi(\mathbf{x},t)$ is the time-dependent wave function, $\hbar$ is the reduced Planck's constant, $V(\mathbf{x},t)$ is the interaction potential, and H is the Hamiltonian[1]. To begin with we will investigate a few familiar time-independent potentials. These can be used as simple test cases to understand the efficiency and accuracy of the numerical methods we use. Moreover, at a later stage these more pedagogical potentials will allow us to evaluate the effectiveness of a Bayesian tracker based on the Schrödinger equation.

All the potentials we will consider in this report, as mentioned, will have no time dependence, i.e., $V(\mathbf{x},t) = V(\mathbf{x})$. We will discuss the numerical solution of the TDSE equation for the familiar harmonic, anharmonic and linear potentials, namely

$$V_{\mathrm{HO}}(x) = \frac{1}{2}m\omega^2x^2 \quad , \tag{2}$$

$$V_{\mathrm{AHO}}(x) = \frac{1}{2}x^2 + \epsilon x^4 \quad , \tag{3}$$

and

$$V_{\mathrm{L}}(x) = \begin{cases} \infty & , \quad x < 0 \\ \\ mgx & , \quad x > 0 \end{cases} \ . \tag{4}$$

These potentials could possibly be used as part of a Bayesian tracker to track an object that moves harmonically, almost harmonically or is falling in a gravitational potential. These potentials would not necessarily be of any use in any real-world tracking problem, but are rather useful toy models.

All of the potentials under consideration are time independent and therefore we can use the separation of variables technique. For a general tracking problem we will not necessarily have this simplification and may need to consider a modified approach to what we will discuss here.

[1] If the reader is unfamiliar with basic quantum mechanics see one of the many good undergraduate texts on the subject, e.g., Powell and Crasemann [19] or Bohm [20].

However, many of the same ideas will carry over to this situation. The time evolution of each eigenfunction is then simply given by an exponential factor

$$\Psi_n(\mathbf{x}, t) = \psi_n(\mathbf{x}) \exp(-iE_n t/\hbar) \ , \tag{5}$$

where $\Psi_n(\mathbf{x}, 0) = \psi_n(\mathbf{x})$ is the solution to the time-independent Schrödinger equation (TISE)

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V(\mathbf{x})\right) \psi_n(\mathbf{x}) = E_n \psi_n(\mathbf{x}) \ . \tag{6}$$

To obtain the spatial part of the solution of the TDSE, the time-independent Schrödinger equation (TISE) must be solved. This is an eigenvalue problem, which in general is not exactly solvable. However, the exact solution can be obtained for the harmonic oscillator (Appendix A.1) and linear (Appendix A.2) potentials, but not for the anharmonic potential. Nonetheless, perturbation theory combined with Padé summation can be used to extract an approximate solution for the eigenvalues, which we will not discuss, see Refs. [21–23].

A general solution to the TDSE equation can be built from a linear combination of these solutions. Thus, if we know the initial state of the system is for example a Gaussian,

$$\Psi(x, 0) = f(x) = Cexp(-(x-\mu)^2/2\sigma^2) \quad , \tag{7}$$

we can project out this solution onto the eigenfunctions $\psi_n(x)$ using the orthogonality of the eigenfunctions,

$$A_n = \frac{\int_{-\infty}^{\infty} f(x)\psi_n(x)dx}{\int_{-\infty}^{\infty} |\psi_n(x)|^2 dx} \tag{8}$$

obtaining coefficients which allow us to write the initial state as

$$f(x) = \sum_{n=0}^{\infty} A_n \psi_n(x) \quad . \tag{9}$$

This initial state then evolves in time by the exponential time evolution of each of the eigenfunctions, such that

$$\Psi(x, t) = \sum_{n=0}^{\infty} A_n \psi_n(x) e^{-iE_n t/\hbar} \quad . \tag{10}$$

This straightforward evolution of the initial state in time is a result of the time independence of the chosen potentials. Obviously, in practice we will be working with a finite number of eigenfunctions and eigenvalues. Therefore the summation in Eqs. 9 and Eq. 10 will be truncated. Ideally, it will be at an order where the coefficients A_n are small and below some acceptable level of tolerance.

Above we have made the assumption that the initial state is Gaussian. This initial state is what would be obtained through measurements and in general it may not be Gaussian. It is not necessary to make this restriction to an initially Gaussian wave function, but we do so for simplicity. There is no extra complication to extend to a non-Gaussian error model as the integral in Eq. 8 will be done using numerical quadrature.

To solve the TISE we will use Fourier and Chebyshev pseudo-spectral methods. We may, however, need to extend to other polynomial bases in more realistic tracking problems depending on the potential for a particular tracking problem. Each tracking potential will need to be

investigated separately. Possible polynomials worth considering are the Legendre polynomials defined on $[-1, 1]$, which can easily be extended to any finite domain, Laguerre polynomials on the semi-infinite domain and also Hermite polynomials on the infinite domain. We will be restricting our attention to just one dimension, but in higher dimension generalisations it in may be advantageous to use a combination of pseudo-spectral methods, e.g., in two dimensions use of a Fourier grid for a bearing coordinate and a Chebyshev grid for the range coordinate.

3 The numerical problem at hand

We are interested in efficiently obtaining an accurate numerical solution to the Time-Independent Schrödinger equation (TISE), Eq. 6. More abstractly, this is an eigenvalue problem of the form,

$$Lu_n(x) = \lambda_n u_n(x) \ , \tag{11}$$

where L is a linear differential operator (the Hamiltonian), λ_n is an eigenvalue of L associated with the eigenfunction $u_n(x)$ defined on the interval $\Omega = [a, b]$. In the special case of L being self-adjoint (hermitian), its eigenvalues are real and the eigenfunctions corresponding to different eigenvalues are orthogonal. The eigenfunctions are subject to two homomgeneous boundary conditions

$$\alpha_1 u_n(a) + \beta_1 u_n'(a) = 0 \tag{12}$$
$$\alpha_2 u_n(b) + \beta_2 u_n'(b) = 0 \tag{13}$$

where if

- $\beta_i = 0$, we have a Dirichlet boundary condition;
- $\alpha_i = 0$, a Neumann boundary condition;
- $\alpha_i \neq 0$ and $\beta_i \neq 0$, a Robin boundary condition.

When using Chebyshev methods we will only be interested in Dirichlet boundary conditions. However, when using Fourier methods the boundary conditions we impose will be periodic, $u(a) = u(b)$.

An acceptable solution to this problem (which we denote by $\overline{u}_n(x)$ and $\overline{\lambda}_n$) is one that exactly satisfies the boundary conditions and makes the residual

$$R \equiv L\overline{u}_n(x) - \overline{\lambda}_n \overline{u}_n(x) \ , \tag{14}$$

that is the error, small. To define what we mean by small we use the method of weighted residuals.

What we do is look for solutions in a finite dimensional subspace of some Hilbert space. As an example, the trigonometric polynomials $\{e^{ikx}\}_{k=-\infty}^{\infty}$ are dense in $C([a, b])$ and also its completion $L^2([a, b])$. These trigonometric polynomials form a basis, generally referred to as the Fourier basis. It then follows that a function $f \in L^2([a, b])$ can then be approximated by the partial sums of its Fourier series representation. This is effectively a truncation of the Fourier series representation to a finite sum of N terms or equivalently the projection of

the function onto a subspace of $L^2([a,b])$ spanned by the first N trigonometric polynomials, $1, \cos x, \sin x, \ldots$.

The Fourier basis is not the only basis of $L^2([a,b])$. There are in fact many bases, of particular interest are the Legendre polynomials which are traditionally defined on the interval $[-1,1]$, but can be extended to other intervals by simple mappings. The theorem which lies at the heart of this is the Weierstrass approximation theorem. One begins with the monomials $1, x, x^2, \ldots$ and uses the Gram-Schmidt orthogonalisation procedure, the result of which are the normalised Legendre polynomials. Thus every function $f \in L^2([-1,1])$ can be written as an infinite series of Legendre polynomials, i.e.,

$$f(x) = \sum_{j=0}^{\infty} \hat{f}_j P_j(x) \tag{15}$$

and approximated by its truncation or equivalently its projection onto the subspace spanned by $P_0, P_1, \ldots, P_N$. This subspace is the vector space of polynomials of degree no greater than N denoted hereafter as $\mathbb{P}_N$.

More generally, a sequence of orthogonal polynomials $\{\phi_n\}_{n=0}^{\infty}$ defined to be orthogonal on the interval $[a,b]$ with respect to a positive weight function $w(x) \in L^1([a,b])$ are particularly useful. They span a set which is dense in and forms a basis of the weighted space $L^2_w([a,b]) \equiv L^2([a,b], w(x)dx)$ which is defined with the weighted inner product

$$(f,g)_w = \int_a^b f(x)g(x)w(x)dx, \quad \forall f,g \in L^2_w([a,b]) \tag{16}$$

with the associated norm $\|f\|_w^2 \equiv (f,f)_w$. A function $f \in L^2_w([a,b])$ can then once again be approximated by its truncated series

$$f(x) \simeq \sum_{j=0}^{N} \hat{f}_j \phi_j(x) \ . \tag{17}$$

The classical orthogonal polynomials summarised in Table 1 are the only ones we will mention in this report. In App. B we include a summary of their most relevant properties. The normalised orthogonal polynomials can be constructed using the Gram-Schmidt procedure with monomials as mentioned above. However, it is common when using the these polynomials to use other normalisation conventions. These conventions are also given in the appendix.

Orthogonal polynomials arise in many contexts in mathematics and physics. As a result they can be introduced and defined a number of different ways. In particular, we mention that a Gram-Schmidt orthogonalisation procedure is not the only way to define them. They can also be defined through a three-term recurrence relation, which all orthogonal polynomials possess or are a solution to a second order linear differential equation. As an example of the latter, Hermite polynomials multiplied by a Gaussian are solutions to the Schrödinger equation with a harmonic oscillator potential. The generalised Laguerre polynomials appear when solving the Schrödinger equation describing the hydrogen atom.

Mostly we will be concerned with the Chebyshev polynomials of the first kind, T_n, whose expansions have very nice convergence properties. They are a special case of the Jacobi

polynomials, like the Legendre polynomials, with the defining weight function $w(x) = (1 - x^2)^{-1/2}$. They can also be defined explicitly as

$$T_n(x) = \cos\left(n \arccos(x)\right) \ . \tag{18}$$

Through a change of variables, $\theta = \arccos x$, it is plainly seen that expansions in these polynomials are intimately related to Fourier cosine series and their nice convergence properties are largely a consequence of Fourier analysis.

We now introduce a very important type of operator called a projection operator, P_N, which can be defined for any orthogonal basis $\{\phi_j\}_{j=0}^{\infty}$ of a Hilbert space $\mathcal{H}$ as

$$P_N u(x) = \sum_{j=0}^{N} \hat{u}_j \phi_j(x) \tag{19}$$

which is said to project the function $u \in \mathcal{H}$ onto the finite space spanned by the basis $\phi_0, \phi_1, \ldots, \phi_N$. The unknown coefficients in Eq. 19 are given by

$$\hat{u}_j = \frac{(\phi_j, u)_w}{(\phi_j, \phi_j)_w} = \frac{(\phi_j, u)_w}{\|\phi_j\|_w^2} \ , \tag{20}$$

where the weighted inner product is defined as in Eq. 16. This integral cannot, in general, be calculated exactly.

We have now paved the way to be able to state the general procedure for the method of weighted residuals. The method is as follows:

1. Choose a finite set of trial functions $\phi_0, \phi_1, \ldots, \phi_N$ that form an $N+1$-dimensional basis in terms of which we can expand $\overline{u}_n(x)$, i.e.,

$$\overline{u}_n(x) = \sum_{j=0}^{N} \hat{u}_j^{(n)} \phi_j(x) \tag{21}$$

   and the $N+1$ expansion coefficients $\hat{u}_j^{(n)}$ are yet to be determined. This will be referred to as the expansion of the function in the spectral basis.

2. Choose a finite set of test functions $\chi_0, \chi_1, \ldots, \chi_N$ and use these functions to define what it means for the residual to be small by using the Hilbert space inner product. By which we mean we require

$$(\chi_j, R) = 0 \quad \forall j = 0, 1, \ldots, N \ . \tag{22}$$

Various numerical methods follow from the choices made in (1) and (2), for example :

- Finite difference methods arise from choosing the trial functions to be overlapping polynomials of low order.
- Spectral methods follow from choosing globally smooth functions as the trial functions. By global we mean they extend over the whole spatial domain of interest. Examples of common choices are given in Table 1 but others are also possible.

Spectral methods come in a variety of flavours which can be further classified in terms of the choice of test functions used and hence the different approaches taken to minimise the residual function. Two of the most familiar approaches are:

Table 1: Summary of the most common trial and test functions used to construct spectral methods.

Method Name	Trial/Test functions	Domain	Weight $w(x)$
Fourier	e^{ikx}, $\cos kx$, $\sin kx$	$[0, 2\pi]$	1
Legendre	$P_n(x)$	$[-1, 1]$	1
Chebyshev	$T_n(x)$	$[-1, 1]$	$(1-x^2)^{-1/2}$
Laguerre	$L_n(x)$	$[0, \infty)$	e^{-x}
Hermite	$H_n(x)$	$(-\infty, \infty)$	e^{-x^2}

- *Galerkin method:* The test and trial functions are chosen to be the same and each individually satisfies the boundary conditions. These are very important methods, but we will not discuss them further.
- *Pseudo-spectral or collocation method:* The test functions are delta functions on grid of points x_j called collocation points,

$$\chi_j(x) = \delta(x - x_j) \ . \tag{23}$$

  This choice enforces the residual to be exactly zero through the smallness condition

$$0 = (\chi_j, R) = (\delta(x - x_j), R) = R(x_j) \ . \tag{24}$$

  Thus increasing the number of points in the grid, the residual function will be smaller throughout the domain. These methods are simpler than the Galerkin methods and are better suited to non-linear problems.

The most common pseudo-spectral methods are the ones were the trial functions are either the complex exponentials and trigonometric functions or the classical orthogonal polynomials. The methods constructed from the former functions are most appropriately applied to periodic problems. Nonetheless they can also be applied to non-periodic problems, but quite often discontinuities can arise through the artificial periodisation of the function. These discontinuites diminish the fast convergence of these Fourier methods because of the appearance of Gibbs phenomenon [10]. The methods constructed from orthogonal polynomials, like Chebyshev and Legendre polynomials, are more appropriate to non-periodic problems and have been shown to work effectively in a wide range of applications in fluid dynamics and general relativity.

Quadrature methods are used to approximately determine the expansion coefficients. In the case of the Fourier method one uses the composite trapezoidal, rectangular or midpoint rules which converge rapidly for periodic functions [12]. On the other hand, in the case of the orthogonal polynomials the most accurate way to evaluate these integrals is to evaluate them using an $N+1$-point Gaussian quadrature rule which can exactly integrate polynomials up to degree $2N+1$ [12]. Adapting the method to include the boundary points one uses a Gauss-Lobatto quadrature rule which can exactly integrate a $2N-1$ degree polynomial.

For the remainder of this report we will consider only the two most successful pseudo-spectral methods based on the Fourier and Chebyshev polynomial bases. However, in later investigations as we consider more realistic tracking problems and hence different potentials we may need to extend to other, possibly more appropriate, bases depending on the geometry of the

tracking problem and the nature of potential. We will now introduce some mathematical background on orthogonal polynomials, interpolation and numerical quadrature. After which, we will introduce the Fourier and Chebyshev pseudo-spectral methods separately.

The material of the following sections is all very well known and can be found in a number of good textbooks on orthogonal polynomials, analysis, numerical methods and spectral methods. We will not delve too deeply into the subtle mathematical arguments concerning the convergence of spectral expansions and the general applicability of these methods. We will only prove theorems and derive formulas that have considerable practical relevance to the problem at hand. In lieu of this mathematical analysis we will instead illustrate important concepts with examples and refer the reader to the mathematical literature for the technical details.

4 A few properties of orthogonal polynomials

Polynomials are the most familiar of functions and are a joy to work with as they are so easy to use. Most notably, they're straight forward to differentiate and integrate, a fact that we will rely on heavily. Orthogonal polynomials and collocation interpolation using them entirely underlies the pseudo-spectral methods used in the latter sections of this report. This and the following few sections are very brief, but very starchy. They contain definitions, theorems and proofs to provide the reader with relevant background and to place them in the correct frame of mind for what follows. The presentation of the material closely follows Ref. [17] and we refer the reader there for further details.

Let $\{\phi_j(x)\}_{j=0}^{\infty}$ be a sequence of orthogonal polynomials with respect to the positive weight function $w(x)$ on $[a,b]$ where ϕ_j is of degree j. Then there are a number of properties the reader should be aware of, a few of the most important ones are summarised below.

Definition 4.1. (Algebraic polynomial) A polynomial of degree N in the monomial basis is denoted

$$p_N(x) = k_N x^N + k_{N-1}^{(N)} x^{N-1} + \ldots + k_0^{(N)}\,, \tag{25}$$

where the all the coefficients are real and the leading coefficient of $p_N(x)$, i.e., k_N is non-zero.

Theorem 4.1. *(Three term recurrence relation) Let $\{\phi_j(x)\}_{j=0}^{n}$ be a sequence of orthogonal polynomials with leading coefficient k_j non-zero. Then there exist a three-term recurrence relation of the form*

$$\phi_{j+1}(x) = (a_j x + b_j)\phi_j(x) - c_j\phi_{j-1}(x) \quad , \quad j \geq 0, \tag{26}$$

with $\phi_{-1} \equiv 0$, $\phi_0(x) \equiv k_0$ and

$$a_j = \frac{k_{j+1}}{k_j} \tag{27}$$

$$b_j = \frac{k_{j+1}}{k_j}\frac{(x\phi_j, \phi_j)_w}{\|\phi_j\|_w^2} \tag{28}$$

$$c_j = \frac{k_{j-1}k_{j+1}}{k_j^2}\frac{\|\phi_j\|_w^2}{\|\phi_{j-1}\|_w^2}\,. \tag{29}$$

Proof: For the proof of this theorem we refer the reader to Refs. [17] and [24]. ■

The recurrence relations of classical polynomials are usually given in this form in books such as Abramowitz and Stegun [18]. However, other normalisations are possible and can be useful in various situations, e.g., monic and orthonormal normalisations.

Theorem 4.2. *(Roots of orthogonal polynomials) If ϕ_n is an orthogonal polynomial defined as above, then ϕ_n has exactly n roots and they are real, simple and lie in the interval $[a, b]$.*

Proof: For the proof of this theorem we refer the reader to Refs. [17] and [24]. ■

Lemma 4.1. *A polynomial ϕ_{n+1} as defined above is orthogonal to any polynomial $q \in \mathbb{P}_n$.*

Proof: For the proof of this theorem we refer the reader to Refs. [17] and [24]. ■

Theorem 4.3. *(Christoffel-Darboux formula) The following formula for a set of orthogonal polynomials $\{\phi_j\}_{j=0}^n$ is known as the Christoffel-Darboux formula*

$$\frac{\phi_{n+1}(x)\phi_n(y) - \phi_n(x)\phi_{n+1}(y)}{x-y} = \frac{k_{n+1}}{k_n} \sum_{j=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_j\|_w^2} \phi_j(x)\phi_j(y) \ . \tag{30}$$

Proof: The Christoffel-Darboux formula is proved by using the three-term recurrence relation, Eq. 26, of a set of orthogonal polynomials [17]. It follows from the Eq. 26 that

$$\begin{aligned} \phi_{j+1}(x)\phi_j(y) - \phi_j(x)\phi_{j+1}(y) &= [(a_j x + b_j)\phi_j(x) - c_j\phi_{j-1}(x)]\,\phi_j(y) \\ &\qquad -\phi_j(x)\,[(a_j y + b_j)\phi_j(y) - c_j\phi_{j-1}(y)] \\ &= a_j\,[x\phi_j(x)\phi_j(y) - \phi_j(x)y\phi_j(y)] \\ &\qquad +b_j\,[\phi_j(x)\phi_j(y) - \phi_j(x)\phi_j(y)] \\ &\qquad -c_j\,[\phi_{j-1}(x)\phi_j(y) - \phi_j(x)\phi_{j-1}(y)] \\ &= a_j\phi_j(x)\phi_j(y)(x-y) - c_j\,[\phi_{j-1}(x)\phi_j(y) - \phi_j(x)\phi_{j-1}(y)] \ . \end{aligned} \tag{31}$$

Then using

$$a_j = \frac{k_{j+1}}{k_j} \quad \text{and} \quad c_j = \frac{k_{j-1}k_{j+1}}{k_j^2}\frac{\|\phi_j\|_w^2}{\|\phi_{j-1}\|_w^2} \tag{32}$$

from Eq. 27 and Eq. 29 we obtain

$$\begin{aligned} \phi_{j+1}(x)\phi_j(y) - \phi_j(x)\phi_{j+1}(y) &= \frac{k_{j+1}}{k_j}\phi_j(x)\phi_j(y)(x-y) \\ &\qquad -\frac{k_{j-1}k_{j+1}}{k_j^2}\frac{\|\phi_j\|_w^2}{\|\phi_{j-1}\|_w^2}\,[\phi_{j-1}(x)\phi_j(y) - \phi_j(x)\phi_{j-1}(y)] \ . \end{aligned} \tag{33}$$

Rearranging we get

$$\begin{aligned} &\frac{k_j}{k_{j+1}\|\phi_j\|_w^2}\frac{\phi_{j+1}(x)\phi_j(y) - \phi_j(x)\phi_{j+1}(y)}{x-y} \\ &\qquad +\frac{k_{j-1}}{k_j\|\phi_{j-1}\|_w^2}\frac{\phi_{j-1}(x)\phi_j(y) - \phi_j(x)\phi_{j-1}(y)}{x-y} = \frac{\phi_j(x)\phi_j(y)}{\|\phi_j\|_w^2} \ . \end{aligned} \tag{34}$$

Now sum over j

$$\sum_{j=0}^{n} \frac{k_j}{k_{j+1}\|\phi_j\|_w^2} \frac{\phi_{j+1}(x)\phi_j(y) - \phi_j(x)\phi_{j+1}(y)}{x-y} + \sum_{j=0}^{n} \frac{k_{j-1}}{k_j\|\phi_{j-1}\|_w^2} \frac{\phi_{j-1}(x)\phi_j(y) - \phi_j(x)\phi_{j-1}(y)}{x-y} = \sum_{j=0}^{n} \frac{\phi_j(x)\phi_j(y)}{\|\phi_j\|_w^2} \tag{35}$$

and now shuffle summation indices,

$$\sum_{j=0}^{n} \frac{k_j}{k_{j+1}\|\phi_j\|_w^2} \frac{\phi_{j+1}(x)\phi_j(y) - \phi_j(x)\phi_{j+1}(y)}{x-y} + \sum_{j=-1}^{n-1} \frac{k_j}{k_{j+1}\|\phi_j\|_w^2} \frac{\phi_j(x)\phi_{j+1}(y) - \phi_{j+1}(x)\phi_j(y)}{x-y} = \sum_{j=0}^{n} \frac{\phi_j(x)\phi_j(y)}{\|\phi_j\|_w^2} \ , \tag{36}$$

to finally get

$$\frac{k_n}{k_{n+1}\|\phi_n\|_w^2} \frac{\phi_{n+1}(x)\phi_n(y) - \phi_n(x)\phi_{n+1}(y)}{x-y} = \sum_{j=0}^{n} \frac{\phi_j(x)\phi_j(y)}{\|\phi_j\|_w^2} \ . \tag{37}$$

To get this result note that the sums on the left hand side are a telescoping series and $\phi_{-1} \equiv 0$, which means the first term in the second sum is zero and the rest of the terms in the second series cancel all the terms in the first, except for the last, leaving the final result. Rearranging terms then gives Eq. 30. ■

Proposition 4.1. *(Christoffel-Darboux formula ($y = x$)) The special case of $y = x$ for the Christoffel-Darboux formula is*

$$\phi'_{n+1}\phi_n(x) - \phi'_n(x)\phi_{n+1}(x) = \frac{k_{n+1}}{k_n} \sum_{j=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_j\|_w^2} \phi_j^2(x) \ . \tag{38}$$

Proof: Take the limit of the Christoffel-Darboux formula such that the right hand side evaluates to

$$\lim_{y\to x} \frac{\phi_{n+1}(x)\phi_n(y) - \phi_n(x)\phi_{n+1}(y)}{x-y} = \lim_{y\to x} \sum_{j=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_j\|_w^2} \phi_j(x)\phi_j(y) = \sum_{j=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_j\|_w^2} \phi_j^2(x) \ . \tag{39}$$

To evaluate the left hand side limit note that the left hand side can be re-written as

$$\lim_{y\to x} \frac{\phi_{n+1}(x) - \phi_{n+1}(y)}{x-y} . \phi_n(y) - \frac{\phi_n(x) - \phi_n(y)}{x-y} . \phi_{n+1}(y) \tag{40}$$

and note that the limits of the quotients are nothing other than the definition of the derivative of the polynomials. Thus, Eq. 38 is obtained. ■

5 Interpolation

Interpolation is the process of finding a function that fits a set of given data points and then evaluating it at intermediate points. When this function is constrained to pass through each point this approach is called collocation. Interpolation can be done in a number of different ways, but we will be mostly concerned with polynomial interpolation in this section. However, it can also be done, for example, using Fourier series and generalisations thereof.

A whole slew of numerical methods depend on polynomial interpolation, in particular many numerical integration and differentiation methods make use of it. They use it because polynomials can be evaluated, differentiated and integrated with great ease. In fact, interpolation will under lie everything we do in this report. Polynomial interpolation has had a somewhat bad reputation in the past, but if done correctly it is extremely good. In particular, see Ref. [25] and the appendix *"Six myths of polynomial interpolation and quadrature"* of Ref. [11].

A polynomial of order N that passes through $N+1$ points is unique, but there are many ways this interpolating polynomial can be expressed, i.e., through

- a solution of a linear system of equations
- Lagrange's interpolation formula
- Newton's divided differences
- Hermite's interpolation formula
- barycentric interpolation formulas .

We will use the latter, but we will also discuss the first two of these. The first is the simplest and most obvious approach which turns out to be numerically unstable, whereas the second has great analytic applicability but is more computationally demanding than one may like. The generalisation of Lagrange's interpolation formula to the barycentric formula addresses both concerns of stability and efficiency.

To illustrate these different approaches to constructing the Nth degree interpolating polynomial and to highlight their fundamental flaws and strengths we will define a few terms. Let $\{x_j\}_{j=0}^{N}$ be $N+1$ distinct nodes with corresponding data values f_j, which need not be distinct and, for our purposes, may be considered as evaluations of a function f, i.e., $f_j = f(x_j)$. This set of ordered pairs, (x_j, f_j), $j = 0, 1, \ldots N$ will be referred to as the interpolation points. For simplicity, we will assume that the x_j are real and we will make no assumption about the spacing of this grid of nodes. The problem at hand is to find the unique polynomial $p_N \in \mathbb{P}_N$ that interpolates the function f at the interpolation points, meaning

$$p_N(x_j) = f(x_j) = f_j \,. \tag{41}$$

5.1 The simplest approach

The simplest approach is to formulate the interpolation problem as a linear system of equations. This leads to a situation where the matrix to be inverted is a Vandermonde matrix, i.e., a matrix where in each row there is a geometric progression of terms. This approach is

very simple, but susceptible to round off errors for large N because of the ill conditioned Vandermonde matrix.

An Nth order polynomial can be written down in the monomial basis as

$$p_N(x) = \sum_{j=0}^{N} a_j x^j \,. \tag{42}$$

As $p_N(x)$ is required to pass through the $N+1$ points $(x_j, f(x_j))$, we obviously have the following $N+1$ algebraic equations

$$\begin{aligned} p_N(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_N x_0^N & (43)\\ p_N(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_N x_1^N & (44)\\ \vdots & \qquad\qquad \vdots & (45)\\ p_N(x_N) &= a_0 + a_1 x_N + a_2 x_N^2 + \ldots + a_N x_N^N & (46) \end{aligned}$$

which when expressed in matrix form are written as

$$\mathbf{f} = V\mathbf{a} \,. \tag{47}$$

The terms in Eq. (47) are the following: $\mathbf{f}$ is the vector of function values at the nodes x_j,

$$\mathbf{f} = [f_0 \;\; f_1 \;\; \ldots \;\; f_N]^\top \,, \tag{48}$$

V is the Vandermonde matrix

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & \ldots & x_0^N \\ \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & & \ddots & & \vdots \\ \vdots & \vdots & & & \ddots & \vdots \\ 1 & x_N & x_N^2 & \ldots & \ldots & x_N^N \end{bmatrix} \tag{49}$$

and $\mathbf{a}$ is the vector of unknown coefficients

$$\mathbf{a} = [a_0 \;\; a_1 \;\; \ldots \;\; a_N]^\top \,. \tag{50}$$

The unknown coefficients can then be found using basic linear algebra. However, the matrix to be inverted is ill conditioned because of the increasing number of powers of x with increasing N.

5.2 Lagrange interpolation

Lagrange's Nth degree interpolating polynomial is defined by the following formula

$$p_N(x) = \sum_{j=0}^{N} f_j \ell_j(x) \,, \quad \ell_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{(x - x_k)}{(x_j - x_k)} \,. \tag{51}$$

This formula is known as Lagrange's interpolation formula and $\ell_j(x)$ is the jth Lagrange basis polynomial. It is the Nth degree interpolating polynomial that satisfies

$$\ell_j(x_k) = \delta_{jk} = \begin{cases} 1, & j = k \\ 0, & \text{otherwise} \end{cases}, \quad j,k = 0,1,\ldots N\,. \tag{52}$$

Obviously, the denominator in $\ell_j(x)$ is a constant and the numerator is a degree N polynomial with zeros at the nodes x_k, with $k \neq j$. Moreover, by direct substitution of Eq. 52 into Eq. 51 we can show that $p_N(x)$ defined as in Eq. 51 does indeed interpolate the data f_j on the grid x_j,

$$p_N(x_k) = \sum_{j=0}^{N} f(x_j)\ell_j(x_k) = \sum_{j=0}^{N} f(x_j)\delta_{jk} = f_k\,, \tag{53}$$

for $j,k = 0,1,\ldots,N$.

A clear strength of writing the interpolating polynomial as in Eq. 51 is that one does not need to solve a linear system of equations to evaluate the interpolating polynomial. Moreover, it has a rather simple structure that can be easily programmed. It also proves to have great analytical applicability being used in various proofs and derivations such as in Gaussian quadrature. However, it is not very efficient computationally. It takes $\mathcal{O}(N^2)$ operations to evaluate $p_N(x)$ for a single point. Also, if we are given another interpolation point we will have to calculate the interpolating polynomial again from scratch.

5.3 Barycentric Lagrange interpolation

The Lagrange interpolation formula can be massaged into a form which is more efficient and can be updated more easily, although it is rarely mentioned in numerical analysis textbooks. This brief presentation closely follows Ref. [25] and [11] and we refer the reader to these for a more complete description. To familiarise ourselves with this more profitable approach we begin by defining the node polynomial

$$\ell(x) = \prod_{j=0}^{N}(x - x_j) \in \mathbb{P}_{N+1} \tag{54}$$

for the grid $\{x_j\}_{j=0}^{N}$. The node polynomial can be used to write the numerator of $\ell_j(x)$ as

$$\prod_{\substack{k=0 \\ k\neq j}}^{N}(x - x_k) = \frac{\ell(x)}{x - x_j}\,. \tag{55}$$

Thus, if we define the so-called barycentric weights λ_j as the denominator of $\ell_j(x)$, i.e.,

$$\lambda_j = \prod_{\substack{k=0 \\ k\neq j}}^{N}(x_j - x_k)^{-1}\,, \tag{56}$$

we can write the jth Lagrange polynomial $\ell_j(x)$ as

$$\ell_j(x) = \ell(x)\frac{\lambda_j}{x - x_j}\,. \tag{57}$$

Substituting this into Eq. 51 we can pull the node polynomial outside of summation yielding

$$p_N(x) = \ell(x) \sum_{j=0}^{N} \frac{\lambda_j}{x - x_j} f_j \ . \tag{58}$$

Equation 58 is known as the first barycentric formula. This formula can be further manipulated into a more appealing form, but already in Eq. 58 we can see an increase in efficiency. The barycentric weights are independent of x and only need to be computed once. Moreover, for some important special cases such as Chebyshev grids these weights are known analytically.

To further improve the efficiency we can interpolate the function which maps every point to 1, i.e.,

$$1 = \sum_{j=0}^{N} \ell_j(x) = \sum_{j=0}^{N} \frac{\ell(x)\lambda_j}{x - x_j} = \ell(x) \sum_{j=0}^{N} \frac{\lambda_j}{x - x_j} \tag{59}$$

and use this to divide the jth Lagrange polynomial, then expand and cancel the node polynomial $\ell(x)$,

$$\ell_j(x) = \frac{\ell_j(x)}{1} = \frac{\ell_j(x)}{\sum_{j=0}^{N} \ell_j(x)} = \frac{\frac{\ell(x)\lambda_j}{x - x_j}}{\sum_{j=0}^{N} \frac{\ell(x)\lambda_j}{x - x_j}} = \frac{\frac{\ell(x)\lambda_j}{x - x_j}}{\ell(x) \sum_{j=0}^{N} \frac{\lambda_j}{x - x_j}} = \frac{\frac{\lambda_j}{x - x_j}}{\sum_{j=0}^{N} \frac{\lambda_j}{x - x_j}} \ . \tag{60}$$

Substituting Eq. 60 into the formula for the Lagrange interpolating polynomial, Eq. 51, we obtain the second form of the barycentric interpolation formula

$$p_N(x) = \frac{\sum_{j=0}^{N} \frac{\lambda_j}{x - x_j} f_j}{\sum_{j=0}^{N} \frac{\lambda_j}{x - x_j}} \ . \tag{61}$$

We must supplement this formula with the special case $p_N(x_j) = f_j$ to avoid the division of infinity by infinity. It is not self-evident that Eq. 61 defines an Nth degree polynomial, let alone a polynomial. It is, however, a polynomial for specific choices of barycentric weights. With regards to its efficiency to evaluate $p_N(x)$, it only takes $\mathcal{O}(N)$ operations for each x once the barycentric weights are known.

For the special case of a Chebyshev extrema grid, the barycentric weights are

$$\lambda_j = \begin{cases} \frac{2^{N-2}}{N}(-1)^j & j = 0, N \\ \frac{2^{N-1}}{N}(-1)^j & \text{otherwise} \end{cases} \ . \tag{62}$$

As the second form of the barycentric formula as given in Eq. 61 is scale invariant, the constants cancel and one accordingly obtains [25]

$$p_N(x) = \frac{\sum_{j=0}^{N}{}'' \frac{(-1)^j f_j}{x - x_j}}{\sum_{j=0}^{N}{}'' \frac{(-1)^j}{x - x_j}} \ , \tag{63}$$

where we have used the convention of placing a double prime on the summation symbol to indicate that the first and last terms are halved.

The trigonometric version is similar and their corresponding formulas for the barycentric interpolant can found for example in Ref. [26]. In our later calculations we use Chebfun's `trigBary()` and `bary()` functions [27].

5.4 Runge phenomenon

Polynomial interpolation underlies our approach to solving the Schrödinger equation. Therefore this report would not be complete without mentioning Runge's phenomenon [28]. It is the problem of large oscillations near the boundaries which can occur when performing high degree polynomial interpolation on a set of equally spaced points. We illustrate this phenomenon in Fig. 1 for Runge's function

$$r(x) = \frac{1}{1+(5x)^2} \ . \tag{64}$$

In Fig. 1 we see the polynomial interpolation of Runge's function in evenly spaced points. As the degree of the polynomial interpolant N is increased, larger and larger oscillations occur near the boundaries.

From Weierstrass' approximation theorem, see for example Ref. [29], it is known that if a function f is continuous on $[a,b]$, then there exists a sequence of polynomials p_n such that

$$\lim_{n\to\infty} p_n(x) = f(x) \tag{65}$$

uniformly on $[a,b]$. However, it can be shown that when interpolating Runge's function in equally spaced points the interpolation error can increase without bound

$$\lim_{n\to\infty} \|f-p_n\|_\infty = \lim_{n\to\infty} \max_{a\le x\le b} |f(x)-p_n(x)| = \infty \ . \tag{66}$$

One way to combat this problem is to change the interpolation points. Choosing points clustered more at the boundaries and generally distributed according to the following density per unit length [10]

$$\text{density} \sim \frac{N}{\pi\sqrt{1-x^2}} \quad , \quad \text{as} \quad N\to\infty \ . \tag{67}$$

Several sets of points can be used. In Fig. 2 we illustrate this with the Chebyshev points of the second kind and the zeros of Legendre polynomials. Both work well. The point we wish the reader to take away from this is the following. Interpolation can be very useful, but if you choose the wrong points to perform the interpolation in you will be sorry.

6 Numerical quadrature

The material in this section is discussed in many textbooks, in particular, we refer the reader to Refs. [10–12, 17, 24, 30] and the references cited herein.

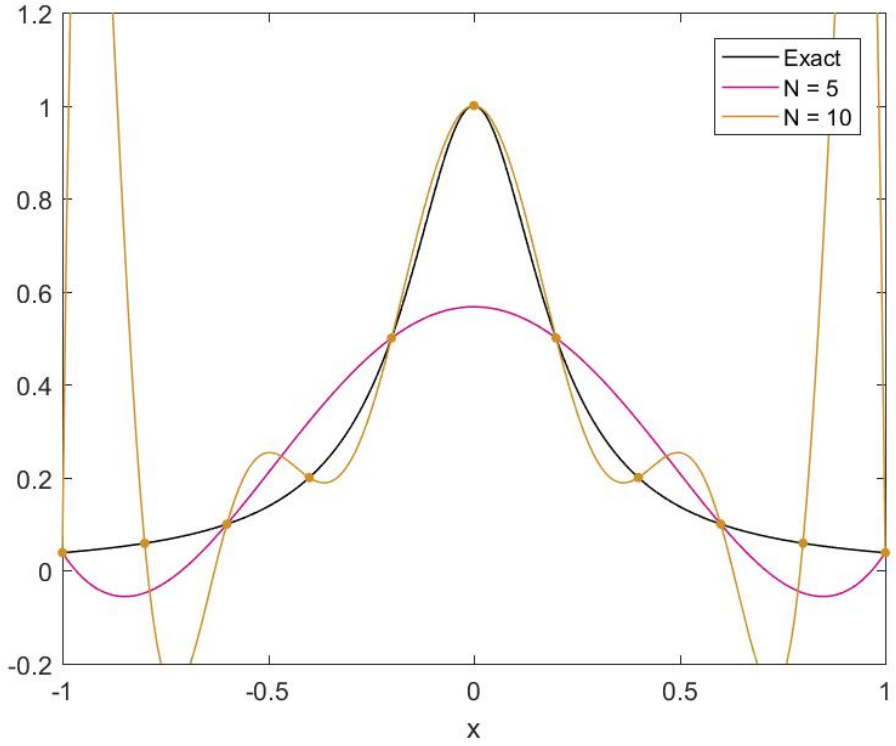
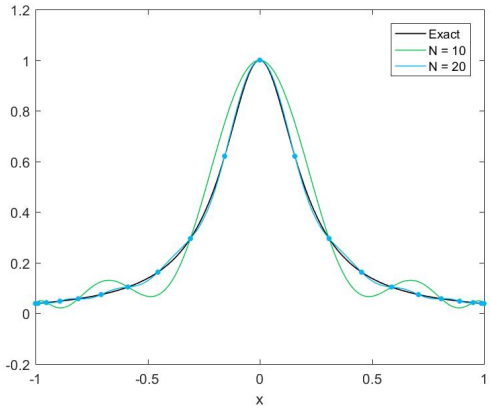
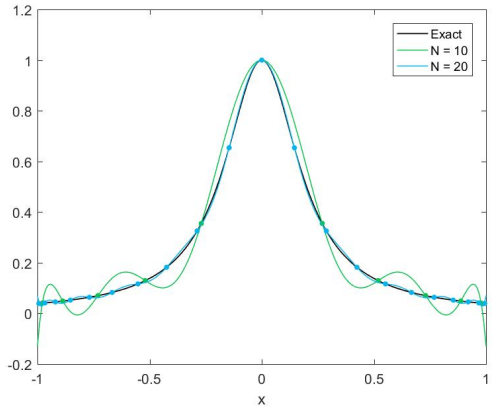


Figure 1: Runge's function interpolated on an equispaced grid exhibiting Runge phenomenon. This plot is produced by using the MATLAB® script `Ex_Script_1_RungePhenomenon.m`.



(i) Runge's function interpolated on $N+1$ Chebyshev points of the second kind.



(ii) Runge's function interpolated on the $N+1$ zeros of the Legendre polynomial L_{N+1}.

Figure 2: Runge's function interpolated on $N+1$ unevenly spaced points. These plots are produced by using the MATLAB® script `Ex_Script_1_RungePhenomenon.m`.

6.1 Gaussian quadrature

Quadrature rules are used to approximate definite integrals by a finite sum of the following form

$$\int_a^b w(x)f(x)dx = \sum_{j=0}^{n} w_j f(x_j) + R_n(f) = I_n(f) + R_n(f)\,, \tag{68}$$

where $w(x)$ is a positive weight function over the domain $[a,b]$, the $n+1$ numbers x_j are called the quadrature nodes, the $n+1$ numbers w_j the weights and $R_n(f)$ the error or residual of the approximation $I_n(f)$ of the exact integral. The quadrature rule given by $I_n(f)$ is said to be exact if the residual $R_n(f)$ is zero.

Gaussian quadrature, which was named after the mathematician Carl Friedrich Gauss, is designed to be exact for all polynomials of degree $2n+1$ or less. This is a significant improvement over other quadrature rules, such as Newton-Cotes, for the $n+1$ function evaluations performed. There are $2n+2$ parameters, i.e., the nodes x_j and weights w_j, in the quadrature rule given by Eq. 68 which we can adjust to exactly integrate a polynomial of degree $2n+1$ which has $2n+2$ coefficients. The nodes are, in general, not equally spaced but rather given by roots of orthogonal polynomials.

Given a set of fixed nodes x_j, the degree n interpolating polynomial $p_n(x) \in \mathbb{P}_n$ of the function $f(x)$ on this grid is given by the Lagrange interpolation formula, Eq. 51. Making this approximation we find

$$\int_a^b w(x)f(x)dx \simeq \int_a^b w(x)p_n(x)dx = \sum_{j=0}^{n} f(x_j)\int_a^b w(x)\ell_j(x)dx\ . \tag{69}$$

Thus the quadrature rule

$$I_n(f) = \sum_{j=0}^{n} w_j f(x_j) \tag{70}$$

with the weights defined as

$$w_j = \int_a^b w(x)\ell_j(x)dx \tag{71}$$

will be exact for polynomials of degree n or less regardless of the nodes we choose to use. These nodes are the roots of an arbitrary polynomial.

We would like to be able to integrate polynomials of largest degree possible. The quadrature nodes are $n+1$ degrees of freedom that we can adjust allowing us to integrate a polynomial $p \in \mathbb{P}_{2n+1}$ exactly. To this end, we introduce a set of orthogonal polynomials $\{\phi_j(x)\}_{j=0}^{n+1}$ which are defined to be orthogonal with respect to the inner product

$$(f,g)_w = \int_a^b f(x)g(x)w(x)dx \tag{72}$$

for a positive weight function $w(x) \in L^1(a,b)$ which is only zero on a set of measure zero, e.g., a finite set.

If we employ polynomial division, the $2n+1$ degree polynomial $p(x)$ can be written as

$$p(x) = \phi_{n+1}(x)q(x) + r(x)\,, \tag{73}$$

where the quotient and remainder polynomials $q(x), r(x) \in \mathbb{P}_n$ and depend in particular on $p(x)$. Using this fact we can rewrite the weighted integral of $p(x)$ as

$$\int_a^b p(x)w(x)dx = \int_a^b \phi_{n+1}(x)q(x)w(x)dx + \int_a^b r(x)w(x)dx \tag{74}$$

$$= (\phi_{n+1}, q)_w + \int_a^b r(x)w(x)dx \tag{75}$$

$$= \int_a^b r(x)w(x)dx\,, \tag{76}$$

where the third equality follows from the fact that the polynomial $\phi_{n+1}(x)$ is orthogonal to all $q \in \mathbb{P}_n$. Equation 76 is exact, we would like to apply the quadrature rule to the left hand side of this equation and get the exact value by choosing the nodes appropriately. Application of the $n+1$ point quadrature rule to the weighted integral of $p(x)$ gives

$$I_n(p) = \sum_{j=0}^{n} w_j p(x_j) \tag{77}$$

$$= \sum_{j=0}^{n} w_j \phi_{n+1}(x_j)q(x_j) + \sum_{j=0}^{n} w_j r(x_j) \tag{78}$$

$$= \sum_{j=0}^{n} w_j \phi_{n+1}(x_j)q(x_j) + \int_a^b r(x)w(x)dx\,. \tag{79}$$

The third equality follows from the fact $r(x) \in \mathbb{P}_n$ and the quadrature rule will exactly integrate a polynomial in $\mathbb{P}_n$. Thus, since $q(x)$ will vary depending on $p(x)$, for the quadrature rule to be exact the nodes should be chosen to be the $n+1$ roots of $\phi_{n+1}(x)$, so that

$$\sum_{j=0}^{n} w_j \phi_{n+1}(x_j)q(x_j) = 0\,. \tag{80}$$

However, since we are integrating over $[a, b]$ these roots need to be real, distinct and lie in $[a, b]$. The crucial specification that the set of polynomials $\{\phi_j\}_{j=0}^{n+1}$ be orthogonal has the important consequence that $\phi_j(x)$ has exactly j real and distinct roots in the interval $[a, b]$.

Theorem 6.1. *(Gaussian Quadrature) If we define a Gaussian quadrature rule for the definite integral*

$$\int_a^b w(x)f(x)dx \tag{81}$$

by

$$I_n(f) = \sum_{j=0}^{n} w_j f(x_j) \tag{82}$$

where the nodes x_j are the $n+1$ roots of a degree $n+1$ orthogonal polynomial, which is orthogonal on $[a, b]$ with respect to the positive weight function $w(x) \in L^1_{(a,b)}$ that is only zero on a set of measure zero (e.g., a finite set), and the weights are defined as

$$w_j = \int_a^b w(x)\ell_j(x). \tag{83}$$

Then $I_n(f)$ is exact for all polynomials f of degree $2n+1$ and less.

Table 2: Summary of the most used orthogonal polynomials for Gaussian quadrature.

Polynomial Name	Symbol	Domain	Weight $w(x)$
Legendre	$P_n(x)$	$[-1,1]$	1
Chebyshev (1st kind)	$T_n(x)$	$[-1,1]$	$(1-x^2)^{-1/2}$
Laguerre	$L_n(x)$	$[0,\infty)$	e^{-x}
Hermite	$H_n(x)$	$(-\infty,\infty)$	e^{-x^2}

Proof: The proof is provided by the preceding text in this section, see also [17, 24]. ■

Theorem 6.2. *Let $f \in C^{2n+2}_{[a,b]}$ and $I_n(f)$ be the Gaussian quadrature rule given in Theorem 6.1. Then the residual is*

$$R_n(f) \equiv \int_a^b w(x)f(x)dx - I_n(f) = \frac{f^{(2n+2)}(s)}{(2n+2)!}\int_a^b \ell^2(x)w(x)dx \tag{84}$$

for some $s \in [a,b]$ and $\ell(x)$ is the node polynomial constructed from the points of Gaussian quadrature, i.e., the roots of the $n+1$ degree orthogonal polynomial.

Proof: For the proof of this theorem we refer the reader to Ref. [24]. ■

Gaussian quadrature is based on the exact weighted integration of polynomials, so if the function in the integrand is well approximated by the orthogonal polynomials used to construct the quadrature method, it should make a reliable approximation to the exact integral. We have presented Gaussian quadrature in terms of an arbitrary orthogonal polynomial, but quite often when Gaussian quadrature is discussed the use of Legendre polynomials is implied. As we are striving to be slightly more general we will refer to this specific form of Gaussian quadrature as Gauss-Legendre quadrature. It is a particularly useful method and the most common because Legendre polynomials are orthogonal with respect to the simplest possible weight function, namely $w(x) = 1$. Legendre polynomials are defined on interval $[-1,1]$ but the Gauss-Legendre method can easily be extended to work on any finite interval $[a,b]$ or even a semi-infinite or infinite interval through the use of a domain map.

A drawback of using this particular set of orthogonal polynomials is that their roots are only known analytically for the first few polynomials and then the roots for the higher degree Legendre polynomials must be found numerically. The need to calculate the roots and weights numerically is also present for other classical orthogonal polynomials that are commonly used, like the Laguerre and Hermite polynomials. However, in the special case of Chebyshev polynomials formulas for both the roots and weights are known analytically. By choosing the orthogonal polynomial used to construct a Gaussian quadrature method carefully, one can hide some of the nastiness (or non-polynomial behaviour) in the weight function which defines the polynomial. For a summary of the domains and weight functions for the most common polynomials used to construct Gaussian quadrature methods see Table 2[2]. Appendix B contains a summary of the properties of the four previously mentioned classical orthogonal polynomials.

Proposition 6.1. *(Positive Weights) The weights of Gaussian quadrature w_j, $j = 0,1,\ldots,n$ are all positive.*

[2]The Legendre and Chebyshev polynomials are special cases of the more general Jacobi polynomials, $P_n^{\alpha,\beta}(x)$ which are defined by the weight function $w(x) = (1-x)^\alpha(1+x)^\beta$ where $\alpha,\beta > -1$ [18].

Proof: Consider the polynomial $p(x) = \ell_j^2(x) \in \mathbb{P}_{2n}$, where $\ell_j(x)$ is the jth Lagrange polynomial as given in Eq. 51. Since the weight function $w(x)$ is positive on $[a,b]$ and the $n+1$ point Gaussian quadrature method is exact for all polynomials in $\mathbb{P}_{2n+1}$ then

$$0 < \int_a^b \ell_j^2(x) w(x) dx = \sum_{k=0}^{n} \ell_j^2(x_k) w_k = \sum_{k=0}^{n} (\delta_{jk})^2 w_k = w_k \ . \tag{85}$$

■

Proposition 6.2. *(Gaussian Weights) The weights of an $n+1$ point Gaussian quadrature method, as defined in Theorem 6.1, i.e., constructed from the set of orthogonal polynomials $\{\phi_j\}_{j=0}^n$, can be expressed as*

$$w_j = \frac{k_{n+1}}{k_n} \frac{\|\phi_n\|_w^2}{\phi_n(x_j)\phi'_{n+1}(x_j)} \ , \tag{86}$$

where k_j is the leading coefficient in the orthogonal polynomial $\phi_j(x)$.

Proof: The x_j are the roots of the orthogonal polynomial ϕ_{n+1}, thus the Christoffel-Darboux formula with $y = x_j$ simplifies to

$$\frac{\phi_{n+1}(x)\phi_n(x_j)}{x - x_j} = \frac{k_{n+1}}{k_n} \sum_{i=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_i\|_w^2} \phi_i(x)\phi_i(x_j) \ . \tag{87}$$

Now perform the weighted integral over $[a,b]$

$$\begin{aligned}
\phi_n(x_j) \int_a^b \frac{\phi_{n+1}(x)}{x - x_j} w(x) dx &= \frac{k_{n+1}}{k_n} \int_a^b dx\, w(x) \sum_{i=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_i\|_w^2} \phi_i(x)\phi_i(x_j) \\
&= \frac{k_{n+1}}{k_n} \|\phi_n\|_w^2 \sum_{i=0}^{n} \frac{1}{\|\phi_i\|_w^2} \int_a^b dx\, w(x)\phi_i(x)\phi_i(x_j) \\
&= \frac{k_{n+1}}{k_n} \|\phi_n\|_w^2 \sum_{i=0}^{n} \frac{1}{\|\phi_i\|_w^2} (\phi_i(x), \phi_i(x_j)) \\
&= \frac{k_{n+1}}{k_n} \|\phi_n\|_w^2 \frac{(\phi_0(x), \phi_0(x_j))}{\|\phi_0\|_w^2} \\
&= \frac{k_{n+1}}{k_n} \|\phi_n\|_w^2 \ .
\end{aligned} \tag{88}$$

Since x_j are the zeros of ϕ_{n+1}, the node polynomial $\ell(x)$ constructed from this grid is simply related to ϕ_{n+1} by

$$\ell(x) = \frac{\phi_{n+1}(x)}{k_{n+1}} \ , \tag{89}$$

where k_{n+1} is the leading coefficient of ϕ_{n+1}. Thus, we have

$$\ell_j(x) = \frac{\ell(x)}{\ell'(x_j)(x - x_j)} = \frac{\phi_{n+1}(x)}{\phi'_{n+1}(x_j)(x - x_j)} \ . \tag{90}$$

Rearranging to obtain

$$\phi_{n+1}(x) = \phi'_{n+1}(x_j)(x - x_j)\ell_j(x) \tag{91}$$

and substituting this result into the left hand side of Eq. 88 we find

$$\begin{aligned}\phi_n(x_j)\int_a^b \frac{\phi_{n+1}(x)}{x-x_j}w(x)dx &= \phi_n(x_j)\int_a^b \frac{\phi'_{n+1}(x_j)(x-x_j)\ell_j(x)}{x-x_j}w(x)dx \\ &= \phi_n(x_j)\phi'_{n+1}(x_j)\int_a^b w(x)\ell_j(x)dx \\ &= \phi_n(x_j)\phi'_{n+1}(x_j)w_j \ . \end{aligned} \tag{92}$$

Putting Eq. 88 and 92 together we have

$$\phi_n(x_j)\phi'_{n+1}(x_j)w_j = \frac{k_{n+1}}{k_n}\|\phi_n\|_w^2 \tag{93}$$

and therefore upon rearrangement we obtain

$$w_j = \frac{k_{n+1}}{k_n}\frac{\|\phi_n\|_w^2}{\phi_n(x_j)\phi'_{n+1}(x_j)} \ . \tag{94}$$

There exists another way to prove this result. The prove relies on the orthogonality with lower degree polynomials and l'Hospital's rule, see Wikipedia page on Gaussian quadrature. ■

6.1.1 Golub-Welsch algorithm

In the previous section we gave two formulas for the weights of Gaussian quadrature. However, there is a better way to calculate these weights which also provides the nodes. It relies on using the three-term recurrence relation that all orthogonal polynomials possess and reformulating the problem of finding the weights and nodes as an eigenvalue problem. This method can be used if the coefficients in the three-term recurrence relation are know. For the classical orthogonal polynomials much is known including this relation. The method is known as the Golub-Welsch algorithm and it is suitable when n is not too big. This is for reasons of computational efficiency and accuracy. When n is large other approaches should be considered instead, see later discussions in the examples of this section. We will not discuss these other approaches. In this report, we will not use this algorithm in later sections. However, it is included here because its use, or an extension of it, will allow the use of other orthogonal polynomials such as Legendre, Laguerre or Hermite which do not have simple closed form expressions for their Gaussian quadrature nodes and weights.

Given the first two polynomials and a three-term recurrence relation , i.e.,

$$\phi_{-1}(x) \equiv 0 \tag{95}$$
$$\phi_0(x) \equiv 1 \tag{96}$$
$$\phi_{j+1}(x) = (a_j x + b_j)\phi_j(x) - c_j\phi_{j-1}(x)\,, \quad j = 0, 1, \ldots \tag{97}$$

with $a_j > 0$, $c_j > 0$ a set of orthogonal polynomials is generated. If we re-write the three-term recurrence relation with $x\phi_{j-1}(x)$ on the left hand side we have

$$x\phi_j(x) = \frac{c_j}{a_j}\phi_{j-1}(x) - \frac{b_j}{a_j}\phi_j(x) + \frac{1}{a_j}\phi_{j+1}(x) \ . \tag{98}$$

We immediately see that we have a system of $n+1$ equations

$$\begin{aligned}
x\phi_0(x) &= \frac{c_1}{a_1}\phi_{-1}(x) - \frac{b_1}{a_1}\phi_0(x) + \frac{1}{a_1}\phi_1(x) = -\frac{b_1}{a_1}\phi_0(x) + \frac{1}{a_1}\phi_1(x) \\
x\phi_1(x) &= \frac{c_2}{a_2}\phi_0(x) - \frac{b_2}{a_2}\phi_1(x) + \frac{1}{a_2}\phi_2(x) \\
x\phi_2(x) &= \frac{c_3}{a_3}\phi_1(x) - \frac{b_3}{a_3}\phi_2(x) + \frac{1}{a_3}\phi_3(x) \\
&\vdots \\
&\vdots \\
x\phi_{n-1}(x) &= \frac{c_n}{a_n}\phi_{n-2}(x) - \frac{b_n}{a_n}\phi_{n-1}(x) + \frac{1}{a_n}\phi_n(x) \\
x\phi_n(x) &= \frac{c_{n+1}}{a_{n+1}}\phi_{n-1}(x) - \frac{b_{n+1}}{a_{n+1}}\phi_n(x) + \frac{1}{a_{n+1}}\phi_{n+1}(x)
\end{aligned} \tag{99}$$

for $j = 0, \ldots, n$. This system of equations can be written in terms of matrices and vectors as

$$x\begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \vdots \\ \vdots \\ \phi_n(x) \end{bmatrix} = \begin{bmatrix} -b_1/a_1 & 1/a_1 & & & & \\ c_2/a_2 & -b_2/a_2 & 1/a_2 & & & \bigcirc \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ \bigcirc & & & & & 1/a_n \\ & & & & c_{n+1}/a_{n+1} & -b_{n+1}/a_{n+1} \end{bmatrix} \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \vdots \\ \vdots \\ \phi_n(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ \phi_{n+1}(x)/a_{n+1} \end{bmatrix} \tag{100}$$

or in more succinct matrix form as

$$x\boldsymbol{\phi}(x) = T\boldsymbol{\phi}(x) + (1/a_{n+1})\phi_{n+1}(x)\mathbf{e}_{n+1} \ . \tag{101}$$

In Eq. 101 T is the tridiagonal matrix in Eq. 100, $\boldsymbol{\phi}$ is the vector $(\phi_0(x), \phi_1(x), \ldots \phi_n(x))^\top$ and $\mathbf{e}_{n+1}$ is the unit vector $(0, 0, \ldots, 1)^\top$. From Eq. 101 we see that $\phi_{n+1}(x) = 0$ if and only if

$$T\boldsymbol{\phi}(x) = x\boldsymbol{\phi}(x) \ , \tag{102}$$

where x is the eigenvalue of the tridiagonal matrix T.

Roots of orthogonal polynomials are very important in pseudo-spectral methods as they are used to define collocation grids. If we are only interested in the roots of an orthogonal polynomial then we can stop here. However, we are currently interested in Gaussian quadrature and therefore we need to also determine the weights. To find these weights it is convenient reformulate the three-term recurrence relation in terms of orthonormal polynomials. This can be accomplished either by dividing Eq. 97 by the norm of ϕ_j or equivalently by performing a diagonal similarity transformation. The resulting similar matrix J is symmetric whereas T, in general, is not. The symmetrised matrix J (commonly called a Jacobi matrix) corresponding

to T is given by

$$J = DTD^{-1} = \begin{bmatrix} \alpha_0 & \beta_1 & & & & \\ \beta_1 & \alpha_1 & \beta_2 & & & \bigcirc \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ \bigcirc & & & & \beta_{n-1} & \\ & & & \beta_{n-1} & a_n & \end{bmatrix} \tag{103}$$

where

$$\alpha_j = -\frac{b_j}{a_j}\,, \quad \beta_j = \left(\frac{c_j}{a_{j-1}a_j}\right)^{1/2} \,. \tag{104}$$

Moreover, each of the weights can be obtained from the first component of the eigenvectors of the Jacobi matrix.

We will now confirm this well known result by first finding the corresponding three-term recurrence relation for the normalised polynomials and hence the analogous eigenvalue problem. We divide the original three-term recurrence relation as given in Eq. 98 by $\|\phi_j\|_w$,

$$\begin{aligned} x\psi_j(x) &\equiv \frac{x\phi_j(x)}{\|\phi_j\|_w} \\ &= \frac{1}{\|\phi_j\|_w}\left(\frac{c_j}{a_j}\phi_{j-1}(x) - \frac{b_j}{a_j}\phi_j(x) + \frac{1}{a_j}\phi_{j+1}(x)\right) \\ &= \frac{1}{\|\phi_j\|_w}\left(\frac{c_j}{a_j}\|\phi_{j-1}\|_w\frac{\phi_{j-1}(x)}{\|\phi_{j-1}\|_w} - \frac{b_j}{a_j}\|\phi_j\|_w\frac{\phi_j(x)}{\|\phi_j\|_w} + \frac{1}{a_j}\|\phi_{j+1}\|_w\frac{\phi_{j+1}(x)}{\|\phi_{j+1}\|_w}\right) \\ &= \frac{c_j}{a_j}\frac{\|\phi_{j-1}\|_w}{\|\phi_j\|_w}\psi_{j-1}(x) - \frac{b_j}{a_j}\psi_j(x) + \frac{1}{a_j}\frac{\|\phi_{j+1}\|_w}{\|\phi_j\|_w}\psi_{j+1}(x)\,. \end{aligned} \tag{105}$$

Now we use Eq. 27 and 29 and simplify

$$\begin{aligned} x\psi_j(x) &= \frac{k_{j-1}k_{j+1}\|\phi_j\|_w^2 k_j}{k_j^2\|\phi_{j-1}\|_w^2 k_{j+1}}\frac{\|\phi_{j-1}\|_w}{\|\phi_j\|_w}\psi_{j-1}(x) - \frac{b_j}{a_j}\psi_j(x) + \frac{k_j}{k_{j+1}}\frac{\|\phi_{j+1}\|_w}{\|\phi_j\|_w}\psi_{j+1}(x) \\ &= \frac{k_{j-1}}{k_j}\frac{\|\phi_j\|_w}{\|\phi_{j-1}\|_w}\psi_{j-1}(x) - \frac{b_j}{a_j}\psi_j(x) + \frac{k_j}{k_{j+1}}\frac{\|\phi_{j+1}\|_w}{\|\phi_j\|_w}\psi_{j+1}(x) \\ &\equiv \beta_j\psi_{j-1}(x) + \alpha_j\psi_j(x) + \beta_{j+1}\psi_{j+1}(x)\,, \end{aligned} \tag{106}$$

where we have defined

$$\alpha_j = -\frac{b_j}{a_j} \quad (j \geq 0) \quad \text{and} \quad \beta_j = \frac{k_{j-1}}{k_j}\frac{\|\phi_j\|_w}{\|\phi_{j-1}\|_w} \quad (j \geq 1) \quad . \tag{107}$$

It is easily confirmed that β_j is as given in Eq. 104 using Eq. 27 and Eq. 29, i.e.,

$$\sqrt{\frac{c_j}{a_j a_{j-1}}} = \sqrt{\frac{\frac{k_{j-1}k_{j+1}}{k_j^2}\frac{\|\phi_j\|_w^2}{\|\phi_{j-1}\|_w^2}}{\frac{k_j}{k_{j-1}}\frac{k_{j+1}}{k_j}}} = \sqrt{\frac{k_{j-1}^2}{k_j^2}\frac{\|\phi_j\|_w^2}{\|\phi_{j-1}\|_w^2}} = \frac{k_{j-1}}{k_j}\frac{\|\phi_j\|_w}{\|\phi_{j-1}\|_w} = \beta_j\,. \tag{108}$$

The new three-term recurrence relation for the orthonormal polynomials ψ_j, $j = 0, 1, \ldots, n$ gives rise to a new system of linear equations,

$$\begin{aligned} x\psi_0(x) &= \alpha_0\psi_0(x) + \beta_1\psi_1 \\ x\psi_1(x) &= \beta_1\psi_0(x) + \alpha_1\psi_1(x) + \beta_2\psi_2(x) \\ \vdots &= \qquad\qquad \vdots \\ x\psi_n(x) &= \beta_n\psi_{n-1}(x) + \alpha_n\psi_n(x) + \beta_{n+1}\psi_{n+1}(x) \end{aligned} \tag{109}$$

and the new matrix equation is

$$x\begin{pmatrix} \psi_0(x) \\ \psi_1(x) \\ \vdots \\ \vdots \\ \vdots \\ \psi_{n-1}(x) \\ \psi_n(x) \end{pmatrix} = \begin{pmatrix} \alpha_0 & \beta_1 & & & & & \\ \beta_1 & \alpha_1 & \beta_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & & \beta_n & \alpha_n \end{pmatrix} \begin{pmatrix} \psi_0(x) \\ \psi_1(x) \\ \vdots \\ \vdots \\ \vdots \\ \psi_{n-1}(x) \\ \psi_n(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ \beta_{n+1}\psi_{n+1}(x) \end{pmatrix} . \tag{110}$$

This is a tridiagonal symmetric matrix whereas T, in general, is only tridiagonal. Thus, for x a root of ψ_{n+1} and hence also a root of ϕ_{n+1}, we have the matrix equation

$$x\boldsymbol{\psi}(x) = J\boldsymbol{\psi}(x) \ , \tag{111}$$

where the $\boldsymbol{\psi}(x) = (\psi_0(x), \psi_1(x), \ldots, \psi_n(x))^\top = \left(\frac{\phi_0(x)}{\|\phi_0\|_w}, \frac{\phi_1(x)}{\|\phi_1\|_w}, \ldots, \frac{\phi_n(x)}{\|\phi_n\|_w}\right)^\top$.

It is straightforward to see that J is related to T by a diagonal similarity transformation. The vector $\boldsymbol{\phi}$ which contains the unnormalised polynomials as its elements can be transformed to $\boldsymbol{\psi}$ by multiplying by the diagonal matrix,

$$D = \mathrm{diag}\left(\frac{1}{\|\phi_0\|_w}, \frac{1}{\|\phi_1\|_w}, \ldots, \frac{1}{\|\phi_n\|_w}\right) \tag{112}$$

containing the reciprocal of the norms of the polynomials ϕ_j. Its inverse is simply the diagonal matrix which contains each of the norms along the main diagonal. Therefore applying D to the unnormalised eigenvalue equation, Eq. 102,

$$x\boldsymbol{\psi} \equiv xD\boldsymbol{\phi} = DT\boldsymbol{\phi} = DTD^{-1}D\boldsymbol{\phi} = DTD^{-1}\boldsymbol{\psi} = J\boldsymbol{\psi} \ , \tag{113}$$

we see that they are indeed related by a diagonal similarity transformation with the diagonal matrix given by Eq. 112.

When solving for the eigensystem of the matrix J numerically using Matlab's function `eig()`, the function returns unit eigenvectors $\boldsymbol{\chi}(x_j)$ corresponding to the eigenvalue x_j normalised as $\boldsymbol{\chi}^\top(x_j)\boldsymbol{\chi}(x_j) = 1$.

If we invert Eq. 86 for the weights of Gaussian quadrature,

$$\frac{1}{w_j} = \frac{k_n}{k_{n+1}} \frac{\phi_n(x_j)\phi'_{n+1}(x_j)}{\|\phi_n\|_w^2} \tag{114}$$

and use the Christoffel-Darboux formula evaluated at $x = y = x_j$, and $\phi_{n+1}(x_j) \equiv 0$ we see

$$\frac{1}{w_j} = \frac{k_n}{k_{n+1}} \frac{1}{\|\phi_n\|_w^2} \frac{k_{n+1}}{k_n} \sum_{i=0}^{n} \frac{\|\phi_n\|_w^2}{\|\phi_i\|_w^2} \phi_i^2(x_j) = \sum_{i=0}^{n} \frac{\phi_i^2(x_j)}{\|\phi_i\|_w^2} = \boldsymbol{\psi}(x_j)^\top \boldsymbol{\psi}(x_j) \ . \tag{115}$$

This leads us to

$$1 = w_j \boldsymbol{\psi}(x_j)^\top \boldsymbol{\psi}(x_j) \tag{116}$$

and since $\boldsymbol{\psi}(x_j)$ is also an eigenvector of J, corresponding to the eigenvalue x_j, it only differs from $\boldsymbol{\chi}(x_j)$ by a constant, clearly

$$\boldsymbol{\chi}(x_j) = \sqrt{w_j} \boldsymbol{\psi}(x_j) \ . \tag{117}$$

Considering just the first component of each eigenvector in Eq. 117 one obtains a simple expression for the Gaussian weights,

$$\begin{aligned}
w_j &= \left(\frac{(\boldsymbol{\chi}(x_j))_0}{(\boldsymbol{\psi}(x_j))_0} \right)^2 \\
&= \left(\frac{(\boldsymbol{\chi}(x_j))_0}{\frac{\phi_0(x)}{\|\phi_0\|_w}} \right)^2 \\
&= \frac{(\boldsymbol{\chi}(x_j))_0^2}{\phi_0^2(x)} \|\phi_0\|_w^2 \\
&= \frac{(\boldsymbol{\chi}(x_j))_0^2}{k_0^2} (\phi_0, \phi_0) \\
&= \frac{(\boldsymbol{\chi}(x_j))_0^2}{k_0^2} \int_a^b \phi_0^2(x) w(x) dx \\
&= \frac{(\boldsymbol{\chi}(x_j))_0^2}{k_0^2} k_0^2 \int_a^b w(x) dx \\
&= (\boldsymbol{\chi}(x_j))_0^2 \int_a^b w(x) dx \\
&\equiv (\boldsymbol{\chi}(x_j))_0^2 \times \mu_0 \ ,
\end{aligned} \tag{118}$$

where in the last line we have defined the constant μ_0 as the zeroth moment, i.e.,

$$\mu_0 \equiv \int_a^b w(x) dx \ . \tag{119}$$

This constant for the four classical polynomials already mentioned are given in Table 3 along with the tridiagonal terms appearing in the Jacobi matrix.

6.1.2 A few simple examples

Example 6.1. (Gauss-Legendre Quadrature) As an example application of Gauss-Legendre quadrature we evaluate the analytically known integral

$$\int_{-1}^{1} dx \, \frac{x^2}{5 + x^2} = 2 - 2\sqrt{5} \arctan\left(\frac{1}{\sqrt{5}} \right) \ . \tag{120}$$

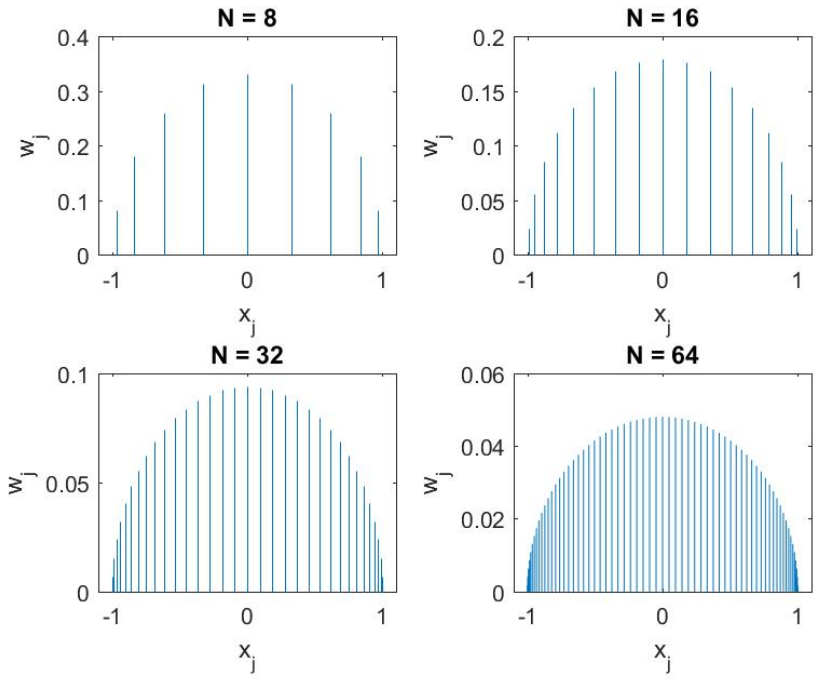


Figure 3: Gauss-Legendre quadrature weights at nodes x_j. These plots are produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

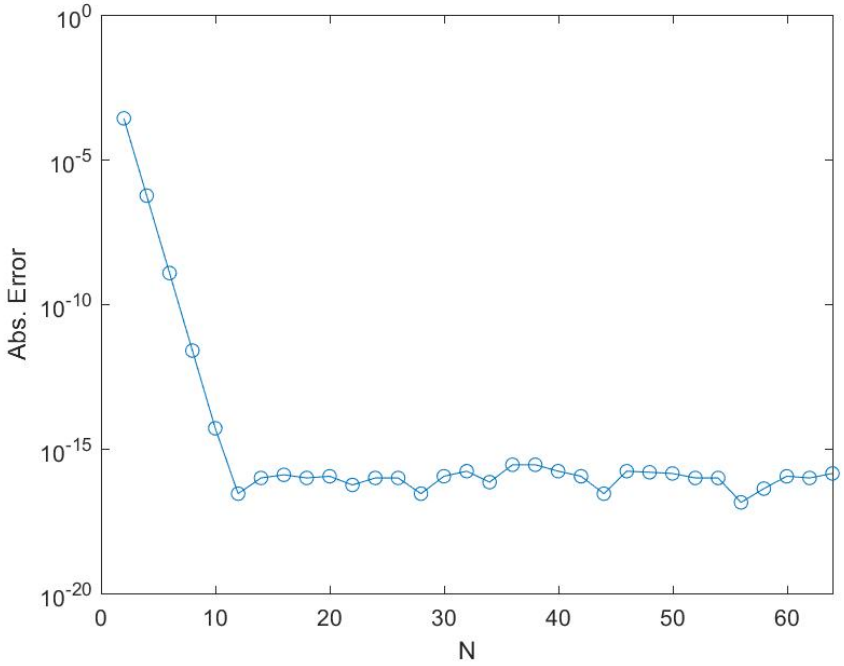


Figure 4: Gauss-Legendre quadrature example error. This plot is produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

Table 3: Summary of the tridiagonal terms in the Jacobi matrix and the zeroth moment for the four most used orthogonal polynomials. The Chebyshev polynomials are of the 1st kind and their nodes and weights can also be found using the analytic formulas provided in Table 4.

Quadrature Name	α_j	β_j	μ_0
Gausss-Legendre	$\alpha_j = 0,\ j = 0, 1, \ldots, n$	$\beta_j = \frac{j}{\sqrt{2j^2-1}},\ j = 1, 2, \ldots, n$	2
Gauss-Chebyshev	$\alpha_j = 0,\ j = 0, 1, \ldots, n$	$\beta_1 = \frac{1}{\sqrt{2}},\ \beta_j = \frac{1}{2},\ j = 2, 3, \ldots n$	π
Gauss-Laguerre	$\alpha_j = 2j + 1,\ j = 0, 1, \ldots n$	$\beta_j = j,\ j = 1, 2, \ldots, n$	1
Gauss-Hermite	$\alpha_j = 0,\ j = 0, 1, \ldots, n$	$\beta_j = \sqrt{\frac{j}{2}},\ j = 1, 2, \ldots, n$	$\sqrt{\pi}$

The weights and nodes of Gauss-Legendre quadrature are found for several values of n and plotted in Fig. 3. We also evaluate the absolute error for the integral in Eq. 120 which is shown in Fig. 4. It is clear that Gauss-Legendre performs very well for this integral.

For large n, better algorithms are available such as Glaser *et al.* [31], Hale and Townsend [32] and [33], the latter replacing the two former methods in the more recent version of Chebfun's `legpts()` function [27].

Example 6.2. (Gauss-Chebyshev Quadrature) We apply Gauss-Chebyshev quadrature to two analytically known integrals. The first is

$$\int_{-1}^{1} \frac{dx}{\sqrt{1-x^2}} \frac{x^2}{5+x^2} = \pi \left(1 - \frac{1}{\sqrt{5}}\right) , \tag{121}$$

which contains the Chebyshev weight function in the integrand and the second is given by Eq. 120 which does not contain the Chebyshev weight function. The weights and nodes of Gauss-Chebyshev quadrature are found for several values of n and plotted in Fig. 5. They can alternatively be found by the their known analytic expressions as given in Table 4. We also evaluate the absolute error for the two integrals in Eq. 121 and 120 which is shown in Fig. 6. It is clear that Gauss-Chebyshev performs very well for the integral which contains the Chebyshev weight, but nowhere near as well for the integrand which does not contain it. We will return to this when we discusss Clenshaw-Curtis quadrature [34, 35] in Section 6.3.

Example 6.3. (Gauss-Laguerre Quadrature) The obvious choice for an example of Gauss-Laguerre quadrature is the gamma function which interpolates $n!$,

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt \ . \tag{122}$$

Choosing $\alpha = 13$, the exact answer is $\Gamma(13) = (13-1)! = 12! = 479\ 001\ 600$. We compute the weights and nodes for several values of n and plot them in Fig. 7. We also evaluate the error for this integral which is shown in Fig. 8. It is clear that Gauss-Laguerre performs very well for this integral which contains the Laguerre weight. However, the natural domain of this quadrature method is the semi-infinite interval, so the nodes become very widely spaced and the weights become correspondingly small very quickly. This eigenvalue method of finding the nodes and weights is not ideal when n is large. To avoid inaccurate values other methods could be used, such methods are provided by Glaser *et al* [31] and Vanlessen [36] and implemented in Chebfun's `lagpts()` function [27].

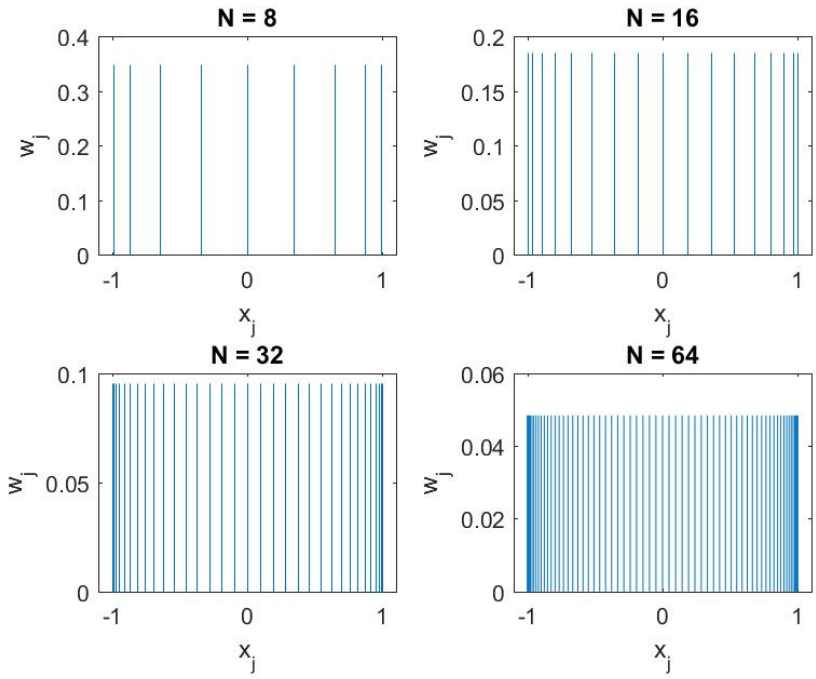


Figure 5: Gauss-Chebyshev quadrature weights at nodes x_j. These plots are produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

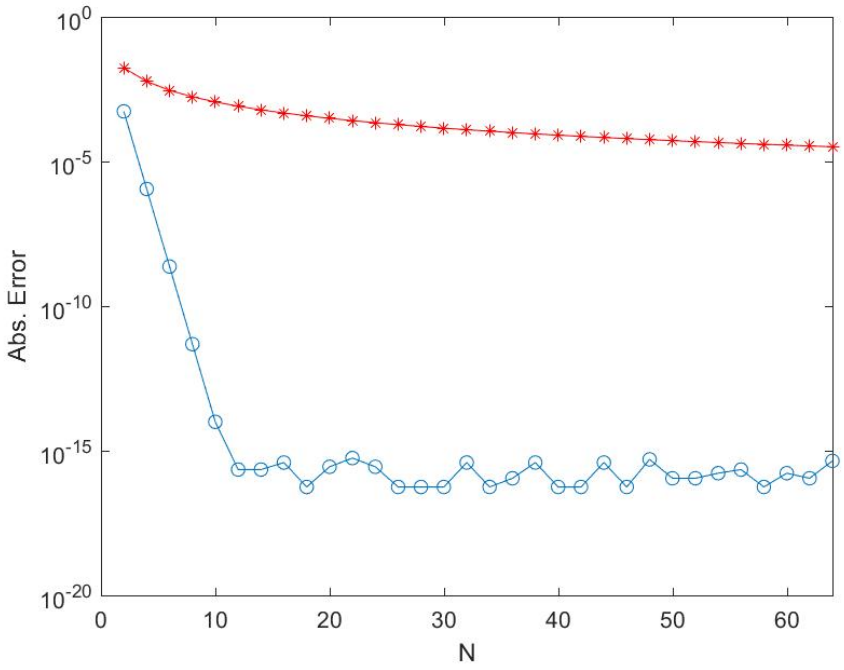


Figure 6: Gauss-Chebyshev quadrature example error. The blue curve corresponds to Eq. 121 which has the integrand that includes the Chebyshev weight and the red curve to Eq. 120 which does not. This plot is produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

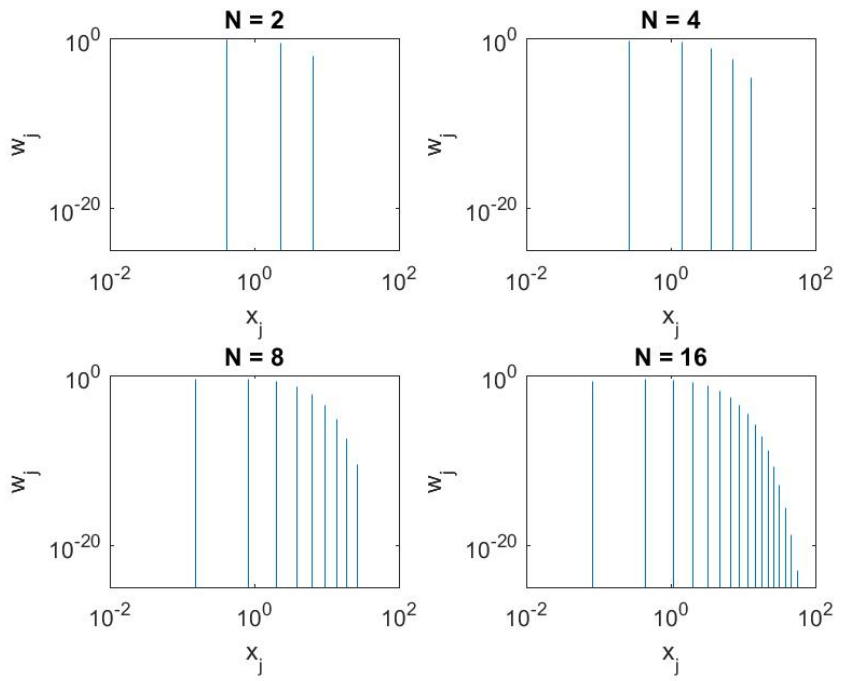


Figure 7: Gauss-Laguerre quadrature weights at nodes x_j. These plots are produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

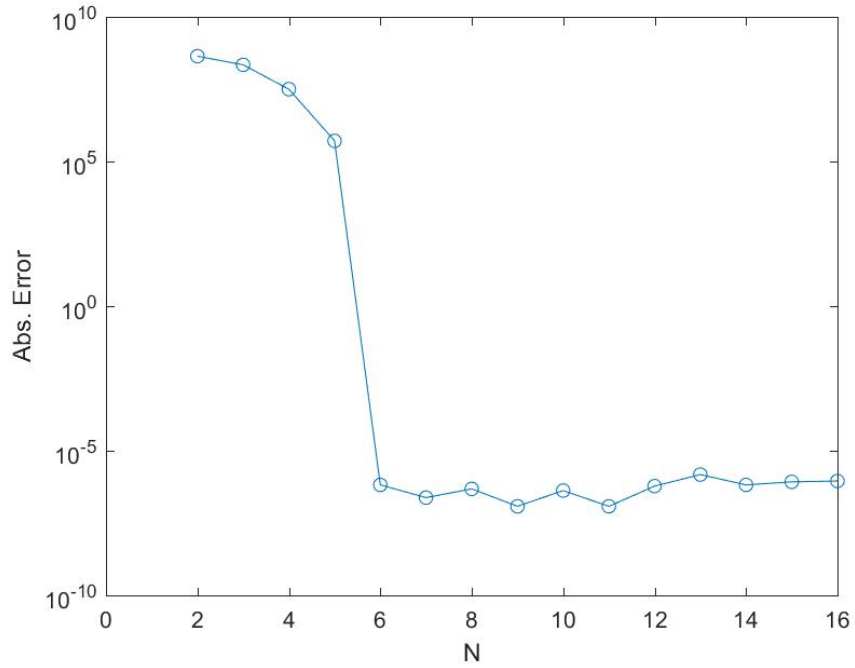


Figure 8: Gauss-Laguerre quadrature example error. This plot is produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

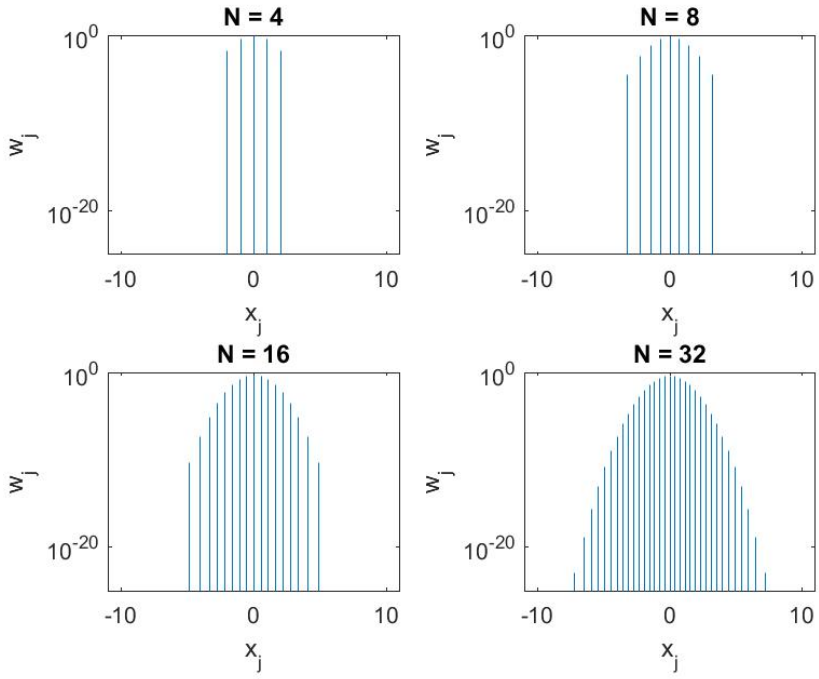


Figure 9: Gauss-Hermite quadrature weights at nodes x_j. These plots are produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

Example 6.4. (Gauss-Hermite Quadrature) As an example application of Gauss-Hermite quadrature we evaluate the analytically know integral

$$\int_{-\infty}^{\infty} dx\, e^{-x^2} \frac{1}{1+x^2} = e\pi \mathrm{Erfc}(1)\ . \tag{123}$$

The weights and nodes of Gauss-Hermite quadrature are found for several values of n and plotted in Fig. 9. We also evaluate the error for the integral in Eq. 123 which is shown in Fig. 10. Despite the integral being over the infinite interval it is clear that Gauss-Hermite quadrature performs reasonably well for this integral. Moreover, the weight function dies off fast as a function of x, even faster than the Laguerre weight function, so the nodes are not as far apart as for Gauss-Laguerre quadrature. The fast methods of Glaser *et al.* [31] and Townsend *et al.* [37] are implemented in Chebfun's [27] `hermpts()` function. These methods would be more appropriate than the Golub-Welsch algorithm for large N.

6.1.3 Gauss-Radau and Gauss-Lobatto quadrature

For our application of solving a differential equation using pseudo-spectral methods we will be using the nodes of a form of Gaussian quadrature as our collocation grid. This will be discussed further in Section 8. As we have presented in the previous sections, the nodes of Gaussian quadrature all lie in the interior of the interval $[a, b]$. Extending Gaussian quadrature by preassigning some of the nodes, particularly including the boundary points, is a very important consideration. This is especially so for some applications such as boundary value problems, where one will want to implement boundary conditions. The two most common examples

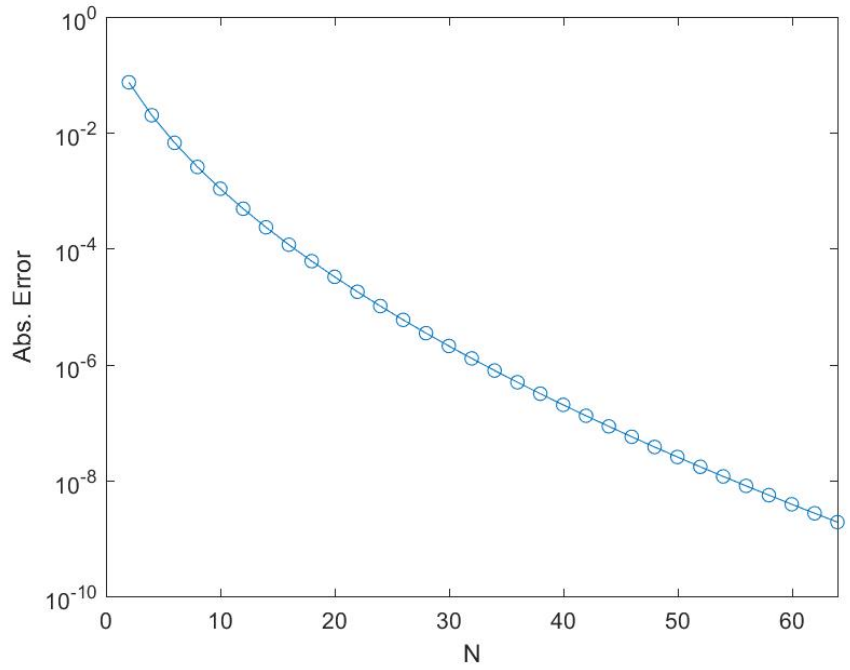


Figure 10: Gauss-Hermite quadrature example error. This plot is produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

of this kind of extension are known as Gauss-Radau and Gauss-Lobatto quadrature, where one or both of the boundary points are included as preassigned nodes, respectively. Once the preassigned nodes are chosen, the remaining nodes and quadrature weights are then adjusted to maximise the degree of exactness of the quadrature rule. Obviously, for each preassigned node the largest degree polynomial that can be integrated exactly is reduced by one. Therefore, if all $n+1$ points of an $n+1$ point quadrature rule are preassigned then the largest degree polynomial that can be exactly integrated is $2n+1-(n+1)=n$. This is something we have already shown in Eqs. 69–71.

As we have already seen, the nodes of an $n+1$ point Gaussian quadrature rule are zeros of the $n+1$ degree polynomial $\phi_{n+1}(x)$ of a set of polynomials defined to be orthogonal on $[a,b]$ with respect to the weight function $w(x)$. The Gauss-Radau nodes are one (and only one) of the boundary points and what turns out to be the zeros of an associated orthogonal polynomial,

$$\varphi_n(x) = \frac{\phi_{n+1}(x) + \alpha_n \phi_n(x)}{x-a} \ , \tag{124}$$

where the constant α_n is chosen such that numerator of φ_n evaluates to zero for a the left boundary point, i.e., $\phi_{n+1}(a) + \alpha_n \phi_n(a) = 0$. This polynomial defines a new sequence of polynomials which are orthogonal with respect to a modified weight function $\tilde{w}(x) = (x-a)w(x)$. Alternatively, taking the right boundary point b we have

$$\varphi_n(x) = \frac{\phi_{n+1}(x) + \alpha_n \phi_n(x)}{b-x} \ , \tag{125}$$

where the constant α_n is once again chosen such that numerator of φ_n evaluates to zero at the included boundary point. The modified weight function is then $\tilde{w}(x) = (b-x)w(x)$.

Table 4: Summary of quadrature formulas for Chebyshev polynomials [15].

Quadrature Name	Nodes x_j	Weights w_j
Gauss-Chebyshev	$\cos\left(\frac{(j+\frac{1}{2})\pi}{n+1}\right)$	$w_j = \frac{\pi}{n+1}$
Gauss-Radau-Chebyshev	$\cos\left(\frac{2\pi j}{2n+1}\right)$	$w_0 = \frac{\pi}{2n+1}$, $w_j = \frac{2\pi}{2n+1}$ for $j \geq 1$
Gauss-Lobatto-Chebyshev	$\cos\left(\frac{\pi j}{n}\right)$	$w_0 = w_n = \frac{\pi}{2n}$, $w_j = \frac{\pi}{n}$ for $j = 1, \ldots, n-1$

The nodes of Gauss-Lobatto quadrature are the boundary points and the zeros of a similarly defined polynomial

$$\varphi_{n-1}(x) = \frac{\phi_{n+1}(x) + \alpha_n \phi_n(x) + \beta_n \phi_{n-1}(x)}{(x-a)(b-x)} \ , \tag{126}$$

where the constants are chosen such that numerator equals zero on the boundary points. Analogously, this polynomial defines a new sequence of polynomials which are orthogonal with respect to the modified weight function $\tilde{w}(x) = (x-a)(b-x)w(x)$. In the case of Chebyshev and Legendre polynomials the nodes are boundary points and the extrema of the polynomial $\phi_n(x)$, i.e., the zeros of $(1-x^2)\phi_n'(x)$.

For the special case of the Chebyshev polynomials not only the nodes and weights of Gaussian quadrature are known analytically, but closed form expressions for the nodes and weights of Gauss-Radau and Gauss-Lobatto quadrature are known as well. All three are summarised in Table 4, see also [12–16]. For other orthogonal polynomials a modified version of the Golub-Welsch algorithm can be used, see Ref. [38] for details. In what follows we will only use the Gauss-Lobatto quadrature when solving the Schrödinger equation with a Chebyshev pseudo-spectral method, using the preassigned nodes to enforce the boundary condition that the wave function be zero on the boundary. Nonetheless, in applying pseudo-spectral methods to the Schrödinger equation with a more diverse set of potentials and to alternative evolution equations, the other quadrature methods may be more relevant.

Proposition 6.3. *(Gauss-Lobatto-Chebyshev nodes) The nodes of the $n+1$ point Gauss-Lobatto-Chebyshev quadrature are given by ± 1 and the extrema of $T_n(x)$.*

Proof: The constants α_n and β_n are defined such that the numerator in Eq. 126 is zero when $x = a$ or $x = b$, i.e.,

$$\phi_{n+1}(a) + \alpha_n \phi_n(a) + \beta_n \phi_{n-1}(a) = 0 \tag{127}$$

$$\phi_{n+1}(b) + \alpha_n \phi_n(b) + \beta_n \phi_{n-1}(b) = 0 \ . \tag{128}$$

Equations 127 and 128 can be re-written as the matrix equation

$$\begin{pmatrix} \phi_n(a) & \phi_{n-1}(a) \\ \phi_n(b) & \phi_{n-1}(b) \end{pmatrix} \begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = - \begin{pmatrix} \phi_{n+1}(a) \\ \phi_{n+1}(b) \end{pmatrix} \ . \tag{129}$$

The solution of Eq. 129 is easily found to be

$$\begin{pmatrix} \alpha_n \\ \beta_n \end{pmatrix} = \frac{1}{\phi_n(a)\phi_{n-1}(b) - \phi_n(b)\phi_{n-1}(a)} \begin{pmatrix} \phi_{n-1}(a)\phi_{n+1}(b) - \phi_{n-1}(b)\phi_{n+1}(a) \\ \phi_n(b)\phi_{n+1}(a) - \phi_n(a)\phi_{n+1}(b) \end{pmatrix} \ . \tag{130}$$

Substituting the boundary points ± 1 and the solution for the constants α_n and β_n given in Eq. 130 into Eq. 126 we find

$$\varphi_{n-1}(x) = \frac{T_{n+1}(x) + \frac{T_{n-1}(-1)T_{n+1}(1) - T_{n-1}(1)T_{n+1}(-1)}{T_n(-1)T_{n-1}(1) - T_n(1)T_{n+1}(-1)} T_n(x) + \frac{T_n(1)T_{n+1}(-1) - T_n(-1)T_{n+1}(1)}{T_n(-1)T_{n-1}(1) - T_n(1)T_{n-1}(-1)} T_{n-1}(x)}{1 - x^2} . \tag{131}$$

Using the properties of Chebyshev polynomials given in Appendix B.1, specifically

$$T_n(1) = 1 \quad \text{and} \quad T_n(-x) = (-1)^n T_n(x) , \tag{132}$$

Eq. 131 simplifies to

$$\varphi_{n-1}(x) = \frac{T_{n+1}(x) - T_{n-1}(x)}{1 - x^2} . \tag{133}$$

At this point we use the recurrence relation for the derivative of Chebyshev polynomials (Eq. B14) and the three-term recurrence relation (Eq. B13) to simplify and rearrange Eq. 133 to

$$(1 - x^2)\varphi_{n-1}(x) = T_{n+1}(x) - T_n(x) = -\frac{2}{n}(1 - x^2)T_n'(x) . \tag{134}$$

Thus the nodes are the extrema of $T_n(x)$ and the boundary points ± 1. ■

6.2 Composite trapezoidal rule and periodic functions

The composite rectangular/midpoint and trapezoidal rules are very widely known, so we will not discuss them in any detail. We will define them and simply point out their importance for periodic functions and hence for the Fourier pseudo-spectral method. The N point quadrature rules approximating the integral

$$I(f) = \int_a^b f(x)dx \tag{135}$$

is given by

$$I_n(f) = h \sum_{j=1}^{N} f(x_j) \tag{136}$$

for the composite rectangular rule and

$$I_n(f) = h \sum_{j=1}^{N}{}'' f(x_j) \tag{137}$$

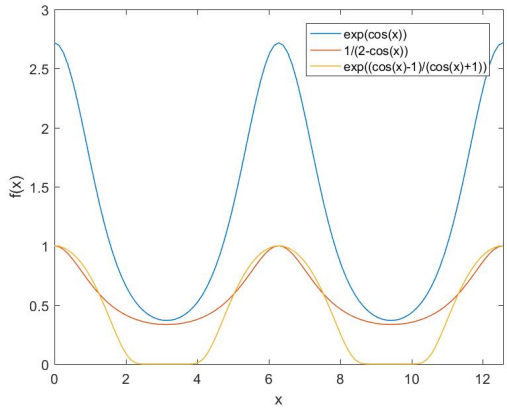
for the composite trapezoidal rule. The function evaluations are performed on the uniformly spaced grid $x_j = jh$ where

$$h = \frac{b - a}{N} \tag{138}$$

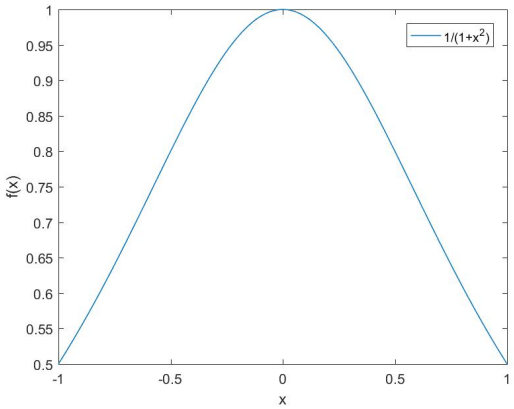
and the double prime on the summation symbol indicates that the first and last terms of the series are multiplied by half.

It turns out that despite what may be said in introductory numerical analysis texts, the composite trapezoidal rule is not necessarily $\mathcal{O}(N^{-2})$. In fact, it can be a very accurate approximation to an integral. That is, if the integral is over the period of a periodic function. However, for non-periodic functions it is still $\mathcal{O}(N^{-2})$.

The composite trapezoidal rule as well as the composite rectangular rule are the analogues of Gaussian quadrature for periodic functions. For N quadrature points the rule will exactly



(i) A few periodic functions.



(ii) A non-periodic function.

Figure 11: A few periodic and non-periodic functions. These plots are produced by using the MATLAB® script `Ex_Script_3_Trapezoidal_Rule.m`.

integrate trigonometric polynomials of degree $2N-2$, i.e., for polynomials composed of a constant, the first $N-2$ cosines and the first $N-2$ sines [12, 39]. For this reason we will be using these rules for normalising wave functions on the Fourier grid. We will also use the trapezoidal rule in defining Clenshaw-Curtis quadrature in the next section.

We illustrate these statements by applying the composite trapezoidal rule to integrate several functions. In Figs. 11i we show several periodic functions and in Fig. 12 we show the error in using the composite trapezoidal rule to integrate these functions over $[0, 2\pi]$. In Fig. 11ii we show a non-periodic function and in Fig. 12 we show the corresponding error of integrating this function over $[-1, 1]$. The last three functions in the legend of Fig. 12 are not integrated, they are merely plotted to illustrate the rates of convergence. As can clearly be seen in Fig. 12, the typical statement that the trapezoidal rule is $\mathcal{O}(N^{-2})$ is certainly pessimistic.

6.3 Clenshaw-Curtis quadrature

Interpolatory quadrature rules rely on being able to interpolate the integrand accurately and then integrating the resulting interpolant easily. An idea we want to drive home is that poorly choosing the distribution of nodes can lead to disastrous consequences and that this choice is dependent on the nature of the function being interpolated.

Gaussian quadrature as presented in earlier sections is an interpolatory quadrature method. As we saw in Section 6.1, to optimally integrate polynomials up to degree $2n+1$ for an $n+1$ quadrature rule, not only were the quadrature weights adjusted but also the nodes as well. The nodes turned out to be zeros of an orthogonal polynomial which in most cases must be computed numerically. This can be accomplished using the algorithm of Golub and Welsch which translates the root-finding problem to an eigenvalue problem requiring $\sim \mathcal{O}(n^2)$ operations in principle. The matlab implementation we used is actually $\mathcal{O}(n^3)$ because it didn't exploit the tridiagonal form of the Jacobi matrix [35, 40].

Recall in Section 5.4 where we looked at Runge's function and we tried to interpolate it on

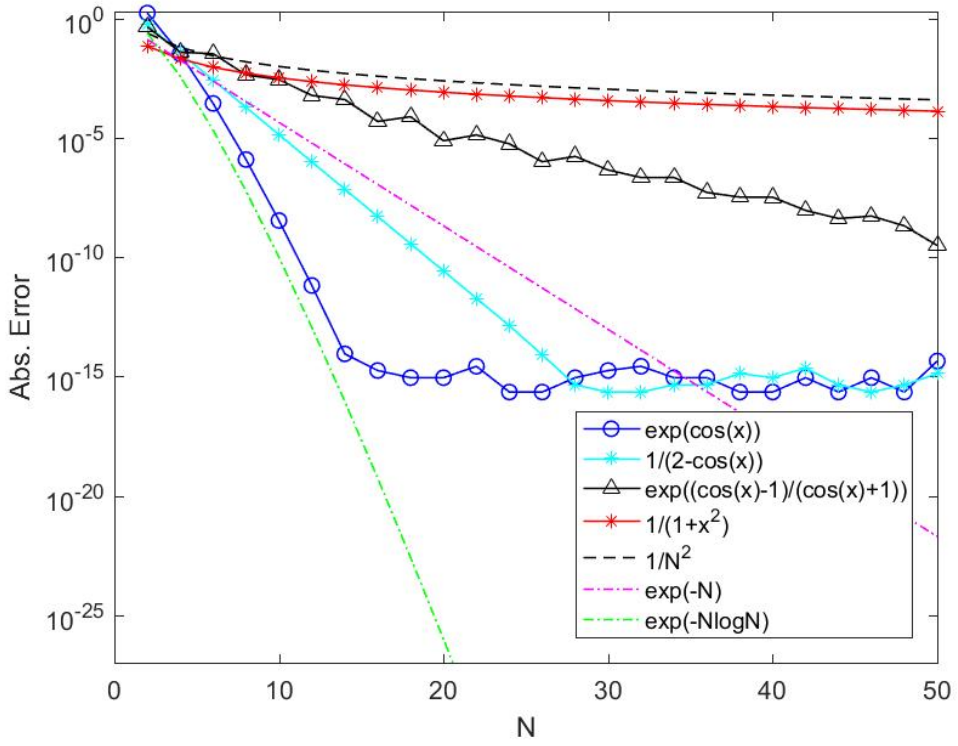


Figure 12: Error in using the trapezoidal rule for a selection of functions as shown in Figs. 11i and 11ii. The last three functions in the legend are not integrated, they are merely plotted to illustrate the rates of convergence. This plot is produced by using the MATLAB® script ***Ex_Script_3_Trapezoidal_Rule.m****.*

a uniformly spaced grid. This non-periodic function interpolated on this equally spaced grid led to the infamous Runge phenomenon, i.e., wild oscillations near the boundaries. This phenomenon is responsible for the divergence of the well known Newton-Cotes quadrature rule in many situations and only converges when the integrand is analytic in a large region about interval of integration [35]. By using a more appropriately distributed set of nodes for this function, such as Chebyshev points of the second kind (i.e., ± 1 and the extrema of $T_n(x)$)

$$x_j = \cos\left(\frac{j\pi}{n}\right) \ , \tag{139}$$

which are not uniformly distributed and clustered more around the boundaries Runge phenomenon can be avoided.

In 1960 Clenshaw and Curtis suggested a numerical integration technique which is not technically as efficient as Gaussian quadrature, but in many situations it can compete very well with it. What they did was to expand the integrand in a Chebyshev series and integrate it. Their method works very well because it is very easy to integrate Chebyshev polynomials and more importantly if the function f to be integrated is even a little smooth (Lipschitz continuous), then the series will converge meaning that the integral should also be increasingly more accurate as we increase the degree of the interpolating polynomial [11].

If we let f be a continuous function on $[-1,1]$ of bounded variation, then without loss of generality we consider the approximation of the integral

$$I(f) = \int_{-1}^{1} f(x)dx \tag{140}$$

by the $n+1$ point quadrature rule

$$I_n(f) = \sum_{j=0}^{n} w_j f(x_j) \ . \tag{141}$$

Equation 141 can be written in matrix form as $\mathbf{w}^\top \mathbf{f}$ where $\mathbf{w}$ and $\mathbf{f}$ are the $n+1$-dimensional vectors of weights w_j and function evaluations $f(x_j)$, respectively.

Clenshaw-Curtis quadrature uses the Chebyshev points of the second kind as the interpolation grid. As we are not adjusting both the nodes and the weights to exactly integrate polynomials of the largest degree possible but rather choosing the nodes to be the extrema of the Chebyshev polynomials, the maximum degree polynomial that this integration method can integrate exactly is n rather than $2n+1$, as is the case for Gauss-Legendre quadrature. This can be shown using the Lagrange form of the interpolant

$$p_n(x) = \sum_{j=0}^{n} f(x_j)\ell_j(x) \tag{142}$$

we see that

$$\int_{-1}^{1} f(x)dx \simeq \int_{-1}^{1} p_n(x)dx = \sum_{j=0}^{n} f(x_j) \int_{-1}^{1} \ell_j(x)dx = \sum_{j=0}^{n} w_j f(x_j) \ , \tag{143}$$

where we have defined

$$w_j = \int_{-1}^{1} \ell_j(x)dx \ . \tag{144}$$

The Clenshaw-Curtis quadrature rule is therefore exact only for all polynomials up to and including degree n, i.e., $\forall p \in \mathbb{P}_n$.

It is not convenient to calculate the Clenshaw-Curtis quadrature weights by Eq. 144 and since we are yet to discuss Chebyshev series, and more generally polynomial transforms, we now instead rely on the readers knowledge of Fourier analysis to present Clenshaw-Curtis quadrature because Chebyshev series are just Fourier Cosine series in disguise.

The integrand in Eq. 140, $f(x)$, has not been assumed to be periodic, but we can make it periodic by parametrising x through the following change of variables

$$x = \cos\theta \quad , \quad dx = -\sin\theta d\theta \tag{145}$$

translating Eq. 140 to

$$\int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\theta)\sin\theta d\theta \ . \tag{146}$$

The desired integral now has a 2π periodic integrand. However, unfortunately we are not able to exploit the rapidly converging trapezoidal rule because the integration is not over the entire

domain of periodicity. Be that as it may, $f(\cos\theta)$ does have a Fourier series representation and since it is an even function only the cosine terms contribute,

$$f(\cos\theta) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\theta) \ , \tag{147}$$

where Fourier coefficients are given by

$$a_i = \frac{1}{\pi}\int_0^{2\pi} f(\cos\theta)\cos(i\theta)d\theta \ . \tag{148}$$

Now we are in luck, because this integral is over a whole period so we can use the trapezoidal rule. Making the observation that this integrand is an even function allows us to half the domain of integration by just doubling the integral

$$a_i = \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)\cos(i\theta)d\theta \ . \tag{149}$$

Applying the trapezoidal rule (Eq. 137) to Eq. 149 we obtain the approximate expression

$$a_i \simeq \frac{2}{\pi}\frac{(\pi-0)}{n}\sum_{j=0}^{n}{}'' f(\cos\theta_j)\cos(i\theta_j) = \frac{2}{n}\sum_{j=0}^{n}{}'' f(\cos\theta_j)\cos(i\theta_j) \tag{150}$$

for the Fourier coefficients where the equally spaced points have been denoted by $\theta_j = \frac{j\pi}{n}$, $j = 0, 1, \ldots, n$. To obtain a quadrature rule of the form of Eq. 141 this expression for the Fourier coefficients must be written in terms of function evaluations on the unevenly spaced Chebyshev points $x_j = \cos\theta_j$, such that

$$a_i \simeq \sum_{j=0}^{n}{}'' M_{ij} f(x_j) \ , \tag{151}$$

where we have defined

$$M_{ij} \equiv \begin{cases} \frac{1}{n}\cos\left(\frac{ij\pi}{n}\right) & , \quad j = 0 \quad or \quad n \\ \\ \frac{2}{n}\cos\left(\frac{ij\pi}{n}\right) & , \quad j = 1, 2, \ldots, n-1 \end{cases} \tag{152}$$

Thus the coefficients a_i, $i = 0, 1, \ldots, n$ can be encapsulated in vector form

$$\mathbf{a} = (a_0, a_1, \ldots, a_n)^{\top} \tag{153}$$

and calculated by matrix product between the matrix

$$M = \frac{2}{n}\begin{pmatrix} \frac{1}{2} & 1 & \ldots & \ldots & \ldots & 1 & \frac{1}{2} \\ \frac{1}{2} & \cos\frac{\pi}{n} & \ldots & \ldots & \ldots & \cos\frac{(n-1)\pi}{n} & \frac{1}{2}\cos\pi \\ \vdots & & \ddots & & & & \vdots \\ \vdots & & & \ddots & & & \vdots \\ \vdots & & & & \ddots & & \vdots \\ \frac{1}{2} & \cos\frac{(n-1)\pi}{n} & \ldots & \ldots & \ldots & \cos\frac{(n-1)^2\pi}{n} & \frac{1}{2}\cos(n-1)\pi \\ \frac{1}{2} & \cos\pi & \ldots & \ldots & \ldots & \cos(n-1)\pi & \frac{1}{2}\cos n\pi \end{pmatrix} \tag{154}$$

defined by Eq. 152 and the vector of function evaluations

$$\mathbf{f} = (f(x_0), f(x_1), \ldots, f(x_n))^\top \ . \tag{155}$$

Explicitly, that is

$$\mathbf{a} = M\mathbf{f} \ . \tag{156}$$

Now returning to the our main calculation of $I(f)$, using the Fourier Cosine representation of $f(\cos\theta)$ in Eq. 146 we find that

$$\begin{aligned} I(f) &= \int_0^\pi f(\cos\theta)\sin\theta d\theta \\ &= \int_0^\pi \left(\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(n\theta)\right)\sin\theta d\theta \\ &= \frac{a_0}{2}\int_0^\pi \sin\theta d\theta + \sum_{n=0}^{\infty} a_n \int_0^\pi \cos(n\theta)\sin\theta d\theta \\ &= a_0 + \sum_{n=0}^{\infty} a_n \left(\frac{1+\cos n\pi}{1-n^2}\right) \\ &= a_0 + \sum_{\substack{n=0 \\ n\,even}}^{\infty} \frac{2a_n}{1-n^2} \ . \end{aligned} \tag{157}$$

It well known that the Fourier coefficients for a smooth and periodic function decay extremely fast. Thus, since our integral $I(f)$ has been written in terms of a converging series of these rapidly decaying Fourier coefficients we can confidently truncate this infinite series to obtain an approximation to it. Truncating the series at order N defines the Clenshaw-Curtis approximation to $I(f)$. This finite series approximation of $I(f)$ can obviously be written as a vector inner product, i.e.,

$$I_N(f) = a_0 + \sum_{\substack{n=0 \\ n\,even}}^{N} \frac{2a_n}{1-n^2} = \tilde{\mathbf{w}}^\top \mathbf{a} \tag{158}$$

where the elements of $\tilde{\mathbf{w}}$ are

$$\tilde{w}_j = \begin{cases} 1 & , \quad j = 0 \\ \frac{2}{1-j^2} & , \quad j \neq 0 \ and \ even \\ 0 & , \quad j \ odd \end{cases} \ . \tag{159}$$

Employing the vector equation for the Fourier coefficients we obtained above in Eq. 156 the Clenshaw-Curtis approximation to the integral can be obtained in terms of function evaluations as follows

$$I_N(f) = \tilde{\mathbf{w}}^\top \mathbf{a} = \tilde{\mathbf{w}}^\top M\mathbf{f} = \mathbf{w}^\top \mathbf{f} = \sum_{j=0}^{N} w_j f(x_j) \ . \tag{160}$$

The explicit form of the Clenshaw-Curtis quadrature weights for even N are

$$w_j = \begin{cases} \frac{1}{N^2-1} & , \quad j = 0 \quad or \quad N \\ \frac{2}{N}\left[1 + \sum_{i=1}^{N/2-1}\left(\frac{2}{1-4i^2}\right)\cos\left(\frac{2ij\pi}{N}\right) + \frac{\cos j\pi}{1-N^2}\right] & , \quad j = 1, 2, \ldots, N-1 \end{cases} \tag{161}$$

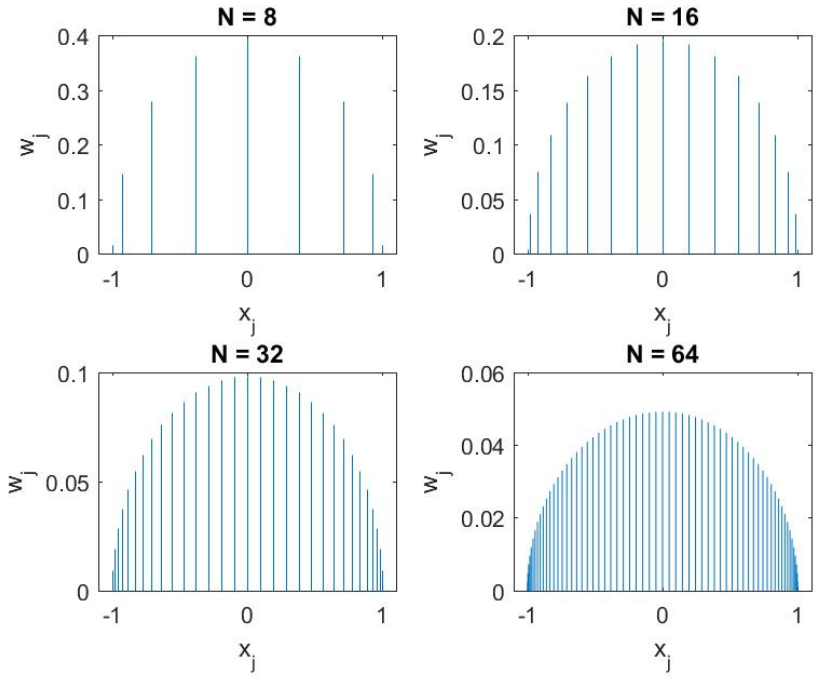


Figure 13: Clenshaw-Curtis weights as computed by Chebfun's `chebpts()` *function [27]. These plots are produced by using the MATLAB® script* `Ex_Script_2_Numerical_Quadrature.m`.

and similarly for odd N they are

$$w_j = \begin{cases} \frac{1}{N^2} & , \quad j = 0 \quad or \quad N \\ \frac{2}{N}\left[1 + \sum_{i=1}^{(N-1)/2} \left(\frac{2}{1-4i^2}\right) \cos\left(\frac{2ij\pi}{N}\right)\right] & , \quad j = 1, 2, \ldots, N-1 \end{cases} . \tag{162}$$

One may choose to work with the quadrature weights explicitly or utilise the FFT to calculate the integral explicitly using Eq. 157. Reference [10] provides a simple matlab function `clencurt()` to calculate these weights and nodes, whereas Ref. [35] gives the alternative matlab function `clenshaw_curtis()` taking the path of the FFT to efficiently perform Clenshaw-Curtis integration of a function using the Fourier coefficients. In our later calculations we will use the Chebfun [27] function `chebpts()` to obtain the Clenshaw-Curtis weights. In Fig. 13 we show the weights plotted at the nodes as calculated by `chebpts()`. We see that these weights and nodes are strikingly similar to ones obtained using Gauss-Legendre quadrature.

Even though it cannot technically be exact for polynomials of degree greater than $n+1$, in many situations it is just as good as Gaussian quadrature, see Ref. [35] for thorough comparison and discussion.

Now that we have presented Clenshaw-Curtis quadrature we would like to make a comparison to Gaussian quadrature. Examples of Gauss-Chebyshev quadrature were shown in Section 6.1. We saw that this quadrature rule works best if the integrand is the Chebyshev weight function multiplied by a polynomial (or function which can be well approximated by a polynomial).

However, it can still be used to integrate other functions which are not explicitly defined as such. This was done for the function given in Eq. 120. In this example, this was accomplished by rewriting the function, i.e., pulling out the weight function, effectively defining a new function. This new function is the one that is evaluated on the quadrature nodes. In Fig. 6, we saw that Gauss-Chebyshev quadrature performs significantly better for Eq. 121 than it did for the example given in Eq. 120. On the other hand, Gauss-Legendre quadrature which has the simplest weight function, $w(x) = 1$, worked beautifully for this example as demonstrated in Fig. 4.

The nodes of Gauss-Chebyshev quadrature are the Chebyshev points of the first kind, i.e, the zeros of T_{N+1}. The grid used in the Clenshaw-Curtis quadrature rule discussed here uses the Chebyshev points of the second kind, i.e., ± 1 and the extrema of T_N. The corresponding Gaussian quadrature on these nodes is Gauss-Chebyshev-Lobatto quadrature which includes the boundary points but this quadrature rule works best if the integrand is of the form of the Chebyshev weight function multiplied by a simple function. Thus, the closest Gaussian quadrature rule to compare Clenshaw-Curtis to is actually Gauss-Legendre quadrature, even though the quadrature nodes in this instance are the zeros of the Legendre polynomial P_{N+1}.

In Fig. 14 we compare the Clenshaw-Curtis, Gauss-Chebyshev and Gauss-Legendre quadrature rules applied to Eq. 120. Gauss-Legendre is obviously the superior method for this example. This is to be expected since it was defined as the $n+1$ point quadrature to optimally integrate a polynomial of degree $2n+1$. Whereas, Gauss-Chebyshev was defined to exactly integrate a polynomial of degree $2n+1$ multiplied by the Chebyshev weight function and as we discussed in this section, Clenshaw-Curtis can only exactly integrate a polynomial of degree n. However, we see in Fig. 14 that this expectation is rather cynical and in this case Clenshaw-Curtis quadrature is almost as good Gauss-Legendre. This is something which has been known for quite some time. Even though Gauss-Legendre can exactly integrate polynomials of degree twice as large as what can be exactly integrated with Clenshaw-Curtis, their rates of convergence are approximately the same for non-analytic integrands. We will not discuss this further, instead we refer the reader to Refs. [11, 35] for a more in depth analysis.

In Section 8 we will use a Chebyshev pseudo-spectral method to solve the Schrödinger equation. Because we want to implement the Dirichlet boundary conditions that the wave function is zero on the boundary, we use the Chebyshev points of the second kind as our collocation grid. The anticipated behaviour of the wave functions, i.e., that they die off as they reach the boundary, then leads us naturally to use Clenshaw-Curtis quadrature to normalise the wave functions found.

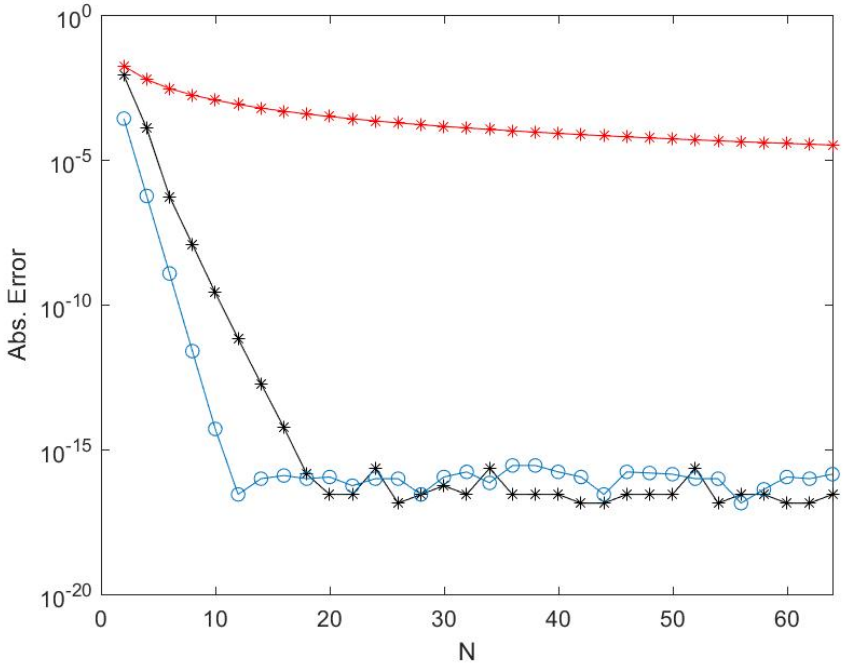


Figure 14: Comparison of error in calculating the example given in Eq. 120 using Clenshaw-Curtis (black asterisks), Gauss-Legendre (blue circles) and Gauss-Chebyshev (red asterisks) quadrature. This plot is produced by using the MATLAB® script `Ex_Script_2_Numerical_Quadrature.m`.

7 Fourier pseudo-spectral method

A continuous Fourier transform of a function $u(x)$, $x \in \mathbb{R}$ is defined[3] as

$$\hat{u}(k) = \int_{-\infty}^{\infty} u(x) e^{-ikx} dx \quad , \quad k \in \mathbb{R} \tag{163}$$

and the function can be reconstructed by using the the inverse Fourier transform

$$u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}(k) e^{ikx} dk \quad , \quad x \in \mathbb{R} \tag{164}$$

where x is the physical variable or coordinate and k is the Fourier variable or wavenumber.

If we restrict ourselves to considering only 2π periodic functions, i.e, $u(x) = u(x + 2\pi)$, then we only need to consider the function over the bounded spatial interval $(0, 2\pi]$. Moreover, the assumption of 2π periodicity not only means the spatial domain is bounded, but also that the Fourier space is discrete. This is because the spectral basis functions e^{ikx} are only periodic if k is an integer. Making the further restriction of considering the function only on an evenly spaced N-point grid

$$x_j = jh \quad , \quad h = \frac{2\pi}{N} \ , \tag{165}$$

where $j = 1, 2, \ldots, N$ and $N \in \mathbb{Z}^+$, then leads to the Fourier space being bounded as well. This also follows from the spectral basis functions as

$$e^{i(k+N)x} = e^{ik + iNj\frac{2\pi}{N}} = e^{ik + ij2\pi} = e^{ikx} \ . \tag{166}$$

This means we can restrict ourselves to $k \in [-\frac{\pi}{h}, \frac{\pi}{h}]$. The configuration and Fourier spaces are now both discrete and bounded. If N is even then $\frac{\pi}{h} = \frac{N}{2}$ is an integer and if N is odd then $\frac{N-1}{2}$ is an integer. Accordingly, if we then consider a discrete Fourier basis that includes the first N basis functions e^{ikx}, where the wavenumbers are

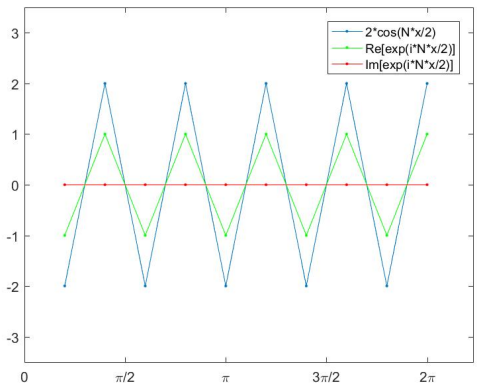
$$k = \begin{cases} -\frac{(N-1)}{2}, \ldots, \frac{N-1}{2} & , \quad N \text{ odd}, \\ \\ -\frac{N}{2} + 1, \ldots, \frac{N}{2} & , \quad N \text{ even}. \end{cases} \tag{167}$$

then the Discrete Fourier transform (DFT) can be defined as

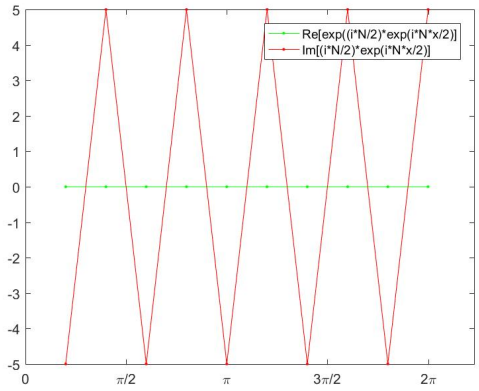
$$\tilde{u}_k = h \sum_{j=1}^{N} u(x_j) e^{-ikx_j} \ , \tag{168}$$

which is essentially the rectangular rule applied to Eq. 163. Note that we denote these coefficients with an overhead tilde rather than a hat, as they are not the exact coefficients in Eq. 163. They are an approximation to them. The Inverse Discrete Fourier transform (IDFT)

[3] We use the convention for the Fourier transform pair as used in Ref. [10]. Please note that this is different from the convention used in Refs. [12] and [15].



(i) The function $u(x) = 2\cos\frac{x}{2}$ and its approximation on the grid.



(ii) The approximation of derivative of the function $u(x) = 2\cos\frac{x}{2}$ on the grid.

Figure 15: Approximation of the function $u(x) = 2\cos\frac{x}{2}$ on the grid as described in the main text. These plots are produced using the MATLAB® script `Ex_Script_4_DFT_Issue.m`.

can then be defined as

$$u(x_j) = \begin{cases} \frac{1}{2\pi} \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \tilde{u}_k e^{ikx_j} & , \quad N \text{ even} \\ \\ \frac{1}{2\pi} \sum_{k=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} \tilde{u}_k e^{ikx_j} & , \quad N \text{ odd} \end{cases} \tag{169}$$

where $j = 1, 2, \ldots, N$.

Note that there is a problem when N is even. In the summation over k the limits are not symmetric, which means that $e^{i\frac{N}{2}x}$ is included but $e^{-i\frac{N}{2}x}$ is not. If we try to approximate the function

$$2\cos\left(\frac{N}{2}x\right) = e^{i\frac{N}{2}x} + e^{-i\frac{N}{2}x} \tag{170}$$

it will be incorrectly approximated as

$$e^{i\frac{N}{2}x} \ . \tag{171}$$

However, there is no such issue if N is odd as the range of k is symmetric. When our example is evaluated on the grid x_j, $j = 1, \ldots N$ it is the sawtooth wave shown in Fig. 15i. Its derivative on this grid is zero

$$\frac{d}{dx}\left(2\cos\left(\frac{N}{2}x\right)\right)\Big|_{x=x_j} = N\sin\left(\frac{N}{2}x_j\right) = N\sin j\pi = 0 \ . \tag{172}$$

However, its derivative will only be approximated by

$$i\frac{N}{2}e^{i\frac{N}{2}x} \tag{173}$$

instead of zero on the grid points. This situation is illustrated in Fig. 15i and Fig. 15ii. As explained in Ch. 3 of Ref. [10], this problem can be fixed by defining $\tilde{u}_{-\frac{N}{2}} = \tilde{u}_{\frac{N}{2}}$ and replacing the above definition for even N by

$$u(x_j) = \frac{1}{2\pi} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}\,\prime\prime} \tilde{u}_k e^{ikx_j} \quad , \quad j = 1, 2, \ldots, N \tag{174}$$

where the double prime once again indicates that the first and last terms are halved.

Given a grid x_j where $j = 1, 2, \ldots, N$ and a set of function evaluations $u(x_j)$ on this grid, we can approximate the function's derivative on this grid by interpolating the function globally, then evaluating the derivative of the interpolant on the grid. The first thing we need to do is construct an interpolant. This can be done in either configuration space or Fourier space. We are free to work in either space, but at times it can be simpler to work in one space rather than the other. Nonetheless, they are related to each other by matrix transformations and can be transitioned between efficiently by using the Fast Fourier Transform (FFT) and its inverse.

The above issue concerning the definition of the IDFT does not mean that Eq. 169 cannot be used to define the inverse. We use Eq. 174 for the purpose of deriving a band-limited interpolant in configuration space of the form

$$I_N u(x) = \sum_{j=1}^{N} C_j(x) u(x_j) \tag{175}$$

where $x \in \mathbb{R}$ and the $C_j(x)$ are the analogues of the Lagrange basis polynomials $\ell_j(x)$ referred to hereafter as the cardinal functions or collectively as the cardinal basis.

Alternatively, if the coefficients $\tilde{u}_k$ are known through Eq. 168, then using the IDFT we can define an interpolant in Fourier space as

$$I_N u(x) = \begin{cases} \frac{1}{2\pi} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}\,\prime\prime} \tilde{u}_k e^{ikx} & , \quad N \text{ even} \\ \frac{1}{2\pi} \sum_{k=-\frac{(N-1)}{2}}^{\frac{(N-1)}{2}} \tilde{u}_k e^{ikx} & , \quad N \text{ odd} \end{cases} \tag{176}$$

where $x \in \mathbb{R}$. The derivative $u(x)$ is then approximated as the derivative of interpolant of $u(x)$ on the grid in either space. We have chosen to work configuration space for this report. The band-limited interpolant in configuration space is derived in Theorem 7.1 below. The derivative on the grid is then approximated as

$$u'(x_i) \simeq (I_N u(x_i))' = \sum_{j=1}^{N} u(x_i) C_j'(x)|_{x=x_i} \ . \tag{177}$$

Obviously, this can be formulated in terms of matrices. The first derivative of the function can then be accomplished using a differentiation matrix defined as

$$(D_N^{(1)})_{ij} = C_j'(x)|_{x=x_i} \ . \tag{178}$$

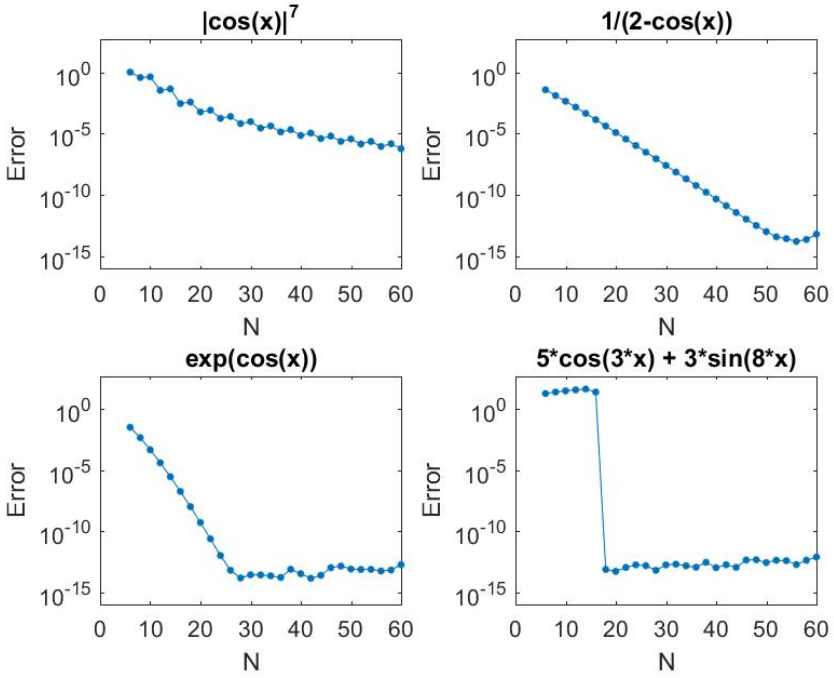


Figure 16: The error in the spectral derivative of four functions with varying levels of smoothness plotted as a function of N. These plots are produced using the MATLAB® script `Ex_Script_5_Fourier_Periodic_Diff.m`.

The matrix multiplication of the differentiation matrix with a vector of function evaluations will then return a vector containing the spectral approximation of the functions derivative on the grid. Differentiation matrices for performing higher derivatives on the grid can be similarly defined. The first and second order differentiation matrices are given in Theorem 7.2 below.

The extension to more general linear differential operators is straightforward. Consider the following eigenvalue problem

$$Lu_n = \lambda_n u \ . \tag{179}$$

The matrix representation of the linear differential operator L is simply

$$(L_N)_{ij} = (LC_j(x))|_{x=x_i} \quad , \quad i,j = 1,2,\ldots,N \ . \tag{180}$$

The eigenvalue problem can then be reduced to the following matrix eigenvalue problem

$$L_N \mathbf{u} = \overline{\lambda}_n \mathbf{u} \quad , \quad \mathbf{u} = (u_n(x_1),\ u_n(x_2),\ \ldots, u_n(x_N))^\top \quad , \quad \overline{\lambda}_n \simeq \lambda_n \ . \tag{181}$$

Standard linear algebra routines can then be used to solve this linear system, e.g., `eig()` in Matlab. Our eigenvalue problem, i.e., the Time Independent Schrödinger Equations (TISE) can be solved in this way.

Accuracy of spectral differentiation using these differentiation matrices is impressive for smooth functions, but can be greatly diminished for less regular functions. Every extra continuous derivative a function possesses, the more accurate the spectral approach to differentiation will

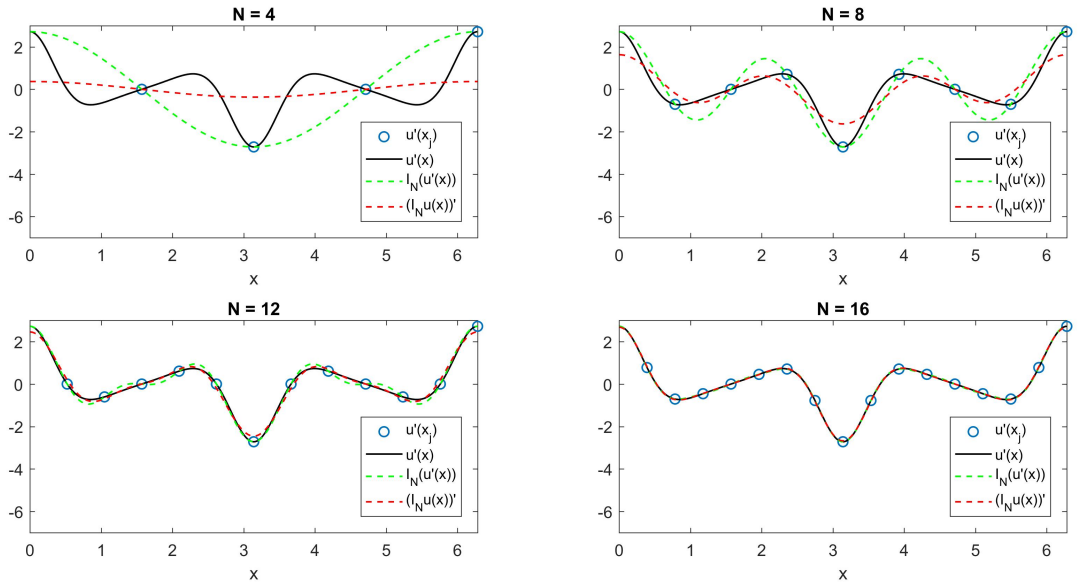


Figure 17: The derivative of the function $u(x) = e^{\cos 2x} \sin x$, the derivative the interpolant $(I_N u(x))'$ and interpolant of the derivative $I_N(u'(x))$. The circles denote the collocation points. These plots are produced using the MATLAB® script `Ex_Script_6_Fourier_Interp.m`.

tend to be. In the way of a few examples, in Fig. 16, we consider four functions with varying levels of smoothness.

It is important to note that the derivative of the interpolant is not equal to the interpolant of the derivative, i.e.,

$$(I_N u(x))' \neq I_N \left(u'(x)\right) \ , \tag{182}$$

however as $N \to \infty$ one cannot tell the difference, see Fig. 17 which illustrates this fact for the function

$$u(x) = e^{\cos 2x} \sin x \ . \tag{183}$$

For comparison, in Fig. 18, we include a calculation of the maximum error in interpolating the function, i.e.,

$$\max_{x \in (0,2\pi]} |I_N u(x) - u(x)| \ . \tag{184}$$

Note that we used the chebfun function `trigBary()` to perform the interpolation which utilises the second form of the barycentric formula.

To derive the the form of the cardinal basis functions appearing in the band-limited interpolant for a periodic function we need Proposition 7.1.

Proposition 7.1. *The following geometrically increasing series is a sum which can be expressed in closed form as*

$$\sum_{k=-M}^{M} r^k = \frac{r^{-M} - r^{M+1}}{1 - r} \ . \tag{185}$$

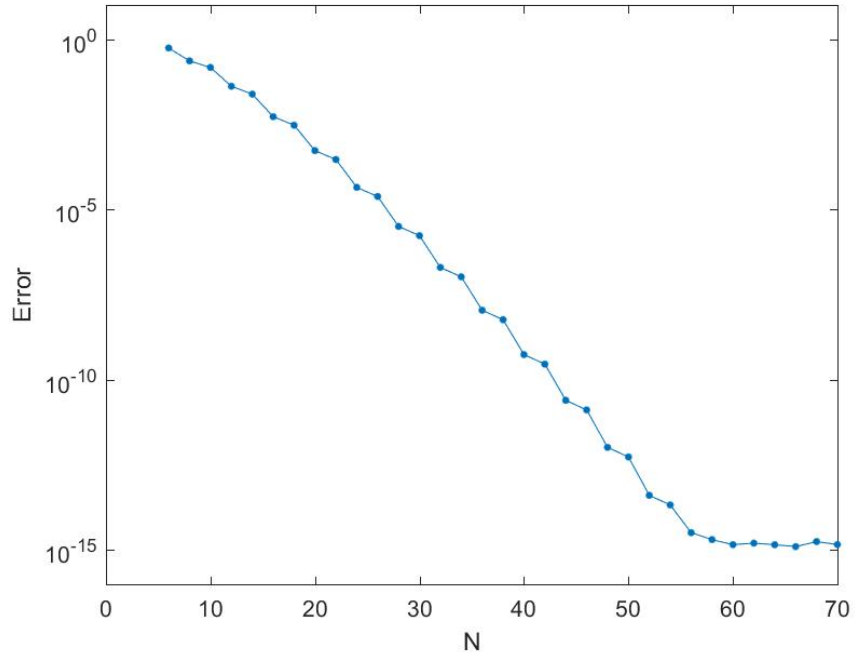


Figure 18: The maximum error of the interpolant of the function $u(x) = e^{cos(2x)} \sin(x)$ *as a function of the number of grid points* N*. This plot is produced using the MATLAB® script* `Ex_Script_7_Fourier_Interp_Error.m`.

Proof: The proof is just a straightforward variant of summing the usual geometric series where $k = -M, -M+1, \ldots, M-1, M$ rather than $k = 0, 1, 2, \ldots, M$. Consider

$$\begin{aligned}(1-r)\sum_{k=-M}^{M} r^k &= (1-r)(r^{-M} + r^{-M+1} + \ldots + r^{M-1} + r^M) \\ &= r^{-M} + r^{-M+1} + \ldots + r^{M-1} + r^M \\ &\qquad - r^{-M+1} - r^{-M+2} - \ldots - r^M - r^{M+1} \\ &= r^{-M} - r^{M+1} \ . \end{aligned} \tag{186}$$

If $r \neq 1$, then

$$\sum_{k=-M}^{M} r^k = \frac{r^{-M} - r^{M+1}}{1-r} \ . \tag{187}$$

■

Theorem 7.1. *(Band-limited Interpolant) The band-limited interpolant for a* 2π *periodic function* $u(x)$ *is given by*

$$I_N u(x) = \sum_{j=1}^{N} C_j(x) u(x_j) \ , \tag{188}$$

where $x_j = jh$, $h = \frac{2\pi}{N}$ *and the cardinal basis functions are*

$$C_j(x) = \begin{cases} \frac{1}{N}\sin\frac{N}{2}(x-x_j)\cot\frac{1}{2}(x-x_j) & , \quad N \quad even, \\ \\ \frac{1}{N}\sin\frac{N}{2}(x-x_j)\csc\frac{1}{2}(x-x_j) & , \quad N \ odd. \end{cases} . \tag{189}$$

Proof: First of all we define the periodic delta function and find its band-limited interpolant, which will be different for even and odd N. The periodic delta function is defined as

$$\delta_j = \delta(x_j) = \begin{cases} 1 & , \quad j \equiv 0 (\mathrm{mod} N) \\ 0 & , \quad \text{otherwise} \end{cases} . \tag{190}$$

The discrete Fourier transform of the periodic delta function is simply a constant,

$$\tilde{\delta}_k = h\sum_{j=1}^{N} e^{-ikx_j}\delta_j = h \ . \tag{191}$$

Consider the case of even N first, then band-limited interpolant of the periodic delta function is

$$\begin{aligned} p(x) &= \frac{1}{2\pi}{\sum_{k=-N/2}^{N/2}}'' e^{ikx}\tilde{\delta}_k \\ &= \frac{h}{2\pi}{\sum_{k=-N/2}^{N/2}}'' e^{ikx} \\ &= \frac{h}{2\pi}\left(\frac{1}{2}\sum_{k=-N/2}^{N/2-1} e^{ikx} + \frac{1}{2}\sum_{k=-N/2+1}^{N/2} e^{ikx}\right) \\ &= \frac{h}{2\pi}\frac{1}{2}\left(\sum_{k=-N/2}^{N/2-1} e^{ikx} + \sum_{k=-N/2+1}^{N/2} e^{ikx}\right) \\ &= \frac{h}{2\pi}\frac{1}{2}\left(e^{-ix/2}\sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} + e^{ix/2}\sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx}\right) \\ &= \frac{h}{2\pi}\frac{1}{2}(e^{-ix/2}+e^{ix/2})\sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} \\ &= \frac{h}{2\pi}\cos\frac{x}{2}\sum_{k=-N/2+1/2}^{N/2-1/2} e^{ikx} \ . \end{aligned} \tag{192}$$

Let $M = \frac{N}{2} - \frac{1}{2}$ and $r = e^{ix}$, then

$$p(x) = \frac{h}{2\pi}\cos\frac{x}{2}\sum_{k=-M}^{M}(e^{ix})^k = \frac{h}{2\pi}\cos\frac{x}{2}\sum_{k=-M}^{M} r^k \ . \tag{193}$$

Using Prop. 7.1 we obtain

$$\begin{aligned}
p(x) &= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{r^{-M}-r^{M+1}}{1-r} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{e^{-iMx}-e^{i(M+1)x}}{1-e^{ix}} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{e^{-i(N/2-1/2)x}-e^{i(N/2-1/2+1)x}}{1-e^{ix}} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{e^{i(-N/2+1/2)x}-e^{i(N/2+1/2)x}}{1-e^{ix}} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)e^{ix/2}\frac{e^{-iNx/2}-e^{iNx/2}}{1-e^{ix}} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{e^{-iNx/2}-e^{iNx/2}}{e^{-ix/2}-e^{ix/2}} \\
&= \frac{h}{2\pi}\cos\left(\frac{x}{2}\right)\frac{\sin\left(\frac{Nx}{2}\right)}{\sin\left(\frac{x}{2}\right)} \\
&= \frac{h}{2\pi}\sin\left(\frac{Nx}{2}\right)\cot\left(\frac{x}{2}\right) \\
&= \frac{1}{N}\sin\left(\frac{Nx}{2}\right)\cot\left(\frac{x}{2}\right) \ . \qquad (194)
\end{aligned}$$

Since a periodic grid function can be expanded in the basis of shifted periodic delta functions by

$$u(x_j) = \sum_{i=1}^{N} u(x_i)\delta_{j-i} \ , \qquad (195)$$

then for even N the band-limited interpolant to the function $u(x)$ is

$$I_N u(x) = \sum_{j=1}^{N} C_j(x)u(x_j) \qquad (196)$$

where the cardinal function is given by

$$C_j(x) = \frac{1}{N}\sin\frac{N}{2}(x-x_j)\cot\frac{1}{2}(x-x_j) \ . \qquad (197)$$

For odd N the calculation follows in a similar manner, except the N point interpolant for the periodic delta function is given by

$$p(x) = \frac{1}{2\pi}\sum_{k=-(N-1)/2}^{(N-1)/2} e^{ikx}\tilde{\delta}_k \ . \qquad (198)$$

Following the same procedure as for even N. Let $M = \frac{N-1}{2}$ and $r = e^{ix}$, then we obtain

$$\begin{aligned}
p(x) &= \frac{h}{2\pi} \sum_{k=-(N-1)/2}^{(N-1)/2} e^{ikx} \\
&= \frac{h}{2\pi} \sum_{k=-M}^{M} r^k \\
&= \frac{h}{2\pi} \frac{r^{-M} - r^{M+1}}{1-r} \\
&= \frac{h}{2\pi} \frac{e^{-iMx} - e^{i(M+1)x}}{1-e^{ix}} \\
&= \frac{h}{2\pi} \frac{e^{-i\frac{(N-1)}{2}x} - e^{i(\frac{N-1}{2}+1)x}}{1-e^{ix}} \\
&= \frac{h}{2\pi} e^{ix/2} \frac{e^{-Nx/2} - e^{iNx/2}}{1-e^{ix}} \\
&= \frac{h}{2\pi} \frac{e^{-Nx/2} - e^{iNx/2}}{e^{-ix/2} - e^{ix/2}} \\
&= \frac{h}{2\pi} \frac{\sin\left(\frac{Nx}{2}\right)}{\sin\left(\frac{x}{2}\right)} \\
&= \frac{h}{2\pi} \sin\left(\frac{Nx}{2}\right) \csc\left(\frac{x}{2}\right) \\
&= \frac{1}{N} \sin\left(\frac{Nx}{2}\right) \csc\left(\frac{x}{2}\right) \ . \qquad (199)
\end{aligned}$$

Thus the cardinal function for odd N is given by

$$C_j(x) = \frac{1}{N} \sin\frac{N}{2}(x - x_j) \csc\frac{1}{2}(x - x_j) \ . \qquad (200)$$

■

The Fourier differentiation matrices are given in a number of works, see for example Refs. [12, 14, 15, 26]. Note however, there is a typo in Eq. 27 of Ref. [26] for the second order differentiation matrix for odd N.

Theorem 7.2. *(Fourier Differentiation Matrices) The first and second order differentiation matrices for even N are*

$$(D_N^{(1)})_{ij} = \begin{cases} 0 & , \quad i = j \\ \frac{1}{2}(-1)^{i-j} \cot\frac{(i-j)h}{2} & , \quad i \neq j \end{cases} , \qquad (201)$$

and

$$(D_N^{(2)})_{ij} = \begin{cases} -\frac{\pi^2}{3h^2} - \frac{1}{6} & , \quad i = j \\ \frac{1}{2}(-1)^{i-j+1} \csc^2\frac{(i-j)h}{2} & , \quad i \neq j \end{cases} . \qquad (202)$$

Likewise for odd N we have

$$(D_N^{(1)})_{ij} = \begin{cases} 0 & , \quad i = j \\ \frac{1}{2}(-1)^{i-j} \csc\frac{(i-j)h}{2} & , \quad i \neq j \end{cases} \qquad (203)$$

and

$$(D_N^{(2)})_{ij} = \begin{cases} -\frac{\pi^2}{3h^2} + \frac{1}{12} & , \quad i = j \\ \frac{1}{2}(-1)^{i-j+1} \csc\frac{(i-j)h}{2} \cot\frac{(i-j)h}{2} & , \quad i \neq j \end{cases} , \tag{204}$$

where $i, j = 1, \ldots, N$.

Proof: The first and second order differentiation matrices are defined as

$$(D_N^{(1)})_{ij} = C_j'(x_i) \quad \text{and} \quad (D_N^{(2)})_{ij} = C_j''(x_i) \ . \tag{205}$$

In what follows we make repeated use of l'Hospitals rule and of the following:

$$\begin{aligned} \sin\left(\frac{N(x_i - x_j)}{2}\right) &= \sin\left(\frac{N(i-j)h}{2}\right) \\ &= \sin\left(\frac{N(i-j)}{2}\frac{2\pi}{N}\right) \\ &= \sin(i-j)\pi \\ &= 0 \end{aligned} \tag{206}$$

and

$$\begin{aligned} \cos\left(\frac{N(x_i - x_j)}{2}\right) &= \cos\left(\frac{N(i-j)h}{2}\right) \\ &= \cos\left(\frac{N(i-j)}{2}\frac{2\pi}{N}\right) \\ &= \cos(i-j)\pi \\ &= (-1)^{i-j} \ , \end{aligned} \tag{207}$$

where $i, j \in \mathbb{Z}$.

Even N, first order differentiation matrix:

- For $i = j$, we begin by calculating the derivative of the cardinal function for even N given in Eq. 189,

$$\begin{aligned} C_j'(x) &= \frac{d}{dx}\left(\frac{1}{N}\sin\frac{N}{2}(x - x_j)\cot\frac{1}{2}(x - x_j)\right) \\ &= \frac{1}{2}\cot\left(\frac{x - x_j}{2}\right)\cos\left(\frac{N(x - x_j)}{2}\right) - \frac{\csc^2\left(\frac{x-x_j}{2}\right)\sin\left(\frac{N(x-x_j)}{2}\right)}{2N} \\ &= \frac{\csc^2\left(\frac{x-x_j}{2}\right)\left(N\sin(x - x_j)\cos\left(\frac{N(x-x_j)}{2}\right) - 2\sin\left(\frac{N(x-x_j)}{2}\right)\right)}{4N} \ . \end{aligned} \tag{208}$$

Naively evaluating the derivative of the cardinal function at x_i will give an indeterminate

expression. To evaluate it we must use l'Hospital's rule twice because of $csc^2(0)$, i.e.,

$$\begin{aligned} C_j'(x_i) &= \lim_{x\to x_i} C_j'(x) \\ &= \frac{1}{4N}\lim_{x\to x_i}\frac{\frac{d^2}{dx^2}\left(N\sin(x-x_j)\cos\left(\frac{N(x-x_j)}{2}\right)-2\sin\left(\frac{N(x-x_j)}{2}\right)\right)}{\frac{d^2}{dx^2}\left(\sin^2\left(\frac{x-x_j}{2}\right)\right)} \\ &= \frac{1}{4N}\lim_{x\to x_i}\frac{1}{16}\left(2n(1-2\cos(x-x_j))\sin\left(\frac{1}{2}n(x-x_j)\right)\right. \\ &\qquad \left.-\left(n^2+4\right)\sin(x-x_j)\cos\left(\frac{1}{2}n(x-x_j)\right)\right) \\ &\qquad /(\frac{1}{2}\cos(x-x_j) \\ &= 0\ . \end{aligned} \tag{209}$$

- For $i \neq j$, we note that in Eq. 208 the second term in the brackets is zero, so

$$\begin{aligned} C_j'(x_i) &= \frac{1}{2}\cot\left(\frac{x_i-x_j}{2}\right)\cos\left(\frac{N(x_i-x_j)}{2}\right) \\ &= \frac{1}{2}\cot\left(\frac{(i-j)h}{2}\right)\cos\left(\frac{N(i-j)}{2}\frac{2\pi}{N}\right) \\ &= \frac{1}{2}\cot\left(\frac{(i-j)h}{2}\right)\cos\left((i-j)\pi\right) \\ &= \frac{1}{2}(-1)^{i-j}\cot\left(\frac{(i-j)h}{2}\right)\ . \end{aligned} \tag{210}$$

For even N, the second order differentiation matrix:

- For $i = j$, we begin by calculating the second derivative of the cardinal function as given in Eq. 189

$$\begin{aligned} C_j''(x) &= -\frac{1}{2}\cos\frac{N(x-x_j)}{2}\csc^2\frac{(x-x_j)}{2} - \frac{N}{4}\cot\frac{(x-x_j)}{2}\sin\frac{N(x-x_j)}{2} \\ &\qquad + \frac{1}{2N}\cot\frac{(x-x_j)}{2}\csc^2\frac{(x-x_j)}{2}\sin\frac{N(x-x_j)}{2} \end{aligned} \tag{211}$$

$$\begin{aligned} &= \frac{1}{32N}\csc^3\frac{(x-x_j)}{2}\left(\left(-8-8N+N^2\right)\sin\left(\frac{(x-x_j)(1-N)}{2}\right)\right. \\ &\quad -N^2\sin\frac{(x-x_j)(3-N)}{2} + \left(8-8N-N^2\right)\sin\frac{(x-x_j)(1+N)}{2} \\ &\qquad \left.+N^2\sin\frac{(x-x_j)(3+N)}{2}\right)\ . \end{aligned} \tag{212}$$

Because of the $\csc^3(0)$ we use l'Hospital's rule three times. Thus, the limit of the third

derivative of the numerator divided by the third derivative of the denominator is

$$\begin{aligned} C_j''(x_j) &= \lim_{x\to x_j} \frac{1}{32N}\left(-\left(N^2-8N-8\right)\left(\frac{1-N}{2}\right)^3 \cos\left(\frac{(1-N)(x-x_j)}{2}\right)\right. \\ &\quad +\left(\frac{3-N}{2}\right)^3 N^2\cos\left(\frac{(3-N)(x-x_j)}{2}\right) \\ &\quad -\left(\frac{N+1}{2}\right)^3\left(-N^2-8N+8\right)\cos\left(\frac{(N+1)(x-x_j)}{2}\right) \\ &\quad \left.-\left(\frac{N+3}{2}\right)^3 N^2\cos\left(\frac{(N+3)(x-x_j)}{2}\right)\right) \\ &\quad /\left(\frac{3}{4}\cos^3\left(\frac{x-x_j}{2}\right)-\frac{21}{8}\cos\left(\frac{x-x_j}{2}\right)\sin^2\left(\frac{x-x_j}{2}\right)\right) \\ &= \frac{1/16(-2-N^2)}{3/4} \\ &= -\frac{1}{12}(2+N^2) \\ &= -\frac{\pi^2}{3h^2}-\frac{1}{6}\,. \end{aligned} \tag{213}$$

- For $i\neq j$, first note that the second and third terms in Eq. 211 are zero, so

$$\begin{aligned} C_j''(x_i) &= -\frac{1}{2}\cos\frac{N(x-x_j)}{2}\csc^2\frac{(x-x_j)}{2} \\ &= -\frac{1}{2}\cos\frac{N(i-j)}{2}\frac{2\pi}{N}\csc^2\frac{(i-j)h}{2} \\ &= -\frac{1}{2}\cos(i-j)\pi\csc^2\frac{(i-j)h}{2} \\ &= \frac{1}{2}(-1)^{i-j+1}\csc^2\frac{(i-j)h}{2}\,. \end{aligned} \tag{214}$$

For odd N, the first order differentiation matrix:

- For $i=j$, we begin by calculating the first derivative of the cardinal function given in Eq. 189

$$\begin{aligned} C_j'(x) &= \frac{1}{2}\cos\left(\frac{N(x-x_j)}{2}\right)\csc\left(\frac{x-x_j}{2}\right) \\ &\quad -\frac{1}{2N}\left(\cot\left(\frac{x-x_j}{2}\right)\csc\left(\frac{x-x_j}{2}\right)\sin\left(\frac{N(x-x_j)}{2}\right)\right) \qquad (215) \\ &= \frac{1}{4N}\csc^2\left(\frac{x-x_j}{2}\right)\left((1+N)\sin\left(\frac{(1-N)(x-x_j)}{2}\right)\right. \\ &\quad \left.+(-1+N)\sin\left(\frac{(1+N)(x-x_j)}{2}\right)\right)\,. \qquad (216) \end{aligned}$$

Naively taking the limit to $x\to x_j$ leads to an indeterminate expression again because of the $csc^2(0)$ out front of Eq. 216, so we must apply l'Hospital's rule again twice. The limit

of the ratio of the second derivative of the numerator divided by the second derivative of the denominator then gives

$$\begin{aligned} C'_j(x_j) &= \lim_{x\to x_j}\left(\frac{1}{4N}\left(-\left(\frac{1}{2}-\frac{N}{2}\right)^2(1+N)\sin\left(\frac{(1-N)(x-x_j)}{2}\right)\right.\right. \\ &\qquad \left.\left.-\left(\frac{1}{2}+\frac{N}{2}\right)^2(-1+N)\sin\left(\frac{(1+N)(x-x_j)}{2}\right)\right)\right) \\ &\qquad /\left(\frac{1}{2}\cos^2\left(\frac{x-x_j}{2}\right)-\frac{1}{2}\sin^2\left(\frac{x-x_j}{2}\right)\right) \\ &= 0\,. \end{aligned} \tag{217}$$

- For $i \neq j$, from Eq. 215 we have that

$$\begin{aligned} C'_j(x_i) &= \frac{\frac{1}{2}\cos\left(\frac{N(x_i-x_j)}{2}\right)}{\sin\left(\frac{x_i-x_j}{2}\right)}-\frac{1}{2N}\frac{\sin\left(\frac{N(x_i-x_j)}{2}\right)\cos\left(\frac{x_i-x_j}{2}\right)}{\sin^2\left(\frac{x_i-x_j}{2}\right)} \\ &= \frac{\frac{1}{2}\cos\left(\frac{N(i-j)h}{2}\right)}{\sin\left(\frac{(i-j)h}{2}\right)}-\frac{1}{2N}\frac{\sin\left(\frac{N(i-j)h}{2}\right)\cos\left(\frac{(i-j)h}{2}\right)}{\sin^2\left(\frac{(i-j)h}{2}\right)} \\ &= \frac{\frac{1}{2}\cos\left(\frac{N(i-j)}{2}\frac{2\pi}{N}\right)}{\sin\left(\frac{(i-j)h}{2}\right)}-\frac{1}{2N}\frac{\sin\left(\frac{N(i-j)}{2}\frac{2\pi}{N}\right)\cos\left(\frac{(i-j)h}{2}\right)}{\sin^2\left(\frac{(i-j)h}{2}\right)} \\ &= \frac{\frac{1}{2}\cos\left((i-j)\pi\right)}{\sin\left(\frac{(i-j)h}{2}\right)}-\frac{1}{2N}\frac{\sin\left((i-j)\pi\right)\cos\left(\frac{(i-j)h}{2}\right)}{\sin^2\left(\frac{(i-j)h}{2}\right)} \\ &= \frac{\frac{1}{2}(-1)^{(i-j)}}{\sin\left(\frac{(i-j)h}{2}\right)}-0 \\ &= \frac{1}{2}(-1)^{(i-j)}\csc\left(\frac{(i-j)h}{2}\right)\,. \end{aligned} \tag{218}$$

For odd N, the second order differentiation matrix:

- For $i = j$, we begin by calculating the second derivative of the cardinal function given

in Eq. 189

$$
\begin{aligned}
C_j''(x) &= -\frac{1}{2}\cos\left(\frac{N(x-x_j)}{2}\right)\cot\left(\frac{x-x_j}{2}\right)\csc\left(\frac{x-x_j}{2}\right) \\
&\quad -\frac{N}{4}\csc\left(\frac{x-x_j}{2}\right)\sin\left(\frac{N(x-x_j)}{2}\right) \\
&\quad +\frac{\sin\left(\frac{N(x-x_j)}{2}\right)}{4N}\left(\cot^2\left(\frac{x-x_j}{2}\right)\csc\left(\frac{x-x_j}{2}\right)+\csc^3\left(\frac{x-x_j}{2}\right)\right) \qquad (219)\\
&= \frac{1}{16N}\csc^3\left(\frac{x-x_j}{2}\right)\left((6-2N^2)\sin\left(\frac{N(x-x_j)}{2}\right)\right. \\
&\quad +(-1-2N-N^2)\sin\left((1-\frac{N}{2})(x-x_j)\right) \\
&\quad \left.+(1-2N+N^2)\sin\left((1+\frac{N}{2})(x-x_j)\right)\right)\ . \qquad (220)
\end{aligned}
$$

We apply l'Hospital's rule three times to handle the $csc^3(0)$ term we have pulled out front in Eq. 220. The limit of the third derivative of the numerator divided by the third derivative of the denominator is then

$$
\begin{aligned}
C_j''(x_j) &= \lim_{x\to x_j}\frac{1}{16N}\left(-\frac{1}{8}N^3(6-2N^2)\cos\left(\frac{N(x-x_j)}{2}\right)\right. \\
&\quad -(1-\frac{N}{2})^3(-1-2N-N^2)\cos\left((1-\frac{N}{2})(x-x_j)\right) \\
&\quad \left.-(1+\frac{N}{2})^3(1-2N+N^2)\cos\left((1+\frac{N}{2})(x-x_j)\right)\right) \\
&\quad /\left(\frac{3}{4}\cos^3\left(\frac{x-x_j}{2}\right)-\frac{21}{8}\cos\left(\frac{x-x_j}{2}\right)\sin^2\left(\frac{x-x_j}{2}\right)\right) \\
&= \frac{1}{16N}\left(-\frac{1}{8}N^3(6-2N^2)\right. \\
&\quad -(1-\frac{N}{2})^3(-1-2N-N^2) \\
&\quad \left.-(1+\frac{N}{2})^3(1-2N+N^2)\right) \\
&\quad /\left(\frac{3}{4}\right) \\
&= \frac{\frac{1}{16}(1-N^2)}{\frac{3}{4}} \\
&= \frac{1}{12}(1-N^2) \\
&= \frac{1}{12}-\frac{\pi^2}{3h^2}\ . \qquad (221)
\end{aligned}
$$

- For $i \neq j$ we have from Eq. 219

$$\begin{aligned} C_j''(x_i) &= -\frac{1}{2}\cos\left(\frac{N(x_i-x_j)}{2}\right)\cot\left(\frac{x_i-x_j}{2}\right)\csc\left(\frac{x_i-x_j}{2}\right) \\ &\quad -\frac{N}{4}\csc\left(\frac{x_i-x_j}{2}\right)\sin\left(\frac{N(x_i-x_j)}{2}\right) \\ &\quad +\frac{\sin\left(\frac{N(x_i-x_j)}{2}\right)}{4N}\left(\cot^2\left(\frac{x_i-x_j}{2}\right)\csc\left(\frac{x_i-x_j}{2}\right)+\csc^3\left(\frac{x_i-x_j}{2}\right)\right) . \end{aligned} \tag{222}$$

  Using Eq. 206 and Eq. 207 we find

$$\begin{aligned} C_j''(x_i) &= \frac{1}{2}(-1)^{i-j+1}\cot\left(\frac{x_i-x_j}{2}\right)\csc\left(\frac{x_i-x_j}{2}\right) \\ &= \frac{1}{2}(-1)^{i-j+1}\cot\left(\frac{(i-j)h}{2}\right)\csc\left(\frac{(i-j)h}{2}\right) . \end{aligned} \tag{223}$$

■

To conclude this section we merely note the important property that the Fourier differentiation matrices satisfy [26]

$$D_N^{(\ell)} = (D_N^{(1)})^\ell \tag{224}$$

if N odd or if N even and ℓ odd, otherwise

$$D_N^{(\ell)} \neq (D_N^{(1)})^\ell . \tag{225}$$

7.1 Time-independent Schrödinger's equation examples

In this section we look at solving the TISE with two example potentials, the harmonic oscillator and the quartic or anharmonic oscillator potential. We do not consider the linear potential because of the similarity of the resulting equation with Airy's equation whose solutions are known to be exponentially decaying in one direction and oscillating in the other. The lack of periodicity of the wave functions will give rise to issues in trying to use the Fourier method.

Example 7.1. (Harmonic Oscillator)

Our first example is the TISE with the harmonic oscillator potential. The exact solution to the TISE with this potential is simply a Gaussian multiplied by a Hermite polynomial with the correct normalisation, see for example Ref. [19] and App. A.1. We will use these exact solutions to judge the accuracy of our numerical approach. The function `HOWaveFunction()` which calculates these wave functions is included in Govdex. This function uses Matlab's `hermiteH()` function to evaluate the Hermite polynomials.

As the solution is well localised, we can simply consider solving the Schrödinger equation in a large box of size $[-L, L]$ rather than on the infinite interval. If we take $L = 8$, then the Gaussian which multiplies the Hermite polynomials in the exact solution is approximately

$$e^{-L^2/2} \simeq 10^{-14} , \tag{226}$$

so this should be a large enough interval to begin our analysis.

To be specific, we will be solving the following differential equation

$$\left(-\frac{d^2}{dx^2}+x^2\right)\psi_n(x)=\tilde{\lambda}_n\psi_n(x) \quad , \tag{227}$$

where we have set the constants $\hbar = m = \omega = 1$ in Eq. A1 and we will impose the periodic boundary conditions $\psi_n(-L) = \psi_n(L)$. The exact eigenvalues are

$$\tilde{\lambda}_n = 2(n+1) \tag{228}$$

and the corresponding wave functions are

$$\psi_n(x)=\frac{1}{\sqrt{2^n n!\sqrt{\pi}}}e^{-x^2/2}H_n(x) \quad . \tag{229}$$

We will be using a Fourier pseudo-spectral method to solve the problem on the interval $[-L, L]$. This can be done by use of a simple domain map,

$$\xi = L\frac{(x-\pi)}{\pi} \ , \tag{230}$$

where $x \in [0, 2\pi]$ and $\xi \in [-L, L]$.

The first step is to construct the matrix representation of the differential operator on the evenly spaced N point grid and then solve for the eigensystem. The Fourier differentiation matrix on the equispaced grid is given by Thm. 7.2. The potential on this grid is simply given by the diagonal matrix

$$\text{diag}(\xi_1^2, \ \ldots \ , \ \xi_N^2) \ . \tag{231}$$

We find the eigensystem using Matlab's in built command `eig()`. The system then must be ordered according to ascending eigenvalues. Note that when solving for the eigensystem numerically, wave functions may appear with the opposite sign than expected because ψ_n and $-\psi_n$ are both valid solutions to the Schrödinger equation with the same eigenvalue.

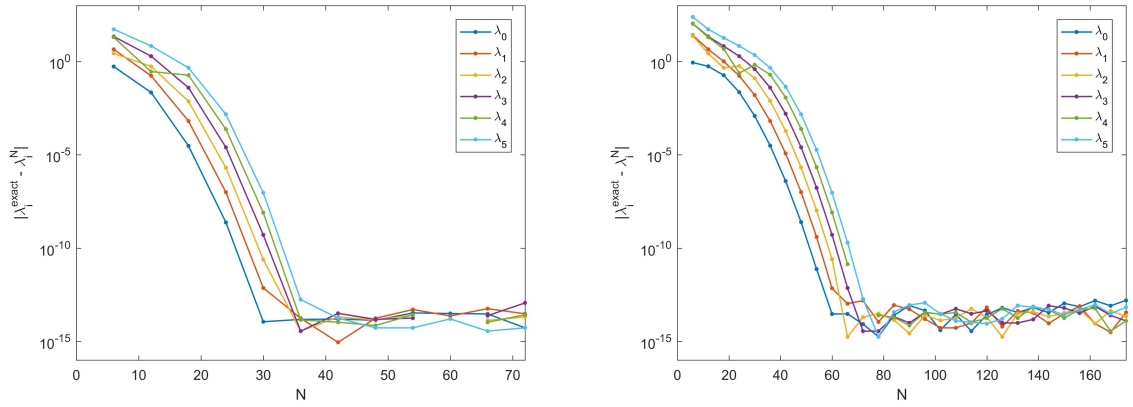
Each eigenvector $\mathbf{u}$ returned by `eig()` is normalised as a vector, that is $\mathbf{u}\cdot\mathbf{u} = 1$. These eigenvectors represent the wave functions on the grid and need to be normalised as a wave function should be, i.e.,

$$\int_{-L}^{L}|\psi_n(x)|^2dx = 1 \quad . \tag{232}$$

This requires an integration to determine the normalisation constant. This integral is done using numerical quadrature, specifically the rectangular rule because we are working on the equidistant grid.

We now have the eigenvalues and normalised eigenfunctions evaluated on the equidistant grid. This grid may not be as fine as desired, so we can then interpolate these eigenfunctions on a finer grid using trigonometric barycentric interpolation. In the example script `Ex_Script_11_Fourier_HO.m`, we have used Chebfun's `trigBary()` function to perform the interpolation, but it could of course been done some other way, e.g., by using the cardinal functions.

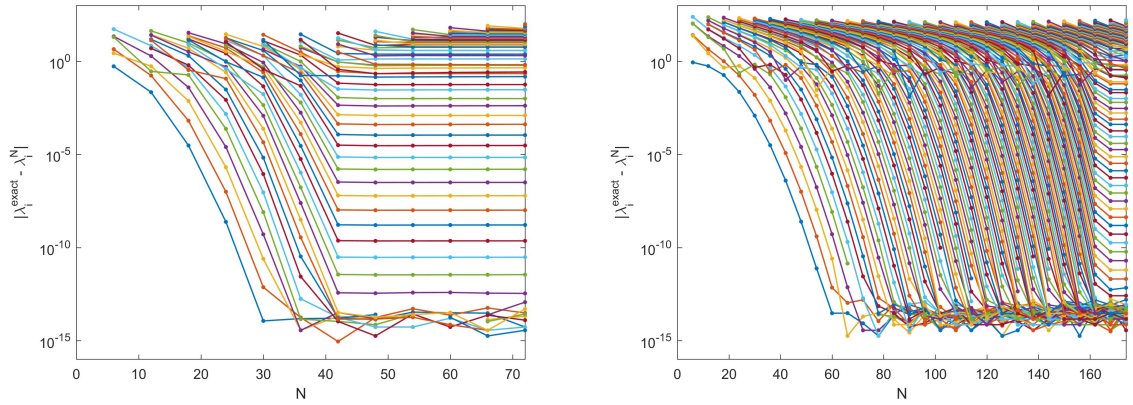
In Figs. 19i and 19ii we show the error in the first 6 eigenvalues for $L = 8$ and $L = 16$. The eigenvalues appear to become more accurately reproduced as N increases. However, as can



(i) Error in the first 6 eigenvalues for $L = 8$, $N = 72$.

(ii) Error in the first 6 eigenvalues for $L = 16$, $N = 174$.

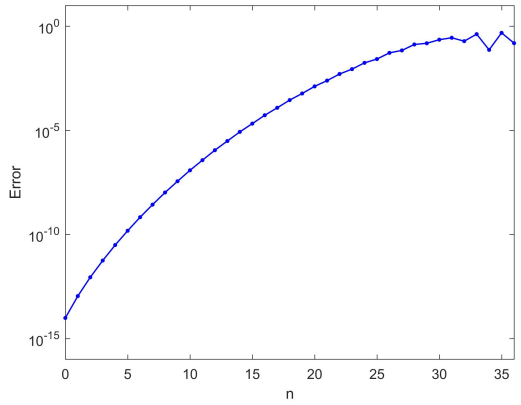
Figure 19: The error in the harmonic oscillator eigenvalues using the Fourier pseudo-spectral method. These plots are produced by modifying the MATLAB® script `Ex_Script_11_Fourier_HO.m`.



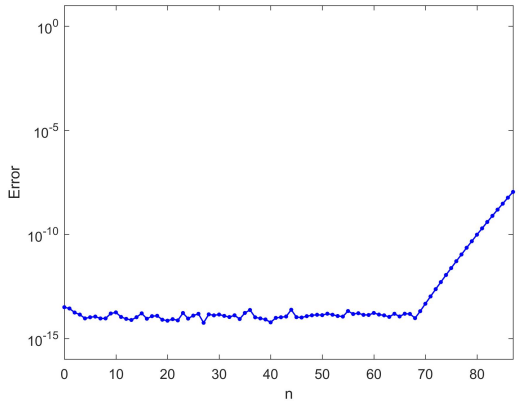
(i) Error in all eigenvalues for $L = 8$, $N = 72$.

(ii) Error in all eigenvalues for $L = 16$, $N = 174$.

Figure 20: The error in the harmonic oscillator eigenvalues. These plots are produced by modifying the MATLAB® script `Ex_Script_11_Fourier_HO.m`.



(i) Error in the harmonic oscillator wave functions for $L = 8$ and $N = 72$.



(ii) Error in the harmonic oscillator wave functions for $L = 16$ and $N = 174$.

Figure 21: Error in harmonic oscillator wave functions. These plots are produced by modifying the MATLAB® script `Ex_Script_11_Fourier_HO.m`.

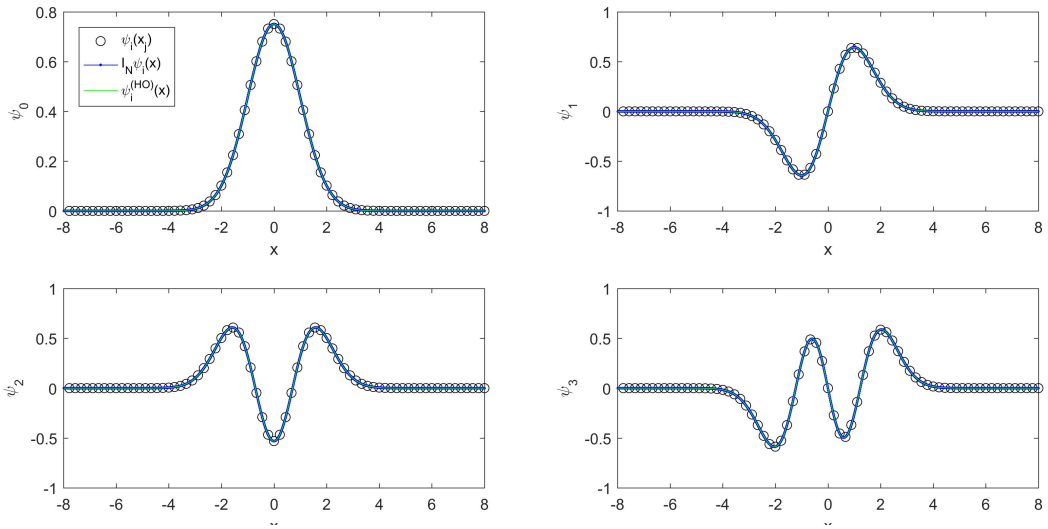


Figure 22: The first four harmonic oscillator wave functions calculated with $L = 8$ and $N = 72$. The case of $L = 16$ and $N = 174$ looks essentially indistinguishable. The circles mark the values at the collocation points, the blue curves are the interpolants calculated using Chebfun's `trigBary()` *function and the green curves are the corresponding exact harmonic oscillator wave functions. These plots are produced by modifying the MATLAB® script* `Ex_Script_11_Fourier_HO.m`.

be seen in Fig. 20i and 20ii, the eigenvalues corresponding to more highly excited states, i.e., wave functions with more nodes, are less accurately calculated. Also it is clear from these figures as we increase the box size, L, N must also be increased to achieve the same accuracy for a given eigenfunction. This is illustrated quite clearly by the first mode, for $L = 8$ the error drops to the round-off plateau at $N = 30$ whereas for $L = 16$ this does not occur until $N = 60$. If we increase L holding N fixed, the N collocation nodes are then distributed in a larger region. Since the wave functions are localised about the origin, this will inevitably lead to fewer collocation nodes per wave length, resulting in a decrease in our ability to the resolve wave functions. To maintain the same accuracy in a larger box, N must also be increased.

In Fig. 22, we show the first four wave functions calculated with $L = 8$ and $N = 72$. In this figure we compare them to the exact solutions. Clearly, they agree well. We do not show the wave functions for $L = 16$ and $N = 174$ as they are essentially indistinguishable. However, we do calculate the maximum error on the grid $\{x_j\}_{j=1}^{N}$ for the eigenfunctions, i.e.,

$$\max \left| \psi_n(x_j) - \psi_n^{(\mathrm{HO})}(x_j) \right| \ , \tag{233}$$

where the exact wave function $\psi_n^{(\mathrm{HO})}(x)$ is calculated using `HOWaveFunction()`. The results of which are shown in Figs. 21i and 21ii. The error increases very sharply as the eigenfunction number is increased for $L = 8$, $N = 72$ with only the first 5 or so eigenfunctions being accurately calculated. However, in Fig. 21ii we see the increase in L and N pays off substantially with the first ~ 80 eigenfunctions being calculated to around $\sim 10^{-10}$ accuracy.

Obviously, not all the wave functions can be accepted for a given N. In situations when we do not have an analytic solution with which we can make a comparison. One could of course make a comparison with successive approximations or look at the size of Fourier coefficients of a wave function to get a handle on how accurate a particular wave function is. This is a very important consideration, although, we will not discuss this further for the Fourier pseudo-spectral method. However, we will take up such an analysis when considering the Chebyshev pseudo-spectral method which is most likely to be more foreign to readers and defer the investigation into the behaviour of the Fourier coefficients to later work.

Example 7.2. (Quartic Potential)

Our second example is the quartic or anharmonic potential,

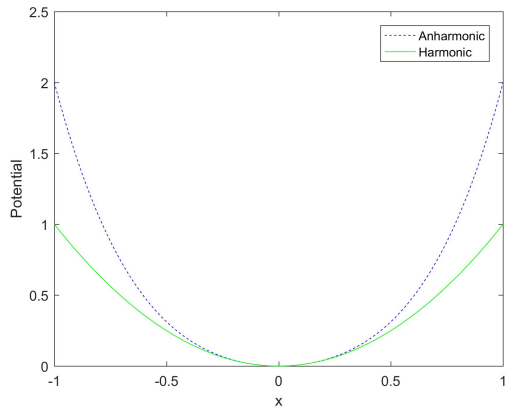
$$V_{AHO}(x) = \frac{1}{2}x^2 + \epsilon x^4 \ , \tag{234}$$

where ϵ is the quartic coupling which could be a considered perturbation parameter.

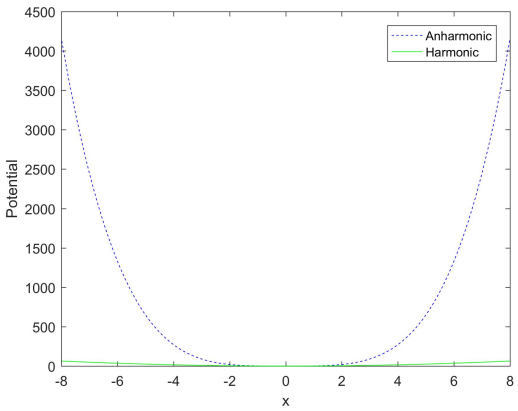
Unlike the harmonic problem, this problem does not have an exactly known analytical solution. An attempt to obtain an approximate analytical solution using a naive application of Rayleigh-Schrödinger perturbation theory one is, in this instance, confronted with a divergent perturbation series for the eigenvalues of the Hamiltonian. However, all is not lost. Padé summation can be used to extract something meaningful from this divergent series [23].

To be specific, we will be solving the following differential equation

$$\left(-\frac{d^2}{dx^2} + x^2 + x^4 \right) \psi_n(x) = \tilde{\lambda}_n \psi_n(x) \quad , \tag{235}$$



(i) The harmonic and anharmonic oscillator potentials on the truncated interval.



(ii) The harmonic and anharmonic oscillator potentials on the larger interval.

Figure 23: The quartic potential with $\epsilon = 0.5$ on the $[-1, 1]$ and $[-L, L]$ intervals. These plots are produced by using the MATLAB® script `Ex_Script_12_Fourier_Quartic.m`.

with the constants $\hbar = m = 1$, $\epsilon = 1/2$ and the periodic boundary conditions $\psi_n(-L) = \psi_n(L)$. The numerical procedure is then the same as for the harmonic oscillator problem.

In Fig. 23i and Fig. 23ii we plot the harmonic (x^2) and anharmonic potential $(x^2 + x^4)$ on the truncated interval $[-1, 1]$ and the larger interval $[-L, L]$, respectively. As can be seen in these figures the quartic potential increases very quickly for $\epsilon = 1/2$. In a dynamical situation where we might want to consider a system moving approximately harmonically ϵ should be chosen to be small, otherwise the quartic term will overwhelm the harmonic term. Because of the similarity to the harmonic problem and the faster rise of the potential with x, we would intuitively expect the anharmonic wave functions to be of a similar shape, but more compressed than, the harmonic wave functions. This is precisely what we see in Fig. 24 where we have calculated with $L = 8$ and $N = 72$.

8 Chebyshev pseudo-spectral method

Much of what was said in the section on Fourier spectral methods carries over to the Chebyshev pseudo-spectral method and more generally to methods based on other orthogonal polynomials. In contrast to the Fourier method previously examined, these bases are more appropriately suited to non-periodic problems.

As we discussed in Section 3 given a function $u(x)$ in a Hilbert space $\mathcal{H}$, its projection onto a finite dimensional subspace spanned by a finite set of trial functions $\{\phi_j\}_{j=0}^{N}$, is defined as

$$P_N u(x) = \sum_{j=0}^{N} \hat{u}_j \phi_j(x) \ , \tag{236}$$

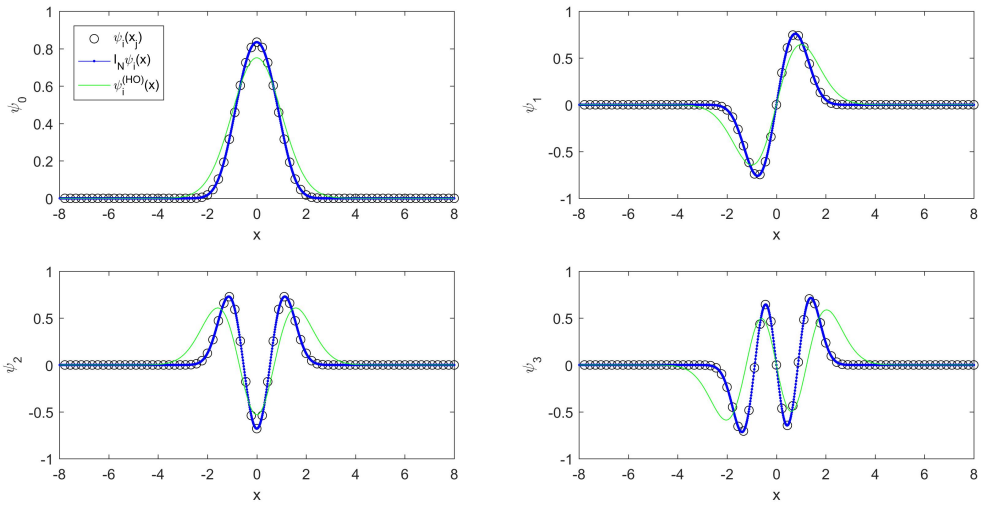


Figure 24: The first four anharmonic oscillator wave functions calculated with $L = 8$ and $N = 72$ compared with the corresponding exact harmonic oscillator wave functions. The circles mark the values at the collocation points, the blue curves are the interpolants calculated using Chebfun's `trigBary()` *function and the green curves are the corresponding exact harmonic oscillator wave functions. These plots are produced by modifying the MATLAB® script* `Ex_Script_12_Fourier_Quartic.m`.

where the exact coefficients in the expansion follow from the Hilbert space inner product,

$$\hat{u}_j = \frac{(\phi_j, u)_w}{(\phi_j, \phi_j)_w} = \frac{(\phi_j, u)_w}{\|\phi_j\|_w^2} \ , \tag{237}$$

which, in general, cannot be exactly determined.

Pseudo-spectral methods approximate these inner products using numerical quadrature. In the case of periodic functions, the Fourier basis is the natural choice and the corresponding method of quadrature one should use is the midpoint or trapezoidal rule. However, when orthogonal polynomials are chosen as the basis, the natural approximation to the integrals is a form of Gaussian quadrature, i.e., Gauss, Gauss-Radau or Gauss-Lobatto quadrature based on the chosen set of orthogonal polynomials. That is, we define the discrete inner product and its associated norm as

$$(u, v)_w \simeq (u, v)_{N,w} := \sum_{j=0}^{N} u(x_j) v(x_j) w_j \ , \quad \|u\|_{N,w}^2 = (u, u)_{N,w} \ , \tag{238}$$

where the $N+1$ numbers x_j are the nodes and the $N+1$ numbers w_j are the weights of Gauss, Gauss-Radau or Gauss-Lobatto quadrature.

The exact expansion coefficients $\hat{u}_k$ in Eq. 237 are then approximated using the discrete inner product as

$$\hat{u}_k \simeq \tilde{u}_k = \frac{1}{\gamma_k^2} \sum_{j=0}^{N} u(x_j) \phi_k(x_j) w_j \tag{239}$$

where

$$\gamma_k^2 = \sum_{j=0}^{N} \phi_k^2(x_j) w_j \quad \text{for} \quad k = 0, 1, \ldots, N \ . \tag{240}$$

Note that the norm of the polynomial basis functions ϕ_k for $k = 0, 1, \ldots, N-1$ is exactly calculated as all three forms of Gaussian quadrature are exact for polynomials up to degree $2N-1$. However, only Gauss-Radau and Gauss quadrature rules are exact for polynomials of degree $2N$ and $2N+1$, respectively. Gauss-Lobatto quadrature will only approximate the norm $\|\phi_N\|_w^2$. That is

$$\gamma_N^2 = \begin{cases} \|\phi_N\|_w^2 & , \quad \text{G. and G.R. quadrature} \\ (\phi_N, \phi_N)_{N,w} & , \quad \text{G.L. quadrature} \end{cases} \ . \tag{241}$$

Equation 239 is the counterpart to the discrete Fourier transform and we will refer to it as the discrete polynomial transform. An inverse discrete transform can then be defined using the approximate coefficients rather than the exact coefficients as

$$u(x_k) = \sum_{j=0}^{N} \tilde{u}_j \phi_j(x_k) \quad , \quad k = 0, 1, \ldots \ , \ N \ . \tag{242}$$

This is then the analogue of the inverse discrete Fourier transform which we will refer to as the inverse discrete polynomial transform.

Equation 239 and Eq. 242 provide us with a discrete transform pair analogous to the DFT and IDFT used in Section 7 . When working with the set of function evaluations $\{u(x_j)\}_{j=0}^N$ we will say we are working in configuration space and when we are working with the coefficients $\{\tilde{u}_j\}_{j=0}^N$ that we are working in coefficient space. Once again we are free to work in either space, but at times one space may be more convenient than the other. We can always transform back and forth between these spaces using Eq. 239 and Eq. 242. This can be done using linear algebra and in the special case of Chebyshev polynomials this can be accomplished efficiently using the Fourier cosine transform in $\mathcal{O}(N \log N)$ floating point operations.

We can then use Eq. 242 to define the polynomial interpolant in the spectral basis as

$$I_N u(x) = \sum_{j=0}^{N} \tilde{u}_j \phi_j(x) \tag{243}$$

where $x \in \mathbb{R}$. In this report we have chosen to work in configuration space, so we would like the interpolant expressed in terms of function evaluations rather than the coefficients. Equation 243 is a polynomial and can therefore also be represented in terms of the Lagrange or Cardinal basis polynomials associated with the nodes $\{x_k\}_{k=0}^N$ obtained from one of the variants of Gaussian quadrature. This means the interpolant can be written in the form

$$I_N u(x) = \sum_{j=0}^{N} u(x_j) C_j(x) \ , \tag{244}$$

where the $C_j(x)$ are the cardinal basis functions that satisfy $C_j(x_k) = \delta_{jk}$.

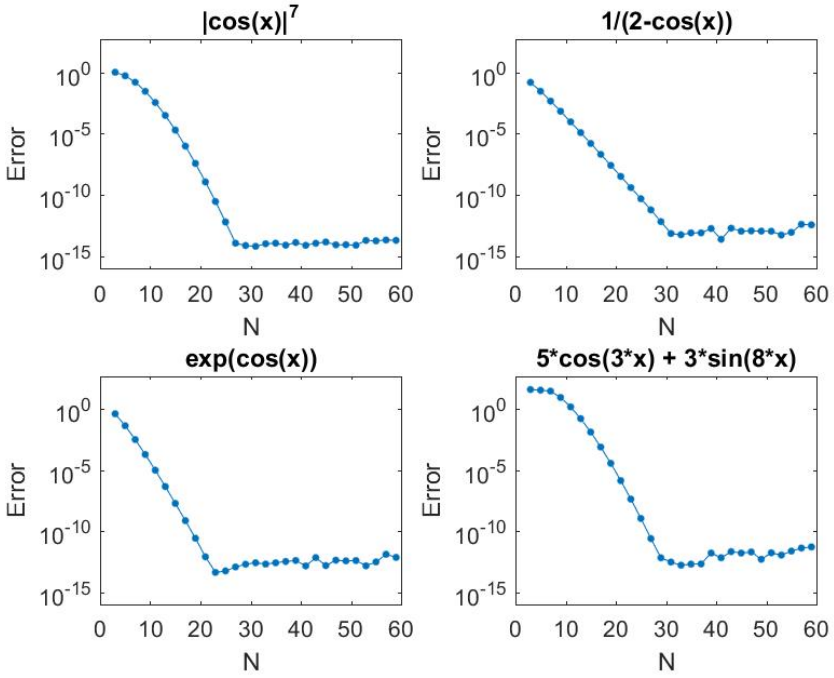


Figure 25: The error in the spectral derivative of four functions with varying levels of smoothness plotted as a function of N. These plots are produced using the MATLAB® script `Ex_Script_8_Chebyshev_Diff.m`.

From here on we specialise to the basis of Chebyshev polynomials of the first kind, i.e., $T_n(x)$. As we want to impose Dirichlet boundary conditions when we solve the Schrödinger equation we will choose to work on the $N+1$ point Gauss-Lobatto-Chebyshev grid. This is the grid consisting of the boundary points ± 1 and the extrema of $T_N(x)$ as already proved in Proposition 6.3. The nodes and weights for this form of quadrature are given in Table 4. The cardinal polynomials with these choices are derived in Theorem 8.1.

Just as we did for the Fourier spectral method we can approximate the derivative of the function on the grid by differentiating the polynomial interpolant

$$u'(x_i) \simeq (I_N u(x))'\big|_{x=x_i} = \sum_{j=0}^{N} u(x_j) C_j'(x_i) \ . \tag{245}$$

Once again, this can be formulated in terms of matrices by defining the first order Chebyshev differentiation matrix as

$$(D_N^{(1)}) = C_j'(x_i) \ , \tag{246}$$

where the cardinal functions $C_j(x)$ are this time derived in Theorem 8.1. The Chebyshev differentiation matrices defined by Eq. 246 are derived in Theorem 8.2. It can be shown that the higher order Chebyshev differentiation matrix can be calculated by matrix multiplication [12]

$$(D_N^{(\ell)})_{ij} = (D_N^{(1)})^{\ell} \ . \tag{247}$$

As we did for the Fourier method in Section 7, we show in Figs. 25 and 26 examples of the accuracy possible with the Chebyshev differentiation matrices. The functions in Fig. 25 are

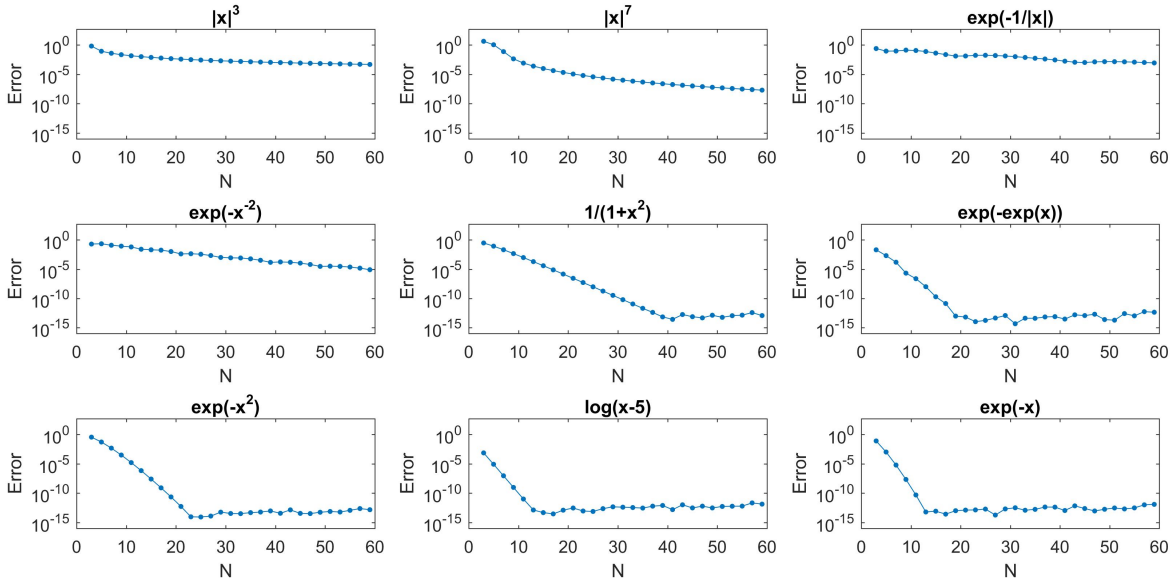


Figure 26: The error in the spectral derivative of several functions with varying levels of smoothness plotted as a function of N. These plots are produced using the MATLAB® script `Ex_Script_8_Chebyshev_Diff.m`.

the same as the ones in Fig. 16 and on comparison the Chebyshev differentiation matrices do better than the Fourier ones for the first three functions. In Fig. 26 we show the error in using the Chebyshev differentiation matrices applied to a selection of non-periodic functions with varying levels of smoothness.

The derivative of the interpolant is not equal to the interpolant of the derivative. However, as the number of points in the collocation grid increases, one cannot tell the difference. This statement was made for the Fourier interpolant but we want to point out that it is still valid for the Chebyshev interpolant. We illustrate this in Fig. 28 with the non-periodic function show in Fig. 27. For comparison we also show the interpolation error in Fig. 29.

Theorem 8.1. *(Chebyshev Interpolant) The Chebyshev interpolant on the Gauss-Lobatto-Chebyshev grid, i.e., the extrema plus endpoints grid*

$$I_N u(x) = \sum_{j=0}^{N} C_j(x) u(x_j) \tag{248}$$

where the cardinal functions are given by

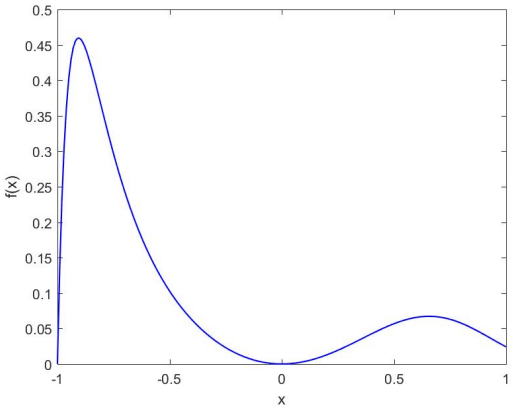
$$C_j(x) = \frac{(-1)^{j+1}(1-x^2)T_N'(x)}{c_j N^2 (x - x_j)} \ , \tag{249}$$

where

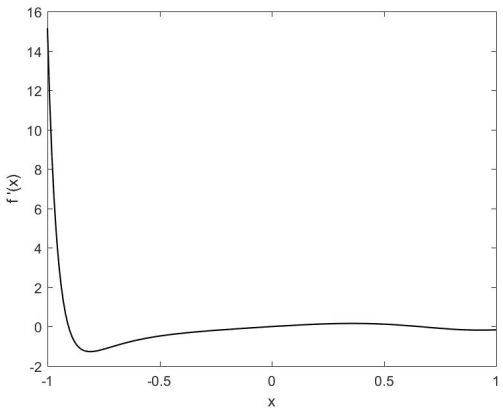
$$c_j = \begin{cases} 1 & , \quad if \quad j = 1, \ \ldots, \ N-1 \\ 2 & , \quad if \quad j = 0 \quad or \quad N \end{cases} \ . \tag{250}$$

Proof: The cardinal functions for the Chebyshev expansion on the Lobatto grid (extrema plus boundary points), i.e.,

$$x_j = \cos\left(\frac{j\pi}{N}\right) \quad , \quad j = 0, 1, \ldots, N \tag{251}$$



(i) The function $f(x) = \frac{x^2 e^{\ xe^{x^2}} \log(x+2)}{(x^3+2)}$.



(ii) The derivative of $f(x)$ *in Fig. 27i.*

Figure 27: An example non-periodic function. These plots are produced using the MATLAB® script `Ex_Script_9_Chebyshev_Interp.m`.

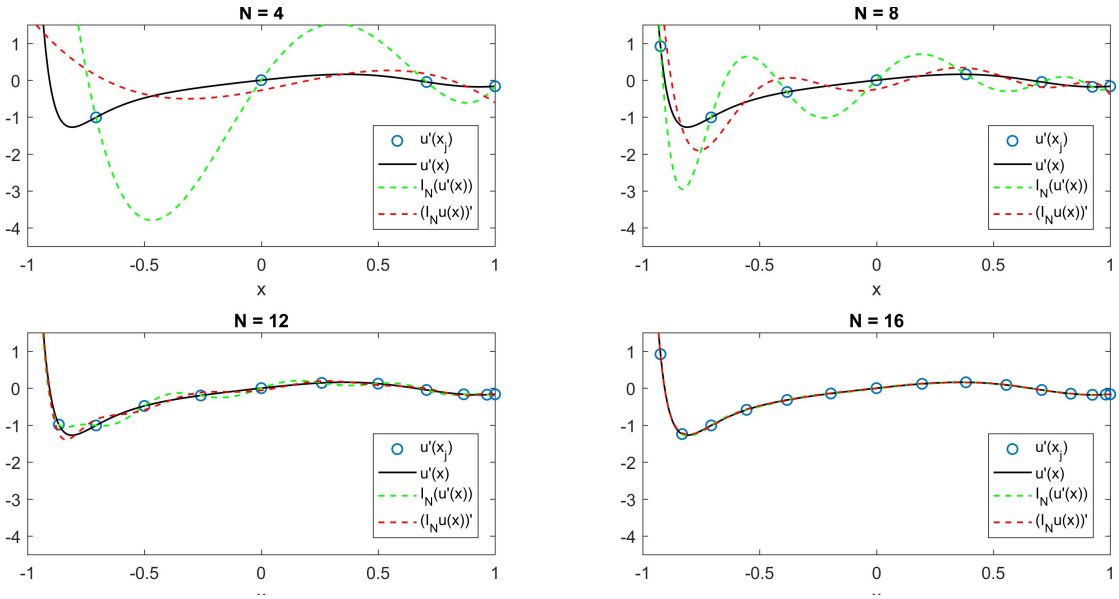


Figure 28: The derivative of the function $f(x)$ *shown in Fig. 27, the derivative the interpolant* $(I_N f(x))'$ *and interpolant of the derivative* $I_N(f'(x))$. *The circles denote the collocation points. These plots are produced using the MATLAB® script* `Ex_Script_9_Chebyshev_Interp.m`.

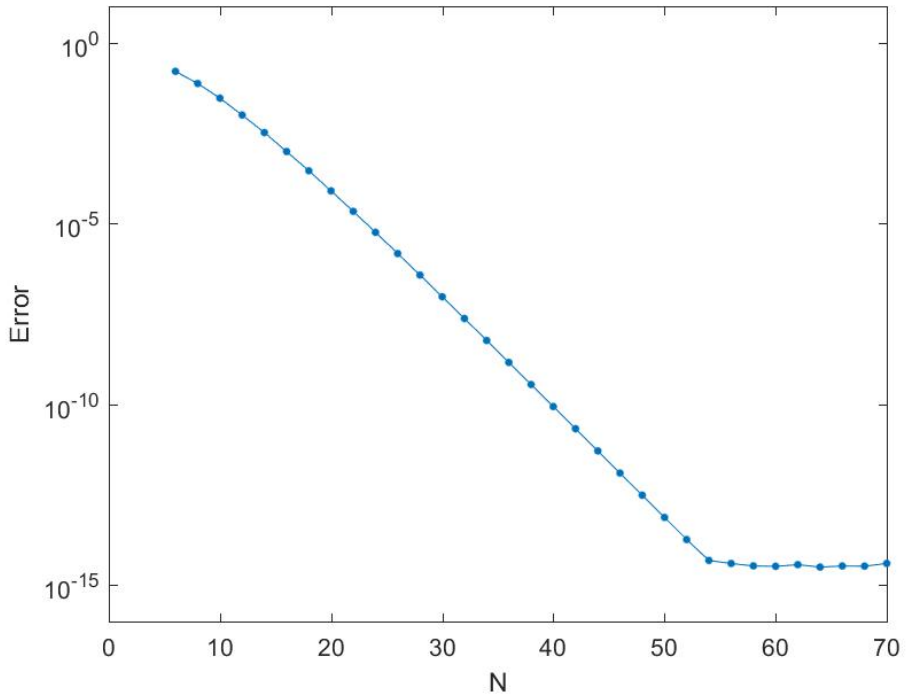


Figure 29: The maximum error of the interpolant of the function $f(x)$ plotted in Fig. 27 shown as a function of N. This figure is produced using the MATLAB® script ***Ex_Script_10_Chebyshev_Interp_Error.mf.***

are (see the argument leading to Eq. 89 and Eq. 90 and Proposition 6.3)

$$\begin{aligned} C_j(x) &= \frac{\phi_{N+1}(x)}{\phi'_{N+1}(x_j)(x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{\left[(1-x^2)T'_N(x)\right]'_{x=x_j}(x-x_j)} \,. \end{aligned} \tag{252}$$

Equation 252 can be shown to be equivalent to Eq. 248. The proof relies on the use of the differential equation that can be used to define the Chebyshev polynomials, Eq. B16.

- For $j = 1, \ldots, N-1$ we have $T'_n(x_j) = 0$. As a consequence of this and using Eq. B16 we obtain

$$\begin{aligned} C_j(x) &= \frac{(1-x^2)T'_N(x)}{\left(-2x_jT'_N(x_j) + (1-x_j^2)T''_N(x_j)\right)(x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{(1-x_j^2)T''_N(x_j)(x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{\left[x_jT'_N(x_j) - N^2T_N(x_j)\right](x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{(-N^2)T_N(x_j)(x-x_j)} \,. \end{aligned} \tag{253}$$

Using the trigonometric form of the Chebyshev polynomial, i.e.,

$$T_N(x_j) = \cos\left(N \arccos\left(\cos\left(\frac{j\pi}{N}\right)\right)\right) = \cos\left(N\frac{j\pi}{N}\right) = \cos(j\pi) = (-1)^j \ , \tag{254}$$

we obtain

$$C_j(x) = \frac{(-1)^{j+1}(1-x^2)T'_N(x)}{N^2(x-x_j)} \ . \tag{255}$$

- For $i = 0$ or N, i.e, $x_j = \pm 1$, we have

$$\begin{aligned} C_j(x) &= \frac{(1-x^2)T'_N(x)}{\left[-2x_jT'_N(x_j) + (1-x_j^2)T''_N(x_j)\right](x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{-2x_jT'_N(x_j)(x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{2\left(-(1-x_j^2)T''_N(x_j) - N^2T_N(x_j)\right)(x-x_j)} \\ &= \frac{(1-x^2)T'_N(x)}{(-2)N^2T_N(x_j)(x-x_j)} \\ &= \frac{(-1)^{j+1}(1-x^2)T'_N(x)}{2N^2(x-x_j)} \ . \end{aligned} \tag{256}$$

Thus Eq. 256 and Eq. 255 give the desired result in Eq. 249. ■

Theorem 8.2. *(Chebyshev Differentiation Matrix) The first order Chebyshev differentiation matrix on the Gauss-Chebyshev-Lobatto grid, i.e., the extrema and end points grid, is given by*

$$(D_N^{(1)})_{00} = \frac{2N^2+1}{6} \tag{257}$$

$$(D_N^{(1)})_{NN} = -(D_N^{(1)})_{00} = -\frac{2N^2+1}{6} \tag{258}$$

$$(D_N^{(1)})_{jj} = -\frac{x_j}{2(1-x_j^2)} \ , \quad j = 1, \ldots, N-1 \tag{259}$$

$$(D_N^{(1)})_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{x_i - x_j} \ , \quad i \neq j \ , \ i,j = 0, \ldots, N \tag{260}$$

with

$$c_i = \begin{cases} 2 & , \quad i = 0 \ or \ \ N \\ 1 & , \quad otherwise \end{cases} \tag{261}$$

Proof: The Chebyshev differentiation matrix is defined as

$$(D_N^{(1)})_{ij} = C'_j(x_i) \ , \tag{262}$$

where the cardinal function is given by Eq. 249. We begin by calculating the derivative of the cardinal function,

$$\begin{aligned} C'_j(x) &= \frac{d}{dx}\left(\frac{(-1)^{j+1}(1-x^2)T'_N(x)}{c_jN^2(x-x_j)}\right) \\ &= (-1)^{j+1}\frac{(x-x_j)\left[-2xT'_N(x) + (1-x^2)T''_N(x)\right] - (1-x^2)T'_N(x)}{c_jN^2(x-x_j)^2} \ . \end{aligned} \tag{263}$$

We can simplify this further by using the differential equation (Eq. B16) that defines the Chebyshev polynomials,

$$C'_j(x) = (-1)^{j+1}\frac{(x-x_j)\left[-xT'_N(x) - N^2T_N(x)\right] - (1-x^2)T'_N(x)}{c_jN^2(x-x_j)^2} \,. \tag{264}$$

Now we handle each case separately.

- For $i \neq j$, $i \neq 0$ or N, but $j = 0, 1, \ldots, N$. For $i \neq j \neq 0$ and $i \neq j \neq N$ we have $T'_N(x_i) = 0$ by definition of our grid. Thus we can simplify Eq. 264 to

$$\begin{aligned} C'_j(x_i) &= (-1)^{j+1}\frac{(x_i-x_j)\left[-x_iT'_N(x_i) - N^2T_N(x_i)\right] - (1-x_i^2)T'_N(x_i)}{c_jN^2(x_i-x_j)^2} \\ &= \frac{(-1)^jT_N(x_i)}{c_j(x_i-x_j)} \\ &= \frac{(-1)^j\cos\left(N\arccos\left(\cos\left(\frac{i\pi}{N}\right)\right)\right)}{c_j(x_i-x_j)} \\ &= \frac{(-1)^j\cos\left(i\pi\right)}{c_j(x_i-x_j)} \\ &= \frac{(-1)^{i+j}}{c_j(x_i-x_j)} \,. \end{aligned} \tag{265}$$

- For $i \neq j$, $i = 0$ or N we have $x_i = \pm 1$ which allows us to simplify Eq. 264 as

$$\begin{aligned} C'_j(x_i) &= (-1)^{j+1}\frac{(x_i-x_j)\left[-x_iT'_N(x_i) - N^2T_N(x_i)\right] - (1-x_i^2)T'_N(x_i)}{c_jN^2(x_i-x_j)^2} \\ &= (-1)^{j+1}\frac{\left[-x_iT'_N(x_i) - N^2T_N(x_i)\right]}{c_jN^2(x_i-x_j)} \,. \end{aligned} \tag{266}$$

  Using the Cheybshev differential equation (Eq. B16) we get

$$\begin{aligned} C'_j(x_i) &= (-1)^{j+1}\frac{\left[-(1-x_i^2)T''_N(x_i) - 2N^2T_N(x_i)\right]}{c_jN^2(x_i-x_j)} \\ &= (-1)^j\frac{2T_N(x_i)}{c_j(x_i-x_j)} \\ &= (-1)^j\frac{2\cos\left(N\arccos\left(\cos\left(\frac{i\pi}{N}\right)\right)\right)}{c_j(x_i-x_j)} \\ &= (-1)^j\frac{2\cos\left(i\pi\right)}{c_j(x_i-x_j)} \\ &= \frac{2(-1)^{i+j}}{c_j(x_i-x_j)} \,. \end{aligned} \tag{267}$$

  Thus Eq. 265 and Eq. 267 together give Eq. 260.

- For $i = j$ but not 0 or N, if we try to evaluate the limit $x \to x_i$ we will get an indeterminate expression. We must use l'Hospital's rule . Thus if we define

$$C'_j(x) = (-1)^{j+1}\frac{(x-x_j)\left[-xT'_N(x) - N^2T_N(x)\right] - (1-x^2)T'_N(x)}{c_jN^2(x-x_j)^2} =: \frac{A}{\tilde{A}} \,, \tag{268}$$

then the quantity of interest is

$$\frac{A'}{\tilde{A}'} = (-1)^{j+1}\left(xT'_N(x) - N^2T_N(x) + (x-x_j)\left[-T'_N(x) - xT''_N(x) - N^2T'_N(x)\right]\right. \\ \left. - (1-x^2)T''_N(x)\right) \\ \Big/\left(2c_jN^2(x-x_j)\right) \ . \tag{269}$$

Using Chebyshev's equation (Eq. B16) again we simplify this to

$$\frac{A'}{\tilde{A}'} = (-1)^{j+1}\frac{\left[-(1+N^2)T'_N(x) - xT''_N(x)\right]}{2c_jN^2} \ . \tag{270}$$

Now taking the limit

$$\begin{aligned} C'_j(x_j) &= \lim_{x\to x_j}\frac{A'}{\tilde{A}'} \\ &= \lim_{x\to x_j}(-1)^{j+1}\frac{\left[-(1+N^2)T'_N(x) - xT''_N(x)\right]}{2c_jN^2} \\ &= \frac{(-1)^j x_j T''_N(x_j)}{2c_jN^2} \ . \end{aligned} \tag{271}$$

Using Chebyshev's equation (Eq. B16) we finally obtain

$$\begin{aligned} C'_j(x_j) &= \frac{(-1)^j x_j}{2c_jN^2}\left(\frac{x_jT'_N(x_j) - N^2T_N(x_j)}{1-x_j^2}\right) \\ &= -\frac{(-1)^j x_j T_N(x_j)}{2c_j(1-x_j^2)} \\ &= -\frac{(-1)^j x_j \cos\left(N\arccos\left(\cos\left(\frac{j\pi}{N}\right)\right)\right)}{2c_j(1-x_j^2)} \\ &= -\frac{(-1)^j x_j \cos(j\pi)}{2c_j(1-x_j^2)} \\ &= -\frac{(-1)^{2j} x_j}{2c_j(1-x_j^2)} \\ &= -\frac{x_j}{2c_j(1-x_j^2)} \\ &= -\frac{x_j}{2(1-x_j^2)} \ . \end{aligned} \tag{272}$$

- For $i = j = 0$ and $i = j = N$, we start from Eq. 270 (note that $c_0 = c_N = 2$)

$$\frac{A'}{\tilde{A}'} = (-1)^{j+1}\frac{\left[-(1+N^2)T'_N(x) - xT''_N(x)\right]}{4N^2} \tag{273}$$

and use

$$
\begin{aligned}
x &= \cos\theta && (274)\\
T_N(x) &= \cos\left(N \arccos(x)\right) = \cos\left(N\theta\right) && (275)\\
T'_N(x) &= \frac{N\sin(N\arccos(x))}{\sqrt{1-x^2}} = \frac{N\sin(N\theta)}{\sin(\theta)} && (276)\\
T''_N(x) &= \frac{N^2\cos(N\arccos(x))}{1-x^2} + \frac{(Nx\sin(N\arccos(x))}{(1-x^2)^{3/2}} && (277)\\
&= -\frac{N^2\cos(N\theta)}{\sin^2(\theta)} + \frac{N\cos(\theta)\sin(N\theta)}{\sin^3(\theta)} && (278)
\end{aligned}
$$

to obtain

$$
\begin{aligned}
\frac{A'}{\tilde{A}'} &= \frac{(-1)^{j+1}}{4N^2}\frac{N}{4}\csc^3(\theta)\Big((-4-2N^2)\sin(N\theta) + (N-N^2)\sin((2-N)\theta) \\
&\qquad + (N+N^2)\sin((2+N)\theta)\Big)\ . \qquad (279)
\end{aligned}
$$

Taking the limit of $x \to x_0$ is equivalent to $\theta \to 0$ and $x \to x_N$ is equivalent to $\theta \to \pi$. If we naively take these limits we get an indeterminate expression again. To evaluate them we need to use l'Hospital's rule three times because of the $\csc^3(\theta)$. Thus, we are interested the third derivative of numerator divided by the third derivative of the denominator in Eq. 279, which is

$$
\begin{aligned}
\frac{B}{\tilde{B}} &:= \frac{(-1)^{j+1}}{16N}\Big(-(-2+N)^3(-1+N)N\cos((-2+N)\theta) + 2N^3(2+N^2)\cos(N\theta) \\
&\qquad -N(1+N)(2+N)^3\cos((2+N)\theta)\Big) \\
&\qquad / \left(6\cos^3(\theta) - 21\cos(\theta)\sin^2(\theta)\right)\ . \qquad (280)
\end{aligned}
$$

Therefore, we finally obtain for $j = 0$

$$
\begin{aligned}
(D_N^{(1)})_{00} &= \lim_{\theta\to 0}\frac{B}{\tilde{B}} \\
&= \frac{\frac{(-1)^{0+1}}{16N}\left(-(-2+N)^3(-1+N)N + 2N^3(2+N^2) - N(1+N)(2+N)^3\right)}{6} \\
&= \frac{2N^2+1}{6} \qquad (281)
\end{aligned}
$$

and for $j = N$

$$
\begin{aligned}
(D_N^{(1)})_{NN} &= \lim_{\theta\to\pi}\frac{B}{\tilde{B}} \\
&= \frac{\frac{(-1)^{N+1}}{16N}\cos N\pi\left(-(-2+N)^3(-1+N)N + 2N^3(2+N^2) - N(1+N)(2+N)^3\right)}{-6} \\
&= \frac{\frac{(-1)^{2N+1}}{16N}\left(-(-2+N)^3(-1+N)N + 2N^3(2+N^2) - N(1+N)(2+N)^3\right)}{-6} \\
&= -\frac{2N^2+1}{6}\ . \qquad (282)
\end{aligned}
$$

■

8.1 Time-independent Schrödinger's equation examples

Example 8.1. (Harmonic Oscillator)

This is the same example we applied the Fourier pseudo-spectral method to in Section 7.1. When using the Fourier method we considered periodic boundary conditions. The Fourier basis is periodic, so no adjustments were necessary to implement the periodic boundary conditions. However, here we will want to implement the Dirichlet boundary conditions that the wave functions are zero on the boundary, so we will have to make a slight modification to enforce them.

The set of nodes we have chosen to work on is the $N+1$-point Gauss-Lobatto grid which includes the boundary points. The inclusion of the boundary points allows us to easily implement boundary conditions at these points by using the boundary bordering method [10, 12]. The boundary bordering method is very simple, one replaces the collocation condition on the boundary (i.e., that the residual is zero on the boundary or equivalently that the differential equation is satisfied) with the enforcement of the boundary conditions. In practice, once the $(N+1)\times(N+1)$ Chebyshev differentiation matrix on the Gauss-Lobatto grid is constructed we strip the first and last row and column and use the $(N-1)\times(N-1)$ submatrix to construct the linear differential operator. Once the eigensystem is found, our boundary condition that the wave functions be zero on the boundary is enforced by adding zero to the beginning and end of each eigenvector.

The subsequent calculation follows the same route as for the Fourier method, except with changes in the choice of numerical quadrature and interpolation method. As we have already alluded to in earlier sections of this report we use Clenshaw-Curtis quadrature to normalise the wave functions and barycentric interpolation to interpolate our solutions to a finer grid. The functions used to do this are `chebpts()` and `bary()` which are included in Chebfun [27].

In Figs. 30i and 30ii we show the error in first 6 eigenvalues for $L=8$ and $L=16$. These are to be compared to Figs. 19i and 19ii, we see immediately that there is about a factor of two efficiency obtained by using the Fourier method, although, the eigenvalues still become more accurately reproduced as N increases just as in the Fourier case. The corresponding figures showing all the eigenvalues obtained using $L=8$, $N=72$ and $L=16$, $N=400$ are Figs. 31i and 31ii . Just as in the Fourier case the eigenvalues corresponding to more highly excited states are less accurately calculated than the lower lying states. We also witness the same effect when we increase the box size, L, i.e., that N must also be increased to achieve the same accuracy for a given eigenfunction.

We calculate the maximum error on the grid for the eigenfunctions. The results of which are shown in Figs. 32i and 32ii. The error increases very sharply as the eigenfunction number is increased for $L=8$, $N=72$ with only the first few eigenfunctions being accurately calculated. However, in Fig. 32ii we see the increase in L and N again pays off with the first ~ 70 eigenfunctions being calculated to around $\sim 10^{-10}$ accuracy.

In the absence of an exact solution one would generally increase the number of points in the grid comparing two successive solutions to determine whether or not some predetermined accuracy had been met or not. As we have previously stated, one can choose to work in either configuration or coefficient space and if desired transition between them. The eigenfunction values on the grid are in one-to-one correspondence with the Chebyshev coefficients. We will

now look at the behaviour of the Chebyshev coefficients for a selection of wave functions to try and get another handle on which eigenfunctions are accurately calculated by this method, because it is quite clear not all them are.

In Figs. 33 and 34 we show the wave function coefficients where $L = 8$, $N = 72$ and $L = 16$, $N = 400$, respectively. To obtain these coefficients we have made use of the Chebfun function `chebcoeffs()`. In regards to these figures we would like to make clear the following. Because the harmonic potential is symmetric, the wave functions are either even or odd functions. The Chebyshev polynomials also have this property, i.e.,

$$T_n(x) = (-1)^n T_n(-x) \ . \tag{283}$$

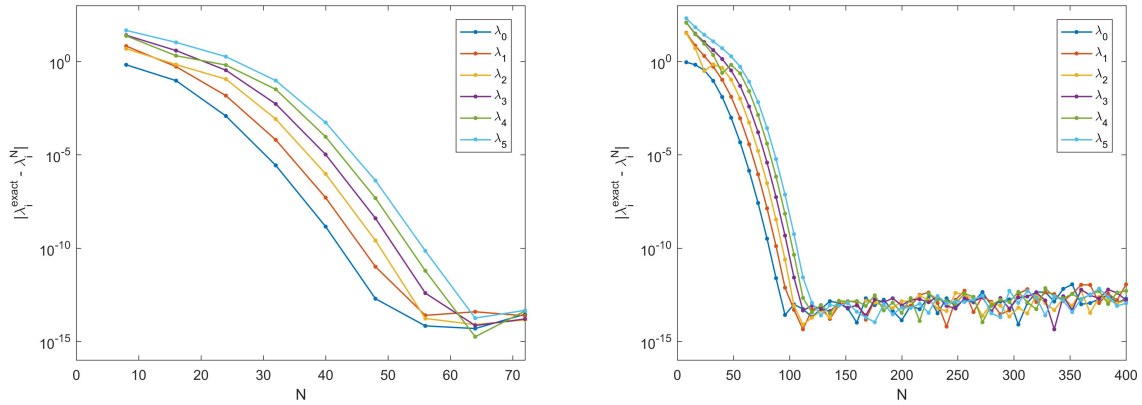
Therefore the even eigenfunctions will only have even coefficients and correspondingly the odd eigenfunctions will only have odd coefficients. In these figures we have separated odd (red) and even (blue) coefficients and the implication from the symmetry of the potential is seen in these figures.

In Fig. 33 we see that only the coefficients of ψ_0 and ψ_4, from the selected wave functions, shows an appreciable decrease. The coefficients of the other wave functions decrease little if any. However, in Fig. 34 which corresponds to $L = 16$, $N = 400$ all the wave functions shown from ψ_0 to ψ_{180} show significant decrease. Overall the coefficients tend to decrease quite quickly beyond a certain point, but this threshold is pushed higher and higher for more excited states. Also, placing the problem in a larger box can be seen to exacerbate the effect. If we compare the plot of ψ_0 coefficients for the two different boxes, i.e., $L = 8$ and $L = 16$, one does not see a significant drop in the values of the coefficients in the larger box till much later, i.e., that is beyond the 100th coefficient as compared to the 60th in the smaller box.

These observations are to be anticipated since higher energy states can naturally have more nodes and be more oscillatory, hence more Chebyshev polynomials will be needed to represent a given wave function. Moreover, as the Chebyshev polynomials tend to oscillate more near the boundaries rather than in the interior of the domain, one would expect more Chebyshev polynomials to be needed to represent a particular wave function. Equivalently in configuration space, more collocation points per wavelength are needed to resolve more oscillatory wave functions. However, Chebyshev points tend to be more clustered towards the boundaries where the wave functions are expected to be small and approaching zero with fewer nodes in the interior were they are needed. This effect is likely to be exacerbated by increasing the box size that we put the problem in, which is why when we compare the plot of ψ_0 coefficients for the two different boxes, one does not see a significant drop in the values of the coefficients till much later in the larger box. This is also an explanation for the similar behaviour of the eigenvalues.

In Figs. 35 and 36 we show the coefficients of the square of a selection of wave functions ($\rho_i = \psi_i^* \psi_i$) for both $L = 8$, $N = 72$ and $L = 16$, $N = 400$, respectively. These functions are even so they only have even coefficients. We see the same behaviour of the appreciable drop in the absolute value of the coefficients beyond a certain point, but it occurs for fewer eigenfunctions.

In the absence of an exact solution, if we use the size of the Chebyshev coefficients of these expansions as a measure of the accuracy of our calculation, it would provide us with a way of thinning the herd of wild eigenfunctions. Simply requiring the absolute value of the wave function coefficients to be smaller than some tolerance would appear to be too lenient on



(i) Error in the first 6 eigenvalues for $L = 8$ and $N = 72$.

(ii) Error in the first 6 eigenvalues for $L = 16$ and $N = 400$.

Figure 30: Error in harmonic oscillator eigenvalues found using the Chebyshev pseudo-spectral method. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

inspection of Figs. 32i and 32ii. In contrast, requiring the coefficients of the square of the wave functions to be smaller than some tolerance is a much stricter constraint, and possibly too severe. A compromise would be to truncate the set of wave functions somewhere between these two extremes. Obviously, a more in depth analysis is needed to determine an optimal way of truncating the set of wave functions. We defer said analysis to future work.

Example 8.2. (Quartic Potential)

This is essentially the same example that we applied the Fourier method too in Section 7.1, except here we will take $\epsilon = 0.001, 0.01$ and 0.5. When ϵ is small the problem is approximately the harmonic oscillator problem. We do not show the eigenfunctions in this case as they are indistinguishable from the figures shown earlier, see Fig. 24. Instead, we show in Figs. 37, 39 and 41 the coefficients of the Chebyshev series for the wave functions for the larger box with $L = 16$ and $N = 400$, respectively for $\epsilon = 0.001, 0.01$ and 0.5. Similarly, in Figs. 38, 40 and 42 we show the corresponding coefficients of the square of the wave functions. As the potential is again symmetric the wave functions are either even or odd functions and therefore their Chebyshev expansions have only even or odd coefficients.

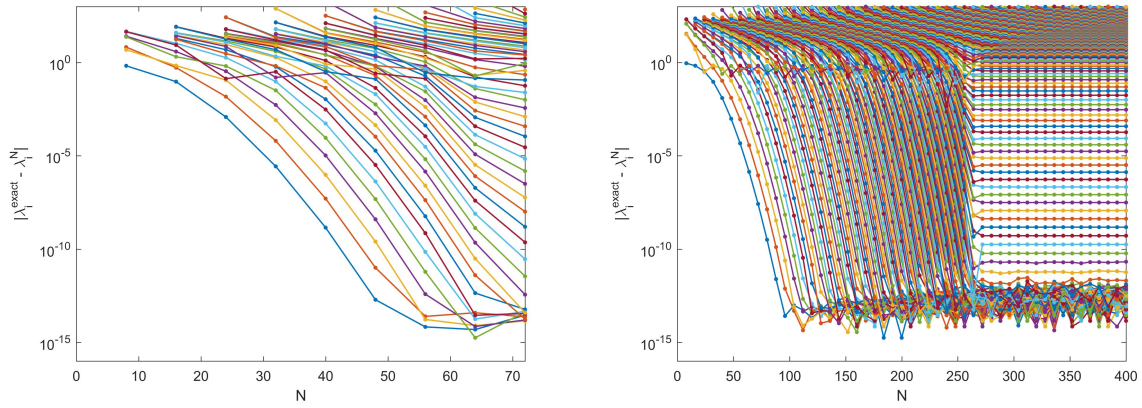
For $\epsilon = 0.001$, the behaviour of the coefficients in both expansions appears to be relatively unchanged by inclusion of the perturbation. As ϵ is increased it is obvious that the coefficients become significantly affected. The drop in the absolute value of the coefficients tends to not occur till later in expansions or even not at all for some of the wave functions. This leads one to conclude that as ϵ increases fewer eigenfunctions are accurately calculated and more must be discarded.

Example 8.3. (Linear Potential)

We now consider the Schrödiger equation with the linear potential

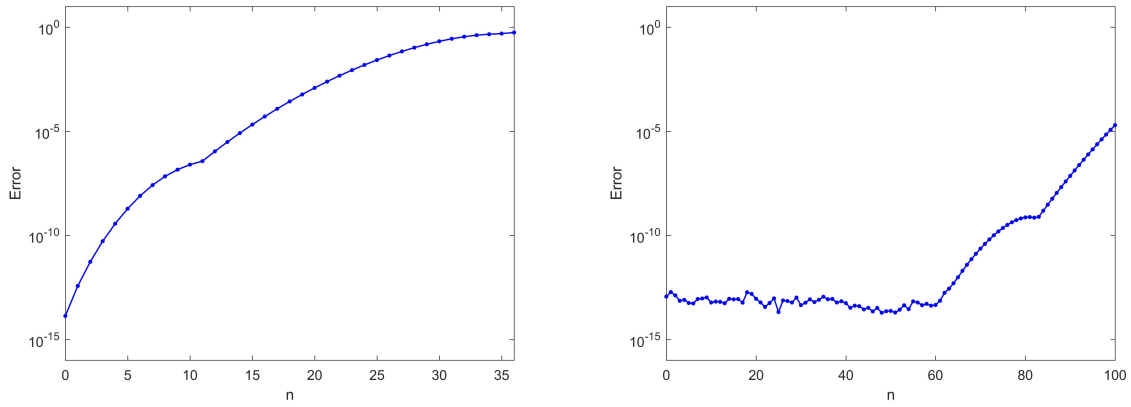
$$V(x) = \begin{cases} \infty & , \quad x < 0 \\ mgx & , \quad x > 0 \end{cases} . \tag{284}$$

The exact solution to this problem is known and is given in App. A.2. Its solution is in terms



(i) Error in all eigenvalues for $L = 8$, $N = 72$. *(ii) Error in all eigenvalues for $L = 16$, $N = 400$.*

Figure 31: The error in the harmonic oscillator eigenvalues using the Chebyshev pseudo-spectral method. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.



(i) Error in the harmonic oscillator wave functions for $L = 8$ and $N = 72$. *(ii) Error in the harmonic oscillator wave functions for $L = 16$ and $N = 400$.*

Figure 32: Error in harmonic oscillator wave functions using the Chebyshev pseudo-spectral method. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

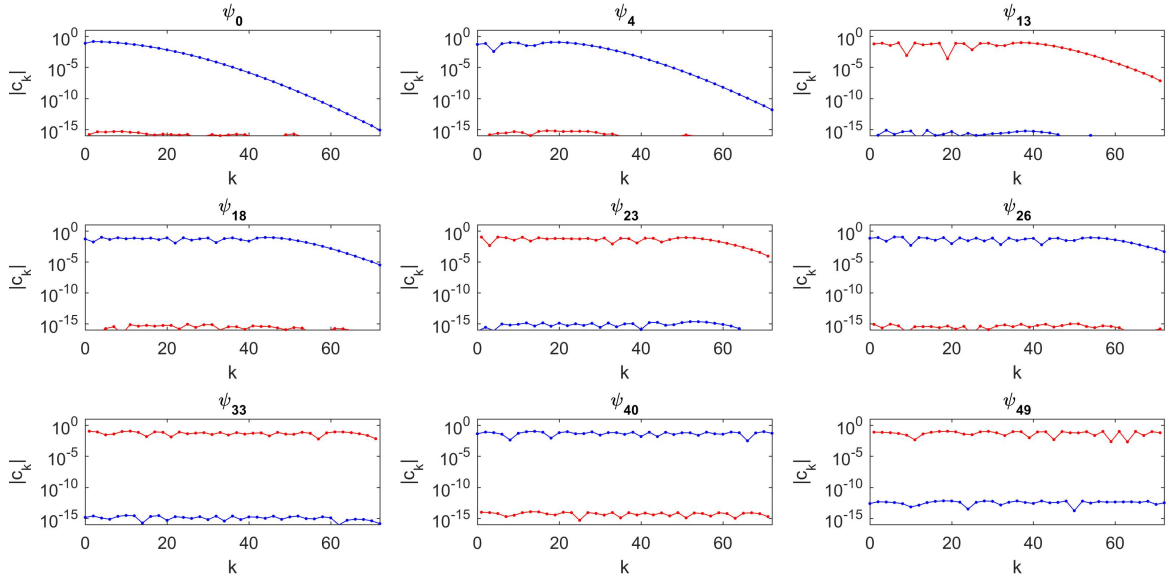


Figure 33: Chebyshev coefficients for a few of the harmonic oscillator wave functions for $L = 8$ and $N = 72$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

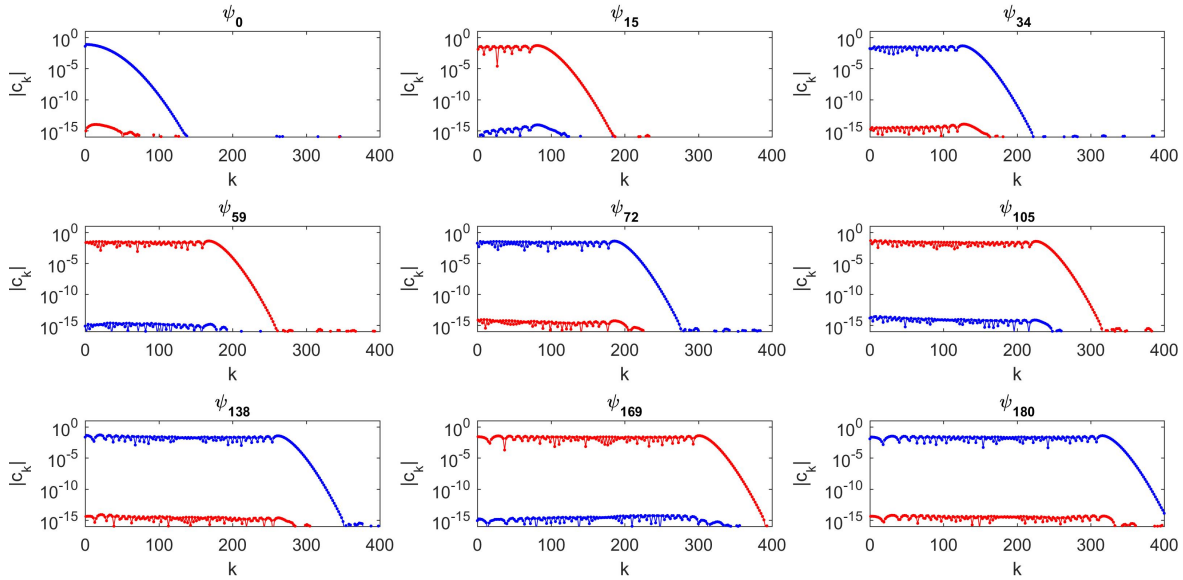


Figure 34: Chebyshev coefficients for a few of the harmonic oscillator wave functions for $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

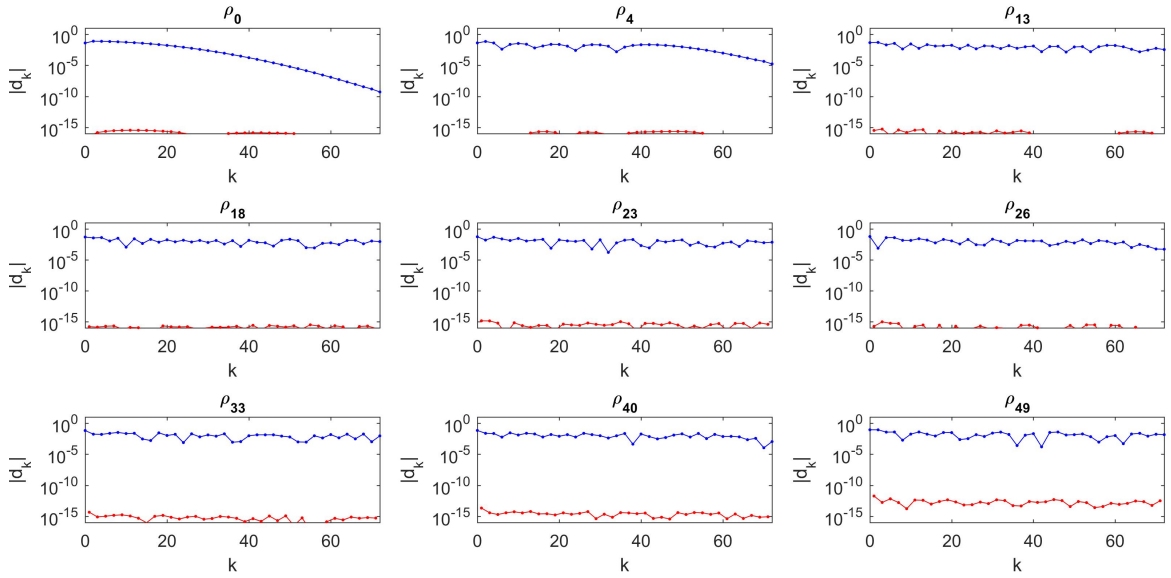


Figure 35: Chebyshev coefficients for the square of a few of the harmonic oscillator wave functions for $L = 8$ and $N = 72$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

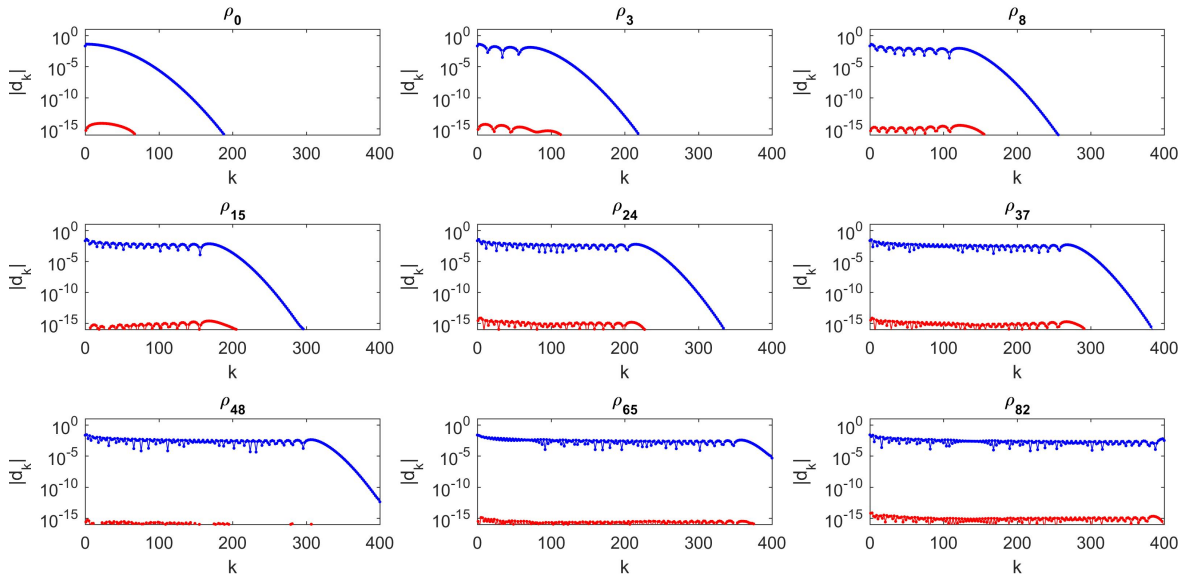


Figure 36: Chebyshev coefficients for the square of a few of the harmonic oscillator wave functions for $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

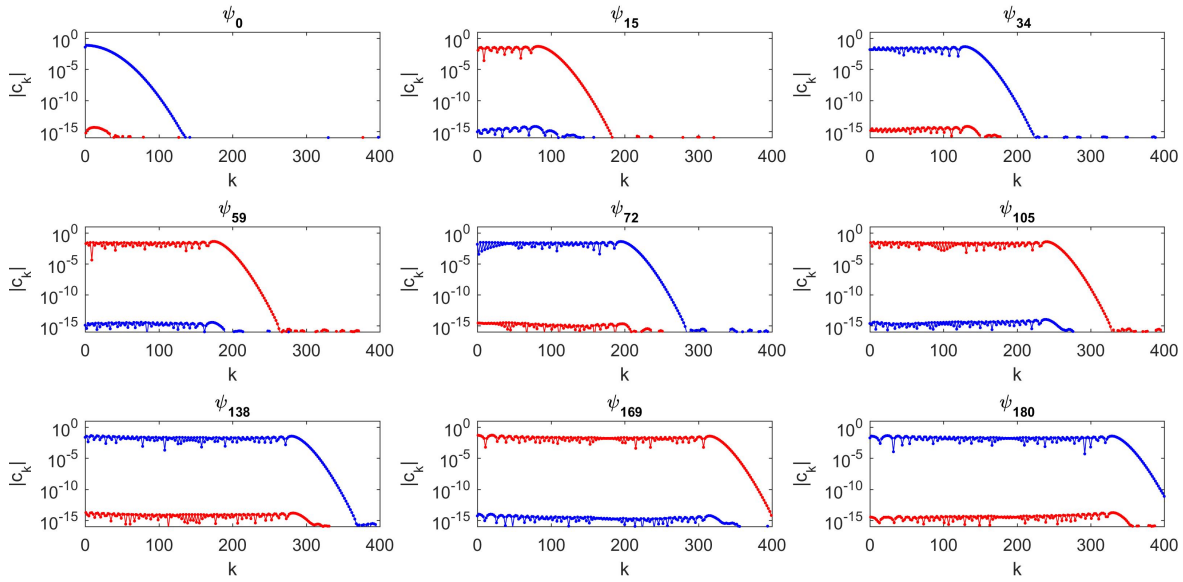


Figure 37: Chebyshev coefficients for a few of the quartic potential wave functions for $\epsilon = 0.001$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

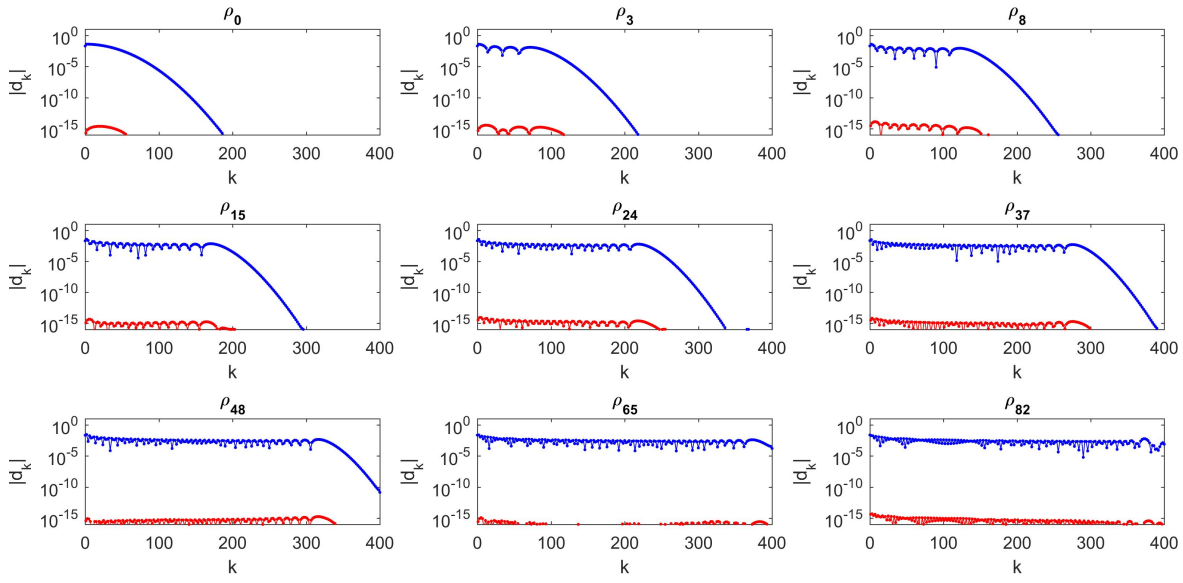


Figure 38: Chebyshev coefficients for the square of a few of the quartic potential wave functions for $\epsilon = 0.001$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

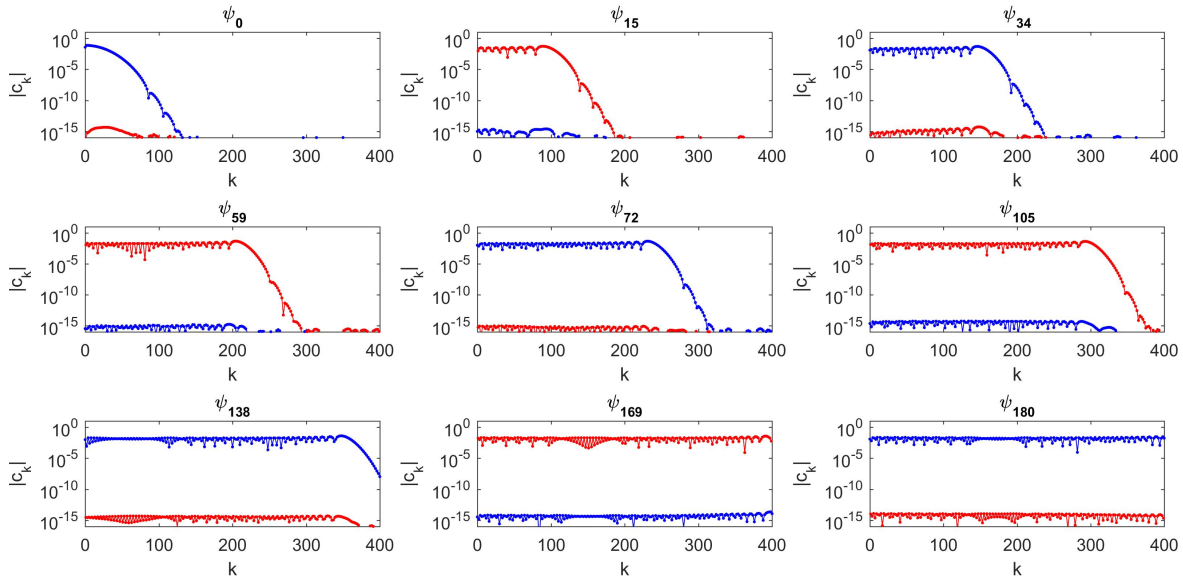


Figure 39: Chebyshev coefficients for a few of the quartic potential wave functions for $\epsilon = 0.01$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

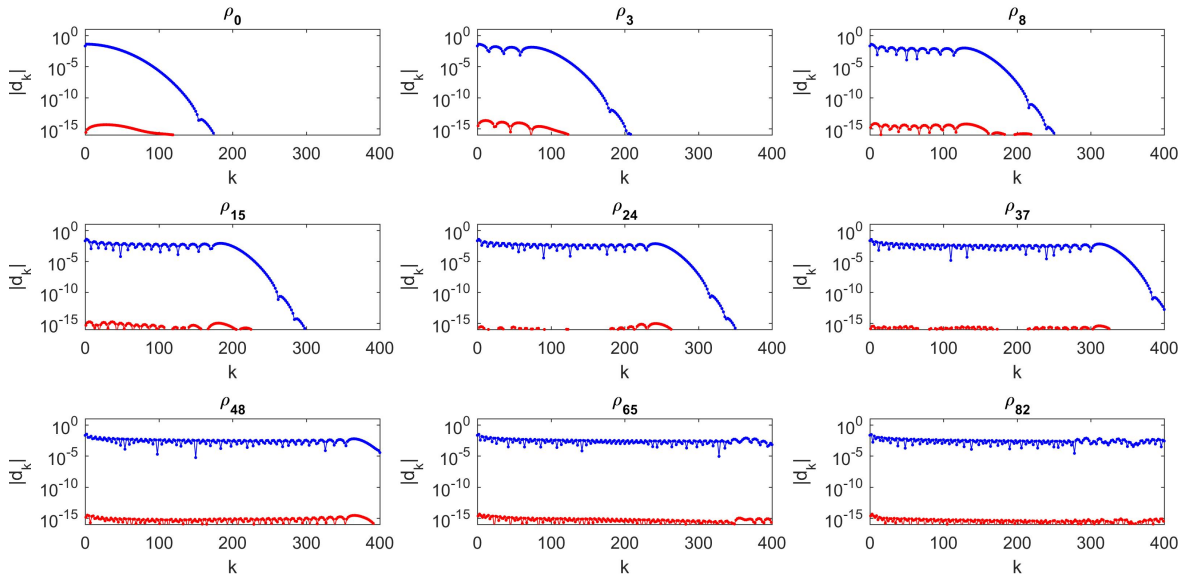


Figure 40: Chebyshev coefficients for the square of a few of the quartic potential wave functions for $\epsilon = 0.01$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

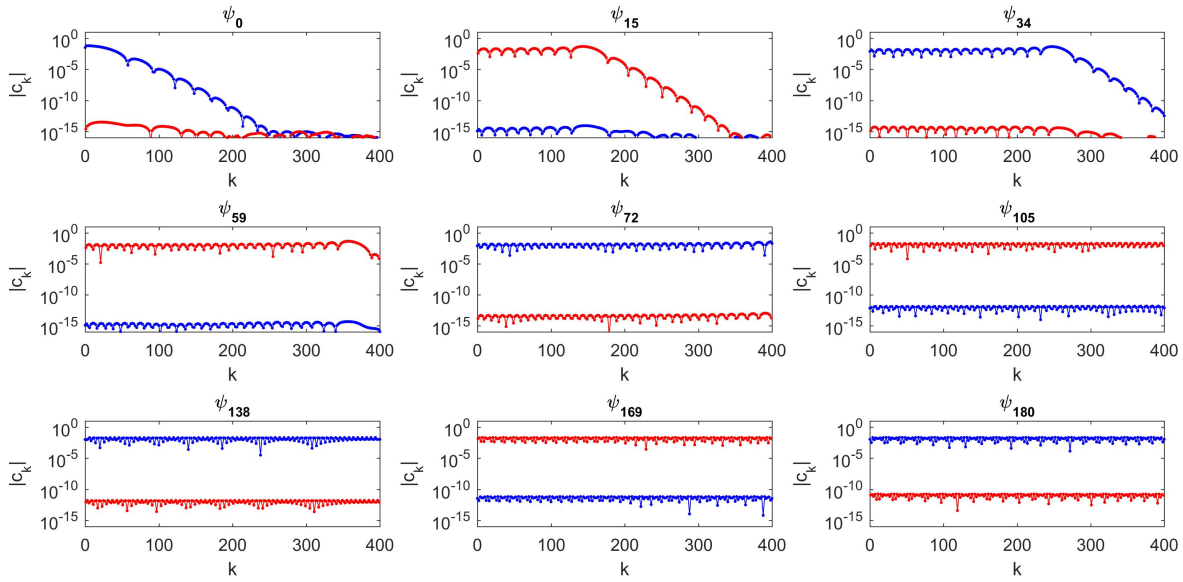


Figure 41: Chebyshev coefficients for a few of the quartic potential wave functions for $\epsilon = 0.5$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

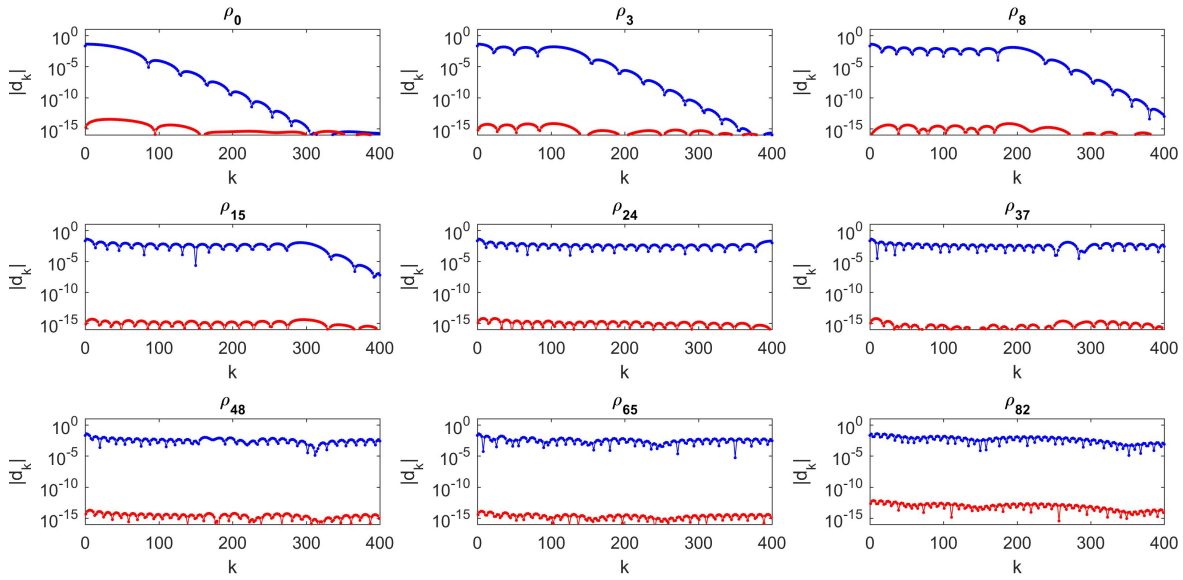


Figure 42: Chebyshev coefficients for the square of a few of the quartic potential wave functions for $\epsilon = 0.5$, $L = 16$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

of Airy's function $\mathrm{Ai}(x)$. To be specific, the numerical problem we will solve is

$$\left(-\frac{d^2}{dx^2}+x\right)\psi_n(x)=\tilde{\lambda}_n\psi_n(x)\ , \tag{285}$$

where have taken the constants in the problem to be

$$m=g=\frac{\hbar^2}{2m}=1\ . \tag{286}$$

Unlike the previous problems we will now be working on the domain $[0,L]$ which can be done by use of a simple domain map. We impose the Dirichlet boundary conditions

$$\psi_n(0)=\psi_n(L)=0\ . \tag{287}$$

The exact eigenvalues are

$$\tilde{\lambda}_n=-z_n\ , \tag{288}$$

where z_n are the zeros of Airy's function . The corresponding wave functions are

$$\psi_n(x)=N_n\mathrm{Ai}(x-z_n)\ , \tag{289}$$

where N_n are normalisation constants. The zeros of Airy's function are not exactly known. The zeros of Airy's function are tabulated in Ref. [18], although, in making a comparison to our numerical solution to Eq. A9 we require more accurate values. We use the results obtained from [41] for the first six eigenvalues quoted to be accurate to 22 digits. Moreover, to make a comparison the normalisation constants have been calculated using Clenshaw-Curtis quadrature. We use this as our "exact" solution with which we compare our Chebyshev pseudo-spectral solution to.

In Figs. 44i and 44ii we show the difference between the first six eigenvalues obtained by the Chebyshev pseudo-spectral method and the "exact" solution. Clearly, the error decreases quickly with N, however, increasing the box size L one again observes that N needs to be larger to achieve the same accuracy for a given eigenfunction. However, this appears to be less severe for this potential. The maximum error of the first six wave functions are shown in Figs. 45i and 45ii. The error increases quickly with the eigenmode number for the small box, but all six wave functions are accurately calculated in the larger box.

The first four wave functions are shown in Fig. 43 for the case $L=16$, $N=72$ they are indistinguishable from the exact solutions. This figure illustrates why the Fourier pseudo-spectral is not the ideal method to solve Eq. A9. It is clearly not a periodic function as the wave functions oscillate in one direction and are exponentially decaying in the other. Artificial periodisation of this function would naturally lead to a discontinuity and as a consequence Gibbs phenomenon will occur. In these plots, we also see why the increase in the box size L didn't require an as large increase in N, as compared with the other potentials, to maintain the same level of accuracy. That is, the error for the first eigenvalue reaches the round-off plateau with $N\sim 40$ for $L=16$ and when the box is doubled in size, N must only be increased to $N\sim 50$. This is related to the fact that these wave functions are not as localised about the origin.

As the linear potential is not symmetric, the Chebyshev series for the wave functions will have both odd and even coefficients. Furthermore, the coefficients of the square of the wave functions will also have both odd and even coefficients. The coefficients of these expansions are shown in Figs. 46, 47, 48 and 49 for the two boxes. The behaviour of these coefficients is very similar to the corresponding ones of the other potentials.

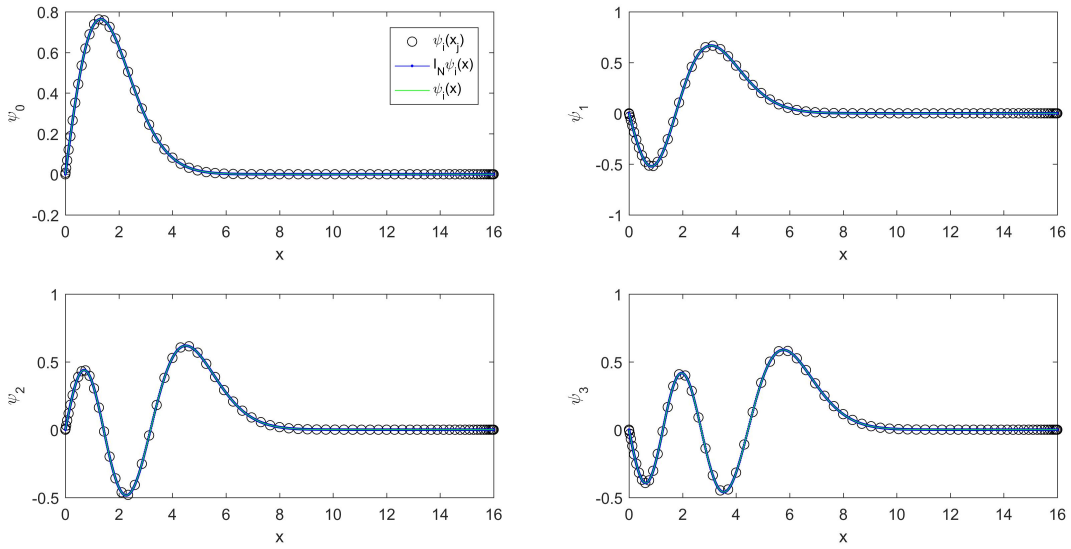
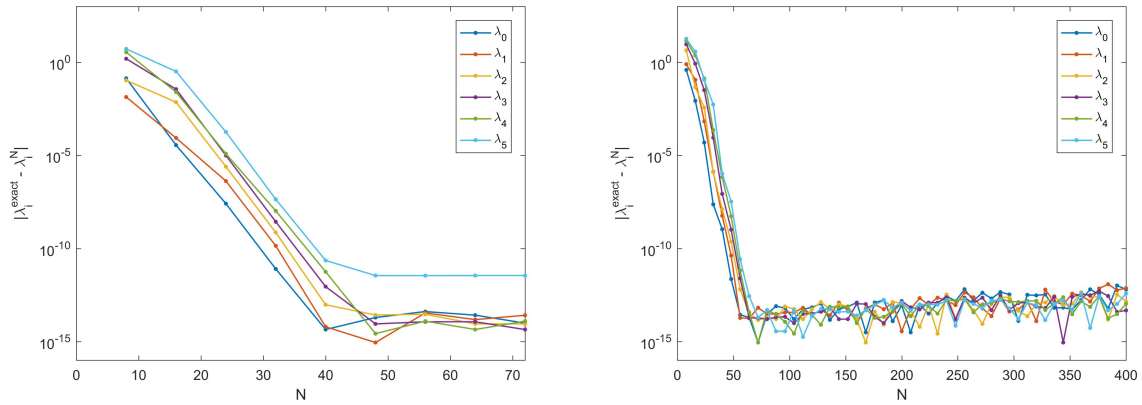


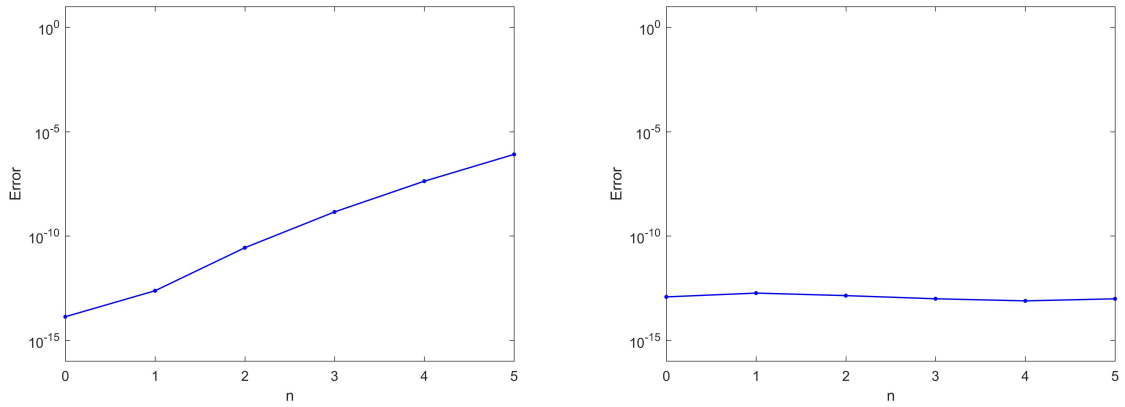
Figure 43: The first four wave functions of the linear potential with $L = 16$, $N = 72$. The circles mark the values at the collocation points, the blue curves are the interpolants calculated using Chebfun's `bary()` *function and the green curves are the corresponding "exact" wave functons. The corresponding plots for $L = 32$, $N = 400$ are essentially indistinguishable from these. These plots are produced by modifying the MATLAB® script* `Ex_Script_15 _Cheby_Linear.m`.



(i) Error in the first 6 eigenvalues for $L = 16$ and $N = 72$.

(ii) Error in the first 6 eigenvalues for $L = 32$ and $N = 400$.

Figure 44: Error in the linear potential eigenvalues found using the Chebyshev pseudospectral method. These plots are produced by modifying the MATLAB® script `Ex_Script_15 _Cheby_Linear.m`.



(i) Error in the wave functions for $L = 16$ and $N = 72$.

(ii) Error in the wave functions for $L = 32$ and $N = 400$.

Figure 45: The error in the linear potential wave functions using the Chebyshev pseudo-spectral method. These plots are produced by modifying the MATLAB® script ***Ex_Script_15 _Cheby_Linear.m****.*

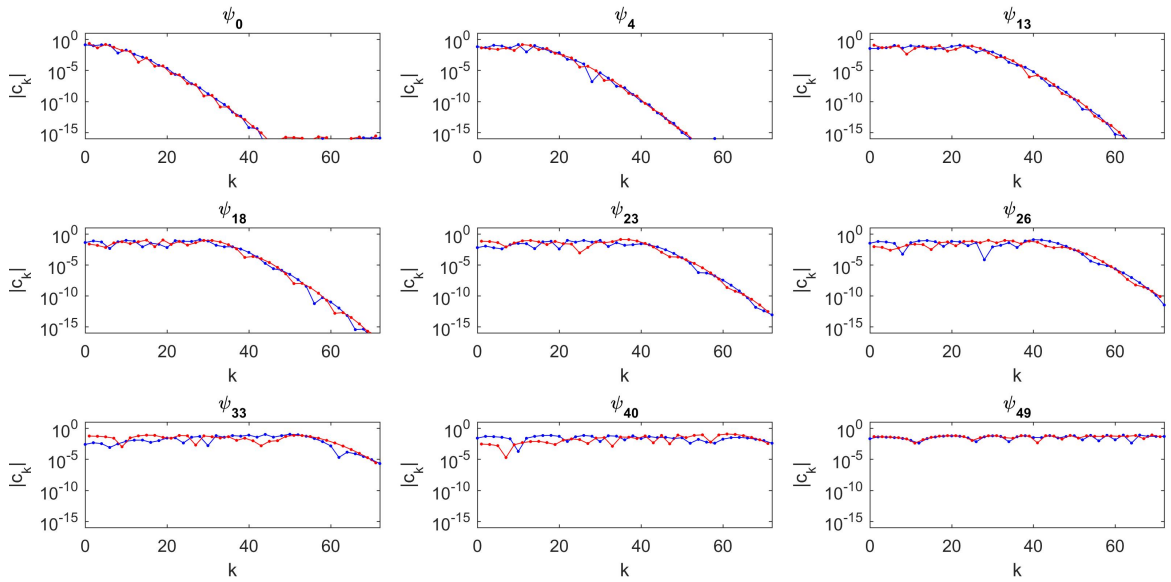


Figure 46: Chebyshev coefficients for a few of the linear potential wave functions coefficients for $L = 16$ and $N = 72$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script ***Ex_Script_15 _Cheby_Linear.m****.*

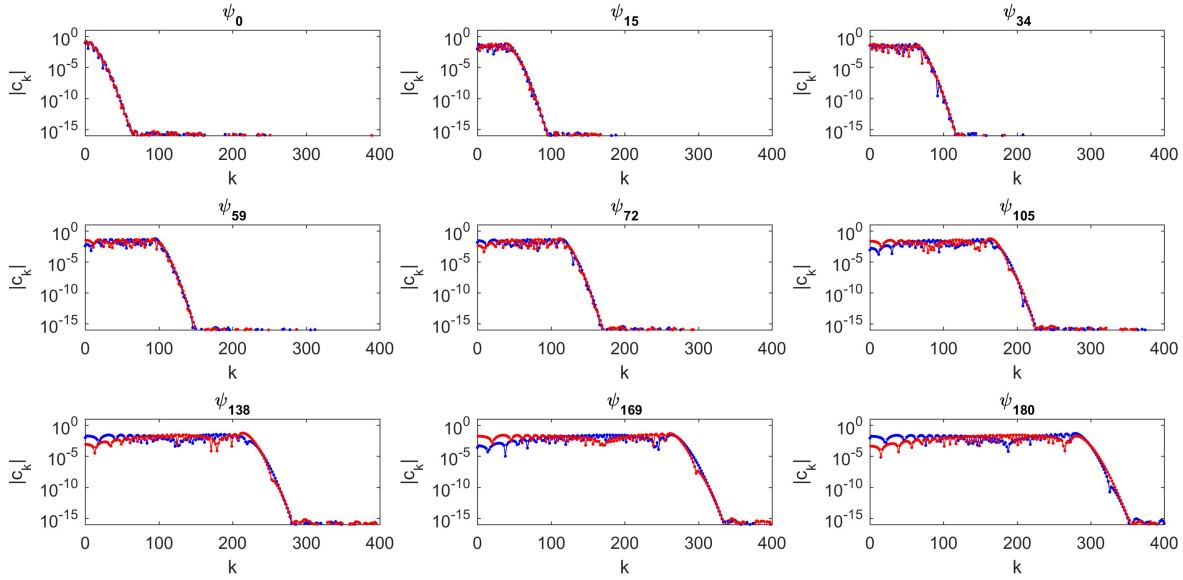


Figure 47: Chebyshev coefficients for a few of the linear potential wave functions coefficients for $L = 32$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_15 _Cheby_Linear.m`.

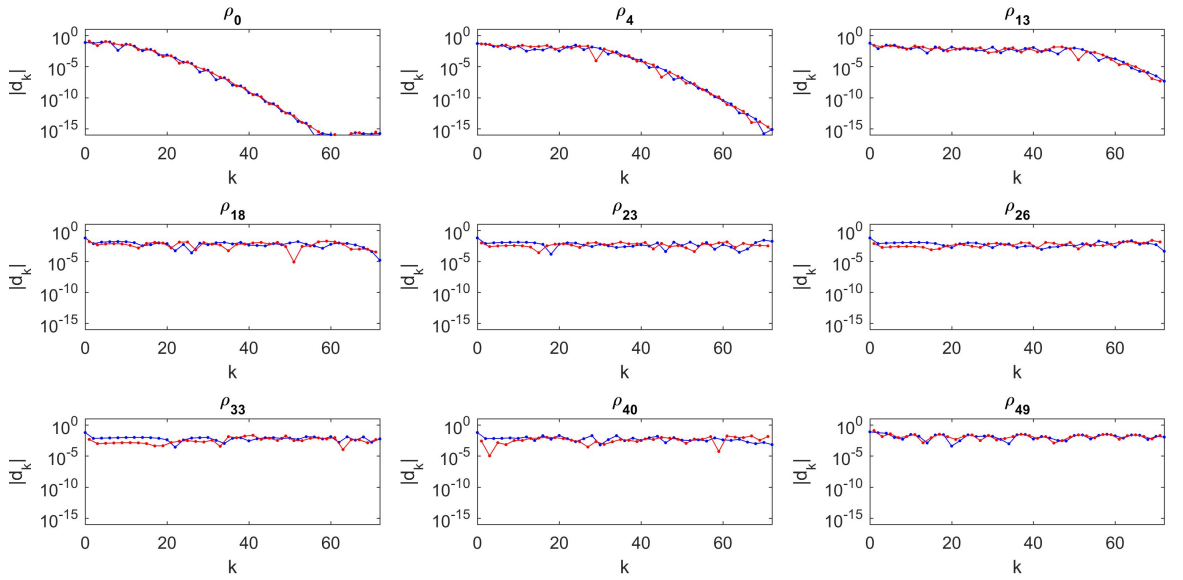


Figure 48: Chebyshev coefficients of the square of a few of the wave functions for $L = 16$ and $N = 72$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_15 _Cheby_Linear.m`.

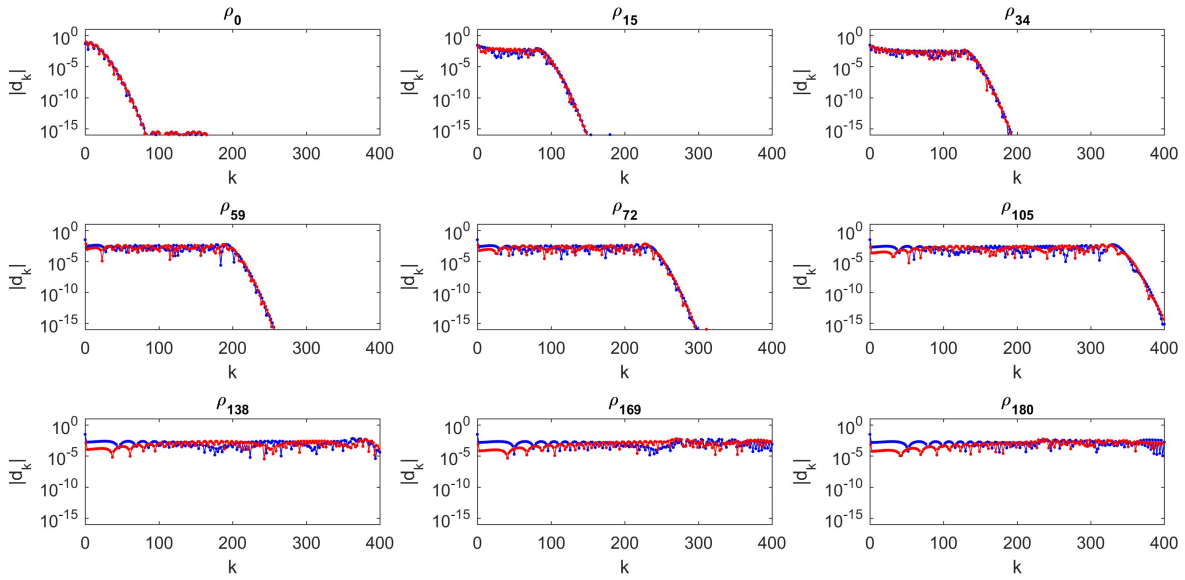


Figure 49: Chebyshev coefficients of the square of a few of the wave functions for $L = 32$ and $N = 400$. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_15 _Cheby_Linear.m`.

8.2 Time-dependent Schrödinger equation examples.

In Section 2 we introduced the time-dependent Schrödinger equation. We discussed that when the potential in the Hamiltonian is independent of time, the problem of solving this equation can be reduced to solving the time-independent Schrödinger equation. A general solution to the equation is just a linear superposition of the wave functions of the time-independent equation and their evolution in the potential is controlled by complex exponentials.

We have solved the time-dependent equation as described above for two potentials using the Chebyshev pseudo-spectral method with $L = 16$ and $N = 400$. The potentials we have chosen are the harmonic oscillator and the quartic potential with small coupling constant, $\epsilon = 0.001$. For both, we project the initial Gaussian state onto a truncated set of eigenfunctions. Our set of eigenfunctions is truncated by only accepting the eigenfunctions with mode numbers $j = 0, 1, \ldots, M/2$, where M is the eigenmode number of the last wave function whose last 10 Chebyshev coefficients have an absolute value less than 10^{-10}.

The initial state is taken to be

$$f(x) = \frac{1}{(\pi\sigma^2)^{1/4}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{290}$$

with $\sigma = 0.5$ and $\mu = -5$ or $\mu = 0$. The projection of this state onto our subset of eigenfunctions $\{\psi_j\}_{j=0}^{M/2}$ is then obtained. The expansion coefficients from this projection are calculated as

$$A_n \simeq \frac{\int_{-L}^{L} f(x)\psi_n(x)dx}{\int_{-L}^{L} |\psi_n(x)|^2 dx} \simeq \frac{\sum_{j=0}^{N} f(x_j)\psi_n(x_j)w_j}{\sum_{j=0}^{N} |\psi_n(x_j)|^2 w_j} = \sum_{j=0}^{N} f(x_j)\psi_n(x_j)w_j \ , \tag{291}$$

where the the $N+1$ numbers x_j are the nodes and the $N+1$ numbers w_j are the weights of Clenshaw-Curtis quadrature scaled for use on the interval $[-L, L]$. The behaviour of these coefficients is shown in Figs. 51 and 52. They appear to exhibit a geometric rate of convergence. However, for the initial state with $\mu = -5$, the coefficients tend to increase first before they start to decrease. Moreover, we see that because the initial state is not located at the origin, both the odd and even coefficients are non-zero as one would expect.

The initial state can be reconstructed using these coefficients by

$$\Psi(x,0) = \sum_{j=0}^{M/2} A_j \psi_j(x) \tag{292}$$

where $\psi_j(x)$ is the barycentric interpolant of the calculated jth wave function.

In Figs. 50i and 50ii we show the initial Gaussian state with $\mu = -5$ (green) which is obscured from view by the reconstructed initial state (blue). The circles indicate the values of the reconstructed wave function at the collocation points. Visually, the initial state appears to be well reconstructed for both potentials.

In the MATLAB® scripts `Ex_Script_13_Cheby_HO.m` and `Ex_Script_14_Cheby_AHO.m` we create short movies of the time evolution of the probability density

$$\rho(x,t) = \Psi^*(x,t)\Psi(x,t) \tag{293}$$

where

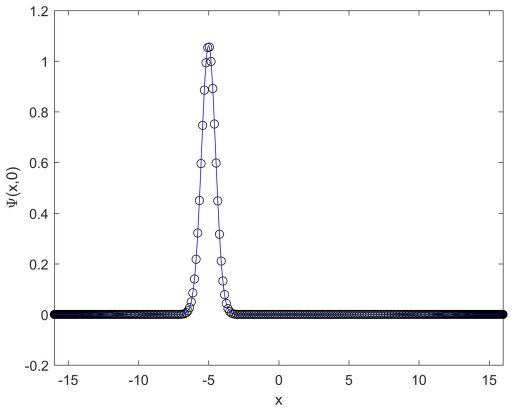
$$\Psi(x,t) = \sum_{j=0}^{M/2} A_j \psi_j(x) e^{-i\tilde{\lambda}_j t} \ . \tag{294}$$

In Fig. 53 and 54 we show the time evolution of the probability densities corresponding to the initial state with $\mu = -5$. The different coloured curves correspond to different instances in time. For the harmonic potential, the probability density remains Gaussian and the centre oscillates back and forth between $x = -5$ and $x = 5$ with its width also varying in time. This behaviour of the probability density can be derived analytically for the harmonic oscillator potential. In contrast, in the quartic potential the probability density starts out as Gaussian, then becomes non-Gaussian, eventually having multiple peaks. We have found that this occurs faster as ϵ is increased.

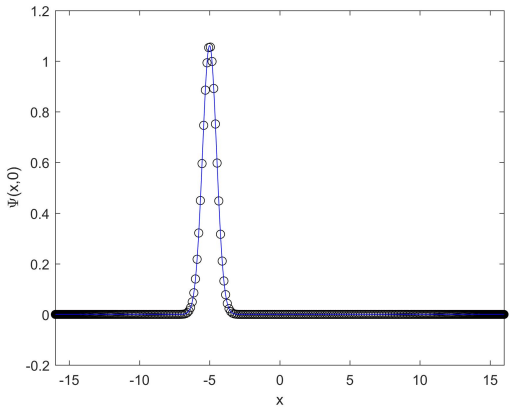
9 Discussion, conclusions and future work

In this report we focussed on solving the Schrödinger equation for several simple potentials using pseudo-spectral methods. The motivation for being that the ability to solve the Schrödinger equation efficiently and accurately will be an important part of constructing a Bohm filter [1].

As we have tried to emphasise, collocation interpolation is the bedrock of the approach we have taken. By this we mean it underlies the representation of functions, their numerical integration and differentiation and hence also the way in which we solve the eigenvalue problem that is

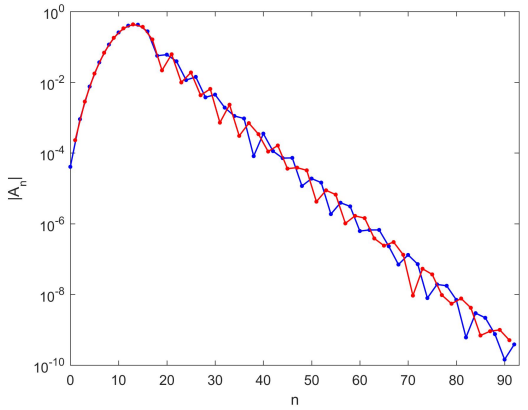


(i) Harmonic oscillator potential.

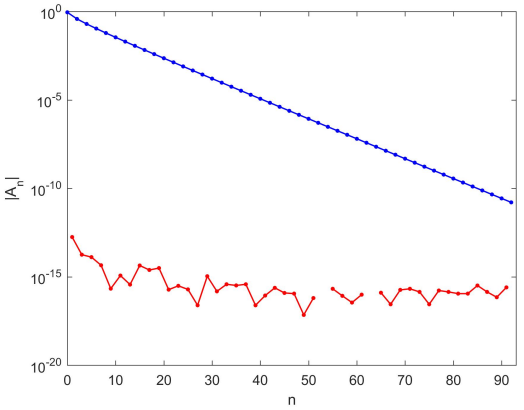


(ii) Quartic potential with $\epsilon = 0.001$.

Figure 50: The circles mark the values at the collocation points, the obscured green curve is the initial Gaussian state wave function and the blue curve is the reconstructed initial state after being projected onto the truncated set of wave functions. These plots are produced by modifying the MATLAB® scripts `Ex_Script_13_Cheby_HO.m` *and* `Ex_Script_14_Cheby_Quartic.m`.

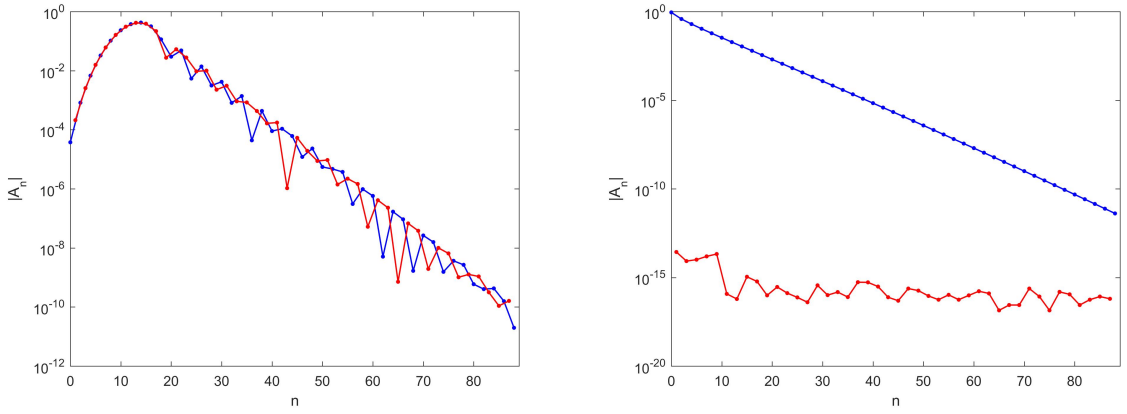


(i) $\mu = -5$



(ii) $\mu = 0$.

Figure 51: Coefficients A_n *for the truncated harmonic oscillator wave function expansion. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script* `Ex_Script_13_Cheby_HO.m`.



(i) $\mu = -5$ *(ii)* $\mu = 0$

Figure 52: Coefficients A_n for the truncated anharmonic oscillator wave function expansion. The odd coefficients are in red and the even ones are in blue. These plots are produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

the time-independent Schödinger equation. This approach has been chosen because of the convergence properties of pseudo-spectral methods for smooth functions [10–16].

We have taken a long path to illustrate one way of solving the time-dependent Schrödinger equation. At several points we could have made different decisions, leading to different but related approaches. We wish to highlight two of the crossroads where our path could have diverged.

- *The reduction to the time-independent Schrödinger equation.*
  The eventual application of the Bohm filter to a realistic tracking problem will require one to solve the time-dependent Schrödinger equation for a potential which may or may not be time dependent. In the special case that potential is independent of time, one can reduce the problem to the time independent equation and the time evolution can be done exactly. Upon this reduction one must then solve an eigenvalue problem. As we have seen obtaining many eigenvalues and eigenfunctions can be done using the methods discussed. However, to get many accurate eigenfunctions one must increase the number of collocation points and hence the size of the matrix eigenvalue problem that must be solved. Extending to two or three dimensions will require the solution of even larger matrix eigenvalue problems.

  For time-dependent potentials we cannot make such a reduction. To solve time-dependent partial differential equations a typical approach to take is to treat the spatial part of the equation spectrally and the temporal part using finite differences. The reason for the different treatment for time derivatives is that there appears not to be an efficient spectral decomposition in time. The use of finite differencing schemes would naturally reduce the exponential rate of convergence achievable with spectral methods. However, it is cheaper to make the time step smaller than the spatial step, particularly when working with multiple spatial dimensions [10]. This approach can of course be applied to the special case of time independent potentials, removing the need to find many eigenfunctions. However, the time evolution will no longer be exact.

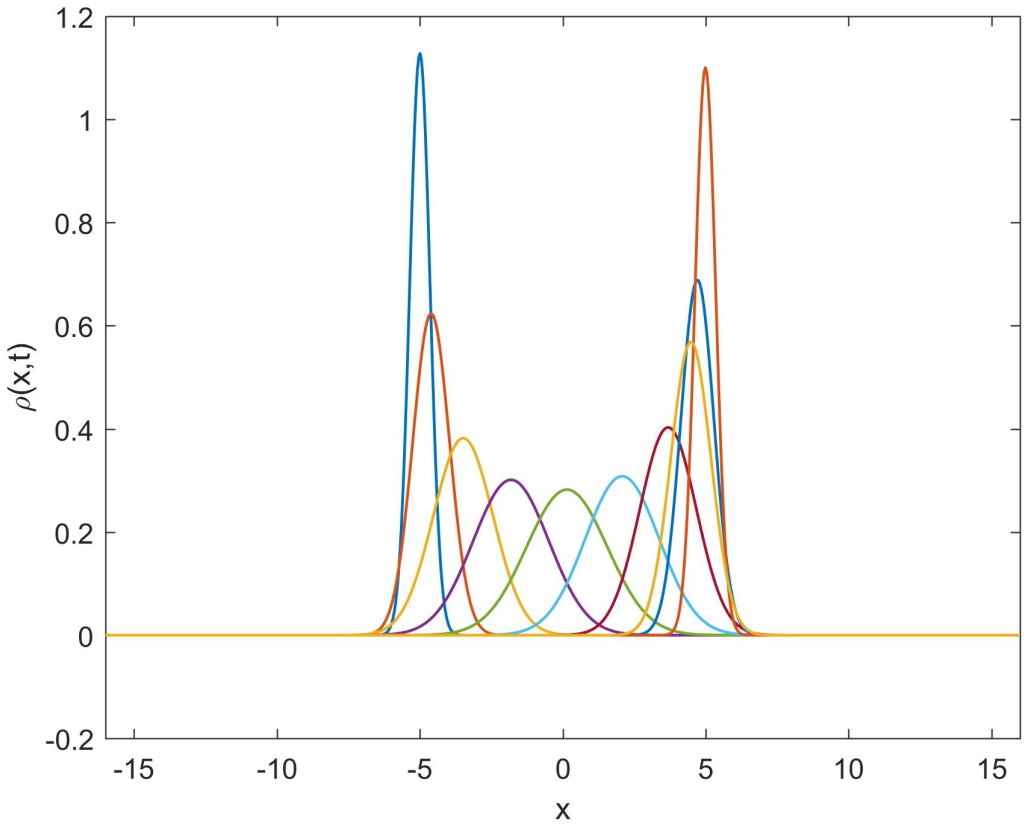


Figure 53: The curves represent the evolution of the probability density in the harmonic oscillator potential obtained by evolving the projected initial Gaussian state. The different colours correspond to different instances of time and the time step is the same as in Fig. 54. This plot is produced by modifying the MATLAB® script `Ex_Script_13_Cheby_HO.m`.

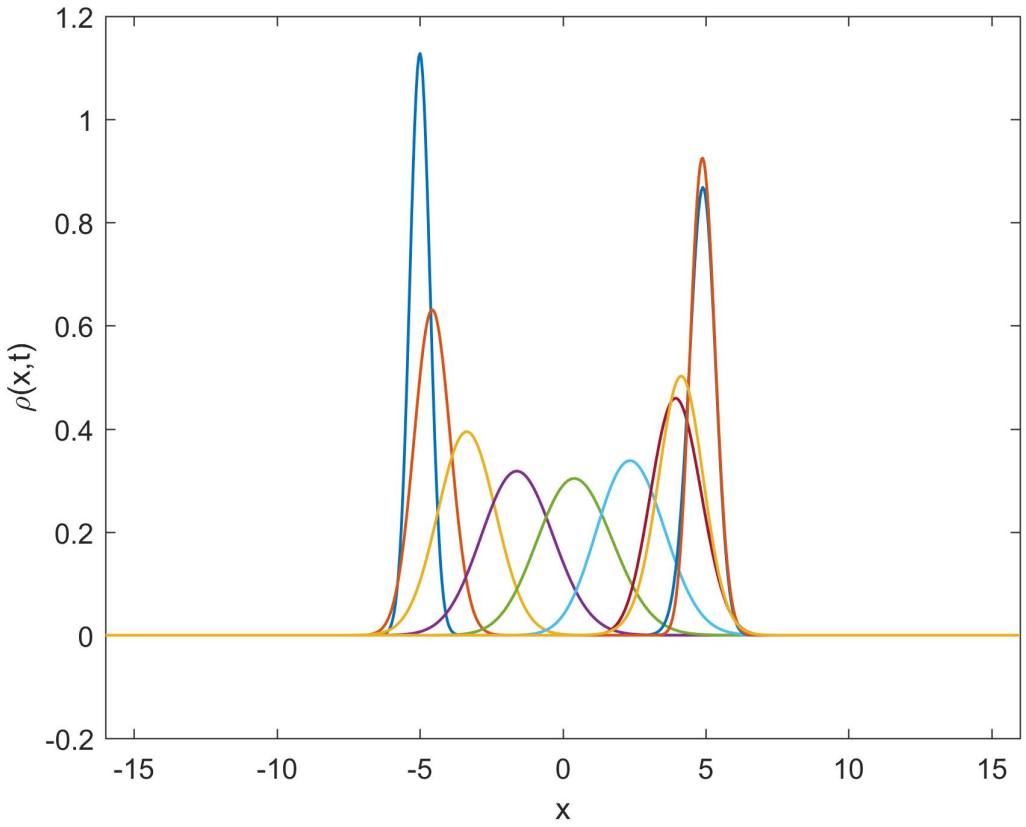


Figure 54: The curves represent the evolution of the probability density in the quartic potential with $\epsilon = 0.001$ obtained by evolving the projected initial Gaussian state. The different colours correspond to different instances of time and the time step is the same as in Fig. 53. This plot is produced by modifying the MATLAB® script `Ex_Script_14_Cheby_Quartic.m`.

- *The choice to work in configuration space rather than coefficient space.*
  As we have stated in this report, it is possible to work in either space. In different situations one may be more preferable than the other. We have chosen to work mostly in configuration space for simplicity rather than efficiency since this is an initial investigation. Also, in applying the Bohm filter to realistic tracking problems we do not know ahead of time which set of trial functions will be best. This would generally be determined by the geometry of the tracking problem and the nature of the potential. We have mentioned five bases, the Fourier, Chebyshev, Legendre, Laguerre and Hermite. The first two of which we have used in this report. These two are known to have fast transforms which allow for efficient methods to solve partial differential equations in coefficient space, i.e., the fast Fourier and cosine transforms. Efficient implementations with the other bases in coefficient space will naturally be more difficult and specialised.

There are many possibilities for future work. The most pressing things to consider would be:

- An application of the Bohm filter to toy tracking problems based on the potentials investigated in this report. For example, a Bohm filter based on the harmonic oscillator potential could be used to track a pendulum whose measurements are polluted with various forms of noise, e.g., Von Mises and other non-Gaussian forms of noise. Comparisons with conventional filters such as the Kalman and extended Kalman filters should also be carried out.
- Investigate the application of pseudo-spectral methods using other bases, e.g., Legendre, Laguerre and Hermite polynomials, to the Schrödinger equation.
- Extension to non-polynomial potentials.
- Extension to two and three dimensions.
- Extension to time dependent potentials by the method discussed above.
- Derivation of potentials for real world tracking problems and investigate their numerical solution.
- Determine whether there is a theoretical connection between the Bohm filter and the stochastic variational method [2–8].
- Consider evolution equations other than the Schrödinger equation.

10 References

1. Drake, S. P. *The Bohm Filter and its Application to Bearings Only Tracking.* Private notes.
2. Yasue, K. 'Stochastic calculus of variations'. In: *Journal of Functional Analysis* **41** (3), 327 –340. ISSN: 0022-1236. DOI: http://dx.doi.org/10.1016/0022-1236(81)90079-3. URL: http://www.sciencedirect.com/science/article/pii/0022123681900793.
3. Koide, T., Kodama, T. and Tsushima, K. 'Unified Description of Classical and Quantum Behaviours in a Variational Principle'. In: *J. Phys. Conf. Ser.* **626** (1), 012055. DOI: 10.1088/1742-6596/626/1/012055. arXiv: 1412.5865 [quant-ph].

4. Koide, T., Kodama, T. and Tsushima, K. 'Stochastic Variational Method as a Quantization Scheme II: Quantization of Electromagnetic Fields'. In: *ArXiv:hep-th/1406.6295*. arXiv: 1406.6295 [hep-th].

5. Kodama, T. and Koide, T. 'Stochastic variational quantization and maximum entropy principle'. In: *Phys. Part. Nucl.* **46** (5), 768–771. DOI: 10.1134/S1063779615050135.

6. Koide, T., Tsushima, K. and Kodama, T. 'Schroedinger Equation in Rotating Frame by using Stochastic Variational Method'. In: *31st International Colloquium on Group Theoretical Methods in Physics (GROUP 31) Rio de Janeiro, RJ, Brazil, June 20-24, 2016*. arXiv: 1611.07570 [math-ph]. URL: https://inspirehep.net/record/1592485/files/arxiv:1611.07570.pdf.

7. Koide, T. and Kodama, T. 'Navier-Stokes, Gross-Pitaevskii and Generalized Diffusion Equations using Stochastic Variational Method'. In: *J. Phys.* **A45**, 255204. DOI: 10.1088/1751-8113/45/25/255204. arXiv: 1108.0124 [cond-mat.stat-mech].

8. Koide, T. and Kodama, T. 'Stochastic variational method as quantization scheme: Field quantization of the complex Klein Gordon equation'. In: *PTEP* **2015** (9), 093A03. DOI: 10.1093/ptep/ptv127. arXiv: 1306.6922 [hep-th].

9. MATLAB. *Version 9.0.0 (R2016a)*. The MathWorks Inc. Natick, Massachusetts.

10. Trefethen, L. N. *Spectral methods in MATLAB*. SIAM.

11. Trefethen, L. N. *Approximation Theory and Approximation Practice*. Society for Industrial and Applied Mathematics.

12. Boyd, J. P. *Chebyshev and Fourier Spectral Methods: Second Revised Edition (Dover Books on Mathematics)*. Dover Publications.

13. Funaro, D. *Polynomial approximation of differential equations*. Springer-Verlag.

14. Canuto, C. et al. *Spectral Methods: Evolution to complex geometries and applications to fluid dynamics*. Springer-Verlag Berlin Heidelberg.

15. Canuto, C. et al. *Spectral Metods in Fluid Dynamics*. Springer-Verlag Berlin Heidelberg.

16. Fornberg, B. *A practical guide to pseudospectral methods*. Cambridge University Press.

17. Shen, J., Tang, T. and Wang, L.-L. *Spectral Methods: Algorithms, Analysis and Applications*. Springer Berlin Heidelberg. Berlin, Heidelberg, 47–140. ISBN: 978-3-540-71041-7. DOI: 10.1007/978-3-540-71041-7_3. URL: http://dx.doi.org/10.1007/978-3-540-71041-7_3.

18. Abramowitz, M. and Stegun, I. A. *Handbook of mathematical functions with formulas, graphs and mathematical tables*. Dover Publications.

19. Powell, J. L. and Crasemann, B. *Quantum Mechanics*. Dover Publications.

20. Bohm, D. *Quantum Theory*. Dover Publications.

21. Bender, C. M. and Wu, T. T. 'Anharmonic oscillator'. In: *Phys. Rev.* **184**, 1231–1260. DOI: 10.1103/PhysRev.184.1231.

22. Bender, C. M. and Wu, T. T. 'Anharmonic oscillator. 2: A Study of perturbation theory in large order'. In: *Phys. Rev.* **D7**, 1620–1636. DOI: 10.1103/PhysRevD.7.1620.

23. Loeffel, J. J. et al. 'Pade approximants and the anharmonic oscillator'. In: *Phys. Lett. B* **30** (9), 656.

24. Stoer, J. and Bulirsch, R. *Introduction to numerical analysis*. Springer-Verlag.

25. Berrut, J. P. and Trefethen, L. N. 'Barycentric Lagrange interpolation'. In: *SIAM Review* **46** (3), 501–517.

26. Weideman, J. A. C. and Reddy, S. C. 'A MATLAB differentiation matrix suite'. In: *ACM Transactions on mathematical software* **26** (4), 465–519.

27. Driscoll, T. A., Hale, N. and Trefethen, L. N., eds. *Chebfun Guide*. Pafnuty Publications, Oxford.

28. Runge, C. 'Uber empirische Funktionen und die Interpolation zwischen aquidistanten Ordinaten'. In: *Zeitschrift fur Mathematik und Physik* **46**, 224–243.

29. Rudin, W. *Principles of mathematical analysis*. Third. McGraw-Hill Education.

30. Rao, S. S. *Applied numerical methods for engineers and scientists*. Prentice Hall.

31. Glaser, A., Liu, X. and Rokhlin, V. 'A fast algorithm for the calculation of the roots of special functions'. In: *SIAM J. Sci. Comput.* **29** (4), 1420–1438.

32. Hale, N. and Townsend, A. 'Fast and accurate computation of Gauss-Legendre and Gauss-Jacobi quadrature nodes and weights'. In: *SIAM J. Sci. Comput.* **35** (2), A652–A674.

33. Bogaert, I. 'Iteration-free computation of Gauss-Legendre quadrature nodes and weights'. In: *SIAM J. Sci. Comput.* **36** (3), A1008–A1026.

34. Clenshaw, C. W. and Curtis, A. R. 'A method for numerical integration on an automatic computer'. In: *Numerische Mathematik* **2**, 197–205.

35. Trefethen, L. N. 'Is Gauss quadrature better than Clenshaw-Curtis?' In: *SIAM Review* **50** (1), 67–87.

36. Vanlessen, M. 'Strong asymptotics of Laguerre-Type orthogonal polynomials and applications in Random Matrix Theory'. In: *Constr. Approx.,* **25:** 125–175.

37. Townsend, A., Trogdon, T. and Olver, S. 'Fast computation of Gauss quadrature nodes and weights on the whole real line'. In: *IMA Journal of Numerical Analysis* **36**, 337–358.

38. Golub, G. H. 'Some modified matrix eigenvalue problems'. In: *SIAM Rev.* **15** (2), 318–334.

39. Trefethen, L. N. and Weideman, J. A. C. 'The Exponentially Convergent Trapezoidal Rule'. In: *SIAM Review* **56** (3), 385 –458.

40. Golub, G. G. and Welsch, J. H. 'Calculation of Gauss quadrature rules'. In: *Mathematics of computation* **23** (106), 221–230.

41. Casio. *Keisan online calculator, Airy function zeros, http://keisan.casio.com/exec/system/1180573400.* URL: http://keisan.casio.com/exec/system/1180573400.

Appendix A: Analytical solutions for toy potentials

A.1 Harmonic oscillator potential

The time-independent Schrödinger equation with harmonic oscillator potential in one dimension has the form

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}+\frac{1}{2}m\omega^2x^2\right)\psi(x)=E\psi(x) \tag{A1}$$

In the problem definition there are three parameters, m, ω and $\hbar$. These can be used to construct a length scale to make the TISE dimensionless. Using these parameters we perform the following change of variables

$$x=\sqrt{\frac{\hbar}{m\omega}}\xi \quad , \quad dx=\sqrt{\frac{\hbar}{m\omega}}d\xi \tag{A2}$$

Using this change of variables the Schrödinger equation

$$\frac{d^2}{dx^2}\psi(x)-\left(\frac{m^2\omega^2x^2}{\hbar^2}-\frac{2mE}{\hbar^2}\right)\psi(x)=0\ , \tag{A3}$$

therefore becomes

$$\frac{d^2\psi(\xi)}{d\xi^2}.\frac{m\omega}{\hbar}-\left(\frac{m^2\omega^2}{\hbar^2}\left(\sqrt{\frac{\hbar}{m\omega}}\right)^2\xi^2-\frac{2mE}{\hbar^2}\right)\psi(\xi)=0\ . \tag{A4}$$

After simplifying one obtains

$$\frac{d^2\psi(\xi)}{d\xi^2}-\left(\xi^2-\frac{2E}{\hbar\omega}\right)\psi(\xi)=0 \quad . \tag{A5}$$

This is one form of Hermite's equation. It can be solved exactly, the solution being the following set of eigenvalues

$$E=E_n=\hbar\omega(n+\frac{1}{2}) \quad , \quad n=0,1,2,\ldots \tag{A6}$$

and to each of these eigenvalues there is an associated eigenfunction, they are

$$\psi(x)=\psi_n(x)=\frac{\sqrt[4]{m\omega/\hbar}}{\sqrt{2^n n!\sqrt{\pi}}}e^{-\frac{m\omega x^2}{2\hbar}}H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right) \tag{A7}$$

or in terms of the dimensionless variable ξ we have

$$\psi_n(\xi)=\frac{1}{\sqrt{2^n n!\sqrt{\pi}}}e^{-\xi^2/2}H_n(\xi) \quad . \tag{A8}$$

To derive this solution begin by looking at the asymptotic behaviour as $\xi\to\pm\infty$ and use the power series (Frobenius) method. This is derived in any good quantum mechanics text, see for example Ref. [19].

A.2 Linear potential

The time-independent Schrödinger equation with a linear potential in one dimension has the form

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}+V(x)\right)\psi(x)=E\psi(x)\ , \tag{A9}$$

where the potential is

$$V(x)=\begin{cases}\infty & , \quad x<0\\ mgx & , \quad x>0\end{cases}\ . \tag{A10}$$

Obviously, the probability to find a particle in this potential at $x<0$ is 0, therefore we have the boundary condition $\psi(0)=0$. In the problem definition there are three parameters, m, g and $\hbar$. These can be used to construct a natural length and energy scale to make the TISE dimensionless. These are

$$\ell\equiv\left(\frac{\hbar^2}{2m^2g}\right)^{1/3}\equiv k^{-1} \tag{A11}$$

and

$$\mathcal{E}\equiv mg\ell=\left(\frac{mg^2\hbar^2}{2}\right)^{1/3}\ , \tag{A12}$$

respectively.

Using the natural length scale of the problem we can write Eq. A9 as

$$\begin{aligned}\frac{d^2\psi(x)}{dx^2} &= \left(\frac{2m^2gx}{\hbar^2}-\frac{2mE}{\hbar^2}\right)\psi(x)\\ &= k^3\left(x-\frac{E}{mg}\right)\psi(x)\ .\end{aligned} \tag{A13}$$

We now perform the following change of variables

$$y=k\left(x-\frac{E}{mg}\right) \tag{A14}$$

and note that upon this change the normalisation condition tells us

$$\int_0^\infty|\psi(x)|^2dx = \int_0^\infty|\psi(x(y))|^2\left|\frac{dx}{dy}\right|dy \tag{A15}$$

$$= \int_0^\infty|\psi(x(y))|^2\ell\,dy \tag{A16}$$

$$=: \int_0^\infty|\tilde{\psi}(y)|^2\,dy\ , \tag{A17}$$

$$\tag{A18}$$

where we have defined

$$\tilde{\psi}(y)=\sqrt{\ell}\psi\left(\ell(y+\frac{E}{\mathcal{E}})\right) \tag{A19}$$

or equivalently

$$\psi(x)=\sqrt{k}\tilde{\psi}(y)\ . \tag{A20}$$

Now the LHS of Schrödinger equation can be rewritten as

$$\frac{d^2\psi(x)}{dx^2} = k^2\sqrt{k}\frac{d^2}{dy^2}\tilde{\psi}(y) \tag{A21}$$

and the RHS as

$$k^3\left(x - \frac{E}{mg}\right)\psi(x) = k^2\sqrt{k}y\tilde{\psi}(y) \tag{A22}$$

allowing one to write the Schrödinger equation as

$$\frac{d^2}{dy^2}\tilde{\psi}(y) = y\tilde{\psi}(y)\ . \tag{A23}$$

This is nothing other than Airy's equation which has the known solutions $\mathrm{Ai}(y)$ and $\mathrm{Bi}(y)$, the latter blows up as $y \to \infty$. It is important to note that we have absorbed the eigenvalues E_n into the definition of y

$$\tilde{\psi}_n(y) = N_n\mathrm{Ai}(y) = N_n\mathrm{Ai}\left(k(x - \frac{E_n}{mg})\right)\ , \tag{A24}$$

where N_n is a normalisation constant. To enforce the boundary condition at the origin ($x = 0$) we must have

$$N_n\mathrm{Ai}\left(-\frac{kE_n}{mg}\right) = 0\ , \tag{A25}$$

which tells us the eigenvalues are simply

$$E_n = -mgz_n\ell\ , \tag{A26}$$

where z_n denotes the zeros of Airy's function. There are approximate analytical solutions for these zeros, but as we are trying to make an accurate comparison with our numerical calculation we use known numerical solutions to Eq. A25 for the first six eigenvalues obtained from [41] quoted to be accurate to 22 digits. In Section 8 we also evaluate the normalisation constant numerically using Clenshaw-Curtis quadrature.

Appendix B: Classical orthogonal polynomials

The properties of a few classical orthogonal polynomials are summarised. For further information consult the standard reference [18].

B.1 Chebyshev polynomials

Domain:

$$[-1, 1] \tag{B1}$$

Weight:

$$w(x) = (1 - x^2)^{-1/2} \tag{B2}$$

Normalisation:

$$h_n = \begin{cases} \pi/2 & n \neq 0 \\ \pi & n = 0 \end{cases} \tag{B3}$$

Standardisation:

$$T(1) = 1 \tag{B4}$$

Leading term:

$$k_n = 2^{n-1} \tag{B5}$$

Explicit expressions:

$$T_n(x) = \frac{n}{2} \sum_{m=0}^{\left[\frac{n}{2}\right]} \frac{(-1)^m (n-m-1)!}{m!(n-2m)!} . (2x)^{n-2m} \tag{B6}$$

and the simpler

$$T_n(x) = \cos(n \arccos(x)) \ . \tag{B7}$$

Special values:

$$T_n(x) = (-1)^n T_n(-x) \tag{B8}$$

$$T_n(1) = 1 \tag{B9}$$

$$T_n(0) = \begin{cases} (-1)^m & , \quad n = 2m \\ 0 & , \quad n = 2m+1 \end{cases} \tag{B10}$$

Three-term recurrence relation:

$$T_0(x) = 1 \tag{B11}$$

$$T_1(x) = x \tag{B12}$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{B13}$$

Derivative:

$$(1-x^2)T_n'(x) = -nxT_n(x) + nT_{n-1}(x) \tag{B14}$$

and using the three-term recurrence relation one obtains

$$(1-x^2)T_n'(x) = \frac{n}{2}\left(T_{n-1}(x) - T_{n+1}(x)\right) \tag{B15}$$

Differential equation:

$$(1-x^2)y'' - xy' + n^2 y = 0 \tag{B16}$$

Product formula:

$$2T_n(x)T_m(x) = T_{n+m}(x) + T_{n-m}(x) \quad , \quad n \geq m \tag{B17}$$

Inequalities:

$$|T_n(x)| \leq 1 \tag{B18}$$

$$\left|\frac{dT_n(x)}{dx}\right| \leq n^2 \tag{B19}$$

Rodrigues formula:

$$T_n(x) = \frac{\sqrt{\pi}}{(-1)^n 2^n \Gamma(n+\frac{1}{2})(1-x^2)^{-1/2}} \frac{d^n}{dx^n}\left[(1-x^2)^{-1/2}(1-x^2)^n\right] \tag{B20}$$

B.2 Legendre polynomials

Domain:

$$[-1, 1] \tag{B21}$$

Weight:

$$w(x) = 1 \tag{B22}$$

Normalisation:

$$h_n = \frac{2}{2n+1} \tag{B23}$$

Standardisation:

$$P_n(1) = 1 \tag{B24}$$

Leading term:

$$k_n = \frac{(2n)!}{2^n (n!)^2} \tag{B25}$$

Explicit expression:

$$P_n(x) = \frac{1}{2^n} \sum_{m=0}^{\left[\frac{n}{2}\right]} (-1)^m \left(\begin{array}{c} n \\ m \end{array} \right) \left(\begin{array}{c} 2n-2m \\ n \end{array} \right) x^{n-2m} \tag{B26}$$

where

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{r!(n-r)!} \tag{B27}$$

is the binomial coefficient.

Special values:

$$P_n(x) = (-1)^n P_n(-x) \tag{B28}$$

$$P_n(1) = 1 \tag{B29}$$

$$P_n(0) = \begin{cases} \frac{(-1)^m}{4^m} \left(\begin{array}{c} 2m \\ m \end{array} \right) & , \quad n = 2m \\ 0 & , \quad n = 2m+1 \end{cases} \tag{B30}$$

Three-term recurrence relation:

$$P_0(x) = 1 \tag{B31}$$

$$P_1(x) = x \tag{B32}$$

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \tag{B33}$$

Derivative:

$$(1-x^2)P_n'(x) = -nxP_n(x) + nP_{n-1}(x) \tag{B34}$$

Differential Equation:

$$(1-x^2)y'' - 2xy' + n(n+1)y = 0 \tag{B35}$$

Inequalities:

$$|P_n(x)| \leq 1 \tag{B36}$$

$$\left|\frac{dP_n}{dx}\right| \leq \frac{1}{2}n(n+1) \tag{B37}$$

$$|P_n(x)| \leq \sqrt{\frac{2}{\pi n}}\frac{1}{4\sqrt{1-x^2}} \tag{B38}$$

Rodrigues formula:

$$P_n(x) = \frac{1}{(-1)^n 2^n n!}\frac{d^n}{dx^n}\left[(1-x^2)^n\right] \tag{B39}$$

B.3 Laguerre polynomials

Domain:

$$[0, \infty) \tag{B40}$$

Weight:

$$w(x) = e^{-x} \tag{B41}$$

Normalisation:

$$h_n = 1 \tag{B42}$$

Standardisation and leading term:

$$k_n = \frac{(-1)^n}{n!} \tag{B43}$$

Explicit expression:

$$L_n(x) = \sum_{m=0}^{n}(-1)^m \begin{pmatrix} n \\ n-m \end{pmatrix}\frac{x^m}{m!} \tag{B44}$$

Special values:

$$L_n(0) = \begin{pmatrix} n \\ m \end{pmatrix} \tag{B45}$$

Three-term recurrence relation:

$$L_0(x) = 1 \tag{B46}$$

$$L_1(x) = -x + 1 \tag{B47}$$

$$(n+1)L_n(x) = (2n+1-x)L_n(x) - nL_{n-1}(x) \tag{B48}$$

Derivative:

$$xL_n'(x) = nL_n(x) - nL_{n-1}(x) \tag{B49}$$

Differential Equation:

$$xy'' + (1-x)y' + ny = 0 \tag{B50}$$

Inequalities:

$$|L_n(x)| \leq e^{x/2} \tag{B51}$$

Rodrigues formula:

$$L_n(x) = \frac{1}{n!e^{-x}} \frac{d^n}{dx^n} \left[x^n e^{-x}\right] \tag{B52}$$

B.4 Hermite polynomials

Domain:

$$(-\infty, \infty) \tag{B53}$$

Weight:

$$w(x) = e^{-x^2} \tag{B54}$$

Normalisation:

$$h_n = \sqrt{\pi} 2^n n! \tag{B55}$$

Leading term:

$$k_n = 2^n \tag{B56}$$

Explicit expression:

$$H_n(x) = n! \sum_{m=0}^{\left[\frac{n}{2}\right]} \frac{(-1)^m}{m!(n-2m)!} (2x)^{n-2m} \tag{B57}$$

Special values:

$$H_n(x) = (-1)^n H_n(-x) \tag{B58}$$

$$H_n(0) = \begin{cases} (-1)^m \frac{(2m)!}{m!} & , \quad n = 2m \\ 0 & , \quad 2m+1 \end{cases} \tag{B59}$$

Three-term recurrence relation:

$$H_0(x) = 1 \tag{B60}$$

$$H_1(x) = 2x \tag{B61}$$

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x) \tag{B62}$$

Derivative:

$$\frac{dH_n(x)}{dx} = 2nH_{n-1}(x) \tag{B63}$$

Differential Equation:

$$y'' - 2xy' + 2ny = 0 \tag{B64}$$

Inequalities:

$$|H_n(x)| < e^{x^2/2}.k.2^{n/2}\sqrt{n!} \quad , \quad k \simeq 1.086435 \tag{B65}$$

$$|H_{2m}(x)| \leq e^{x^2/2}2^{2m}m!\left[2 - \frac{1}{2^{2m}}\begin{pmatrix} 2m \\ m \end{pmatrix}\right] \tag{B66}$$

$$|H_{2m+1}(x) \leq xe^{x^2/2}\frac{(2m+1)!}{(m+1)!} \quad , \quad (x \geq 0) \tag{B67}$$

Rodrigues formula:

$$H_n(x) = \frac{1}{(-1)^n e^{-x^2}}\frac{d^n}{dx^n}\left[e^{-x^2}\right] \tag{B68}$$

DISTRIBUTION LIST

Rapid solution of the Schrödinger equation: Towards a study of the utility of the Bohm filter

Daniel L. Whittenbury, Ayse Kizilersu, Anthony W. Thomas and Samuel P. Drake

Task Sponsor

Sponsor Name/ Position	1

S&T Program

Chief of Cyber and Electronic Warfare Division	1
Research Leader	1
Task Leader	1
Head	1
Science Team Leader	1
Author(s): Daniel L. Whittenbury, Ayse Kizilersu, Anthony W. Thomas and Samuel P. Drake	1

DEFENCE SCIENCE AND TECHNOLOGY GROUP DOCUMENT CONTROL DATA	1. DLM/CAVEAT (OF DOCUMENT)

2. TITLE	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)
Rapid solution of the Schrödinger equation: Towards a study of the utility of the Bohm filter	Document (U) Title (U) Abstract (U)
4. AUTHORS	5. CORPORATE AUTHOR
Daniel L. Whittenbury, Ayse Kizilersu, Anthony W. Thomas and Samuel P. Drake	Defence Science and Technology Group PO Box 1500 Edinburgh, South Australia 5111, Australia

6a. DST Group NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DST-Group–TR–3513	017-243	Technical Report	July, 2018

8. Objective ID	9. TASK NUMBER	10. TASK SPONSOR
N/A	N/A	

13. DST Group Publications Repository	14. RELEASE AUTHORITY
http://dspace.dsto.defence.gov.au/dspace/	Chief, Cyber and Electronic Warfare Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release.

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SOUTH AUSTRALIA 5111

16. DELIBERATE ANNOUNCEMENT

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. RESEARCH LIBRARY THESAURUS

19. ABSTRACT

The second project report for the Efficient Generation and Evolution of Probability Density Maps project is reproduced as a Defence Science and Technology Group technical report. Here we focus on solving the Schrödinger equation numerically for several simple potentials using Fourier and Chebyshev pseudo-spectral methods. The report has been written in such way to be more pedagogical rather than complete.