12-19-2011

Spectral Solution with a Subtraction Method to Improve Accuracy

Matthew Green
Santa Clara University

SANTA CLARA UNIVERSITY
DEPARTMENT OF MECHANICAL ENGINEERING

Date: December 19, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Matthew Green

ENTITLED

Spectral Solution with a Subtraction Method to Improve Accuracy

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

_____

Thesis Advisor

_____

Thesis Reader

_____

Thesis Reader

_____

Department Chair

Spectral Solution with a Subtraction Method to Improve Accuracy

by

Matthew Green

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science in Mechanical Engineering
School of Engineering
Santa Clara University

Santa Clara, California
December 19, 2011

Spectral Solution with a Subtraction Method to Improve Accuracy

Matthew Green

Department of Mechanical Engineering
Santa Clara University
December 19, 2011

ABSTRACT

This work addresses the solution to a Dirichlet boundary value problem for the Poisson equation in 1-D, $\frac{d^2u}{dx^2} = f$ using a numerical Fourier collocation approach. The order of accuracy of this approach can be increased by modifying f so the periodic extension of the right hand side is sufficiently smooth. A proof for the order is given by Sköllermo. This work introduces a subtraction technique to modify the function's right hand side to reduce the discontinuities or improve the smoothness of its periodic extension. This subtraction technique consists of cosine polynomials found by using boundary derivatives. We subtract cosine polynomials to match boundary values and derivatives of f. The derivatives need only be calculated numerically and approximately represent derivatives at the boundaries. Increasing the number of cosine polynomials in the subtraction technique increases the order of accuracy of the solution. The use of cosine polynomials matches well with the Fourier transform approach and is computationally efficient. Implementation of this technique results in a solution with variable accuracy depending on the number of collocation points and approximated boundary derivatives. Results show that the technique can be up to 14^{th} order accurate.

Acknowledgments

I would like to thank Drazen Fabris, Stephan Chiappari, and Sergio Zarantonello for their guidance, insight, and support throughout this thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

The Poisson equation can be solved numerically using a collocation approach. If the right hand side is represented discretely by a set of evenly-spaced collocation points, then its discrete Fourier series can be calculated. Using this information a Fourier series for the numerical solution to the Poisson equation can be calculated. This is a spectral solution. If the problem is periodic, the order of accuracy is the order of the series which is determined by the collocation points.

In reality one would like to calculate the solution of general non-periodic problems. Sköllermo's theorem shows that the solution to the Poisson is second order accurate when the right hand side of the function has non zero and non equal boundary values [19]. Despite the use of a Fourier expansion to represent the functions being a spectral method, a direct Fourier solution will yield only a second order accurate solution. The poor accuracy is directly related to the Gibbs phenomenon, specifically the discontinuous nature of the periodic extension of the function and its representation with a smooth set of functions [10]. Our method involves the modification of the right hand side, using cosine polynomials, into a function with a smoother periodic extension. This method increases the order of accuracy without a large increase in computational load. The solution shown here in 1-D is a proof of concept and can be extended for higher dimensions.

The use of a subtraction function to smooth the periodic extension has been used

by Sköllermo [19] and Averbuch *et al.* [1, 2, 3]. Sköllermo considers the cases where the boundary values of the right hand side match, a smooth periodic extension, and the change in order of accuracy of the solution. The proof to the order of accuracy is given for Sköllermo's method [19]. In Averbuch *et al.*'s work the subtraction functions are known exactly and are algebraic polynomials [1, 2, 3]. In our case the subtraction functions used are calculated numerically and are approximate. In addition, we use cosine polynomials to represent the subtraction function. Cosine polynomials allow use of the Fast Fourier technique to be a computationally efficient calculation of the correction function. In the subtraction technique the cosine polynomial coefficients are calculated numerically from values at the endpoints. The overall accuracy of the technique is adjusted by selecting the order of the cosine polynomial expansion.

This procedure is applicable to numerical elliptic problems where a Laplacian is present. Motivation for this work comes from the Navier-Stokes equations. One step in numerically solving the Navier-Stokes equations involves solving the Poisson equation for viscous boundary conditions. The Poisson equation is the most computationally expensive piece in the solution. The technique outlined in this thesis allows for a highly accurate solution with lower computational load and would be applicable in numerically solving problems in the following fields: heat transfer, image processing, elasticity, and electro magnetism.

Chapter 2

Spectral Solution of the Poisson equation

The Poisson equation can be solved numerically using Fourier Transforms. The order of accuracy of this method depends on the smoothness of the periodic extension of the right-hand side as proved by Sköllermo [19]. If the periodic extension is discontinuous at the endpoints, the solution's accuracy is limited. However, as shown by Averbuch *et al.* [1, 2, 3], a subtraction technique can be applied to correct for these discontinuities. The current work proposes using cosine polynomials, as correction functions, to match a specified number of even-order derivatives at the end points. By using finite differences to calculate the even-order derivatives, and subtracting the *correcting* cosine polynomial from the right-hand side, the proposed methodology produces highly accurate solutions at a very low computational expense. To illustrate our methodology, we consider a Dirichlet boundary value problem for the Poisson equation in 1-D

$$\frac{d^2u(x)}{dx^2} = f(x), \quad u(0) = a, \;\; u(1) = b, \tag{2.1}$$

where $f(x)$ is a continuous real-valued function in the interval $[0, 1]$. In a numerical approach $f(x)$ is represented by its values $f(x_k)$ at evenly spaced collocation points $x_k = k\Delta x$. We assume $\Delta x = \frac{1}{K}$ and $k = 0, 1, 2, ..., K$. We let $\tilde{f}(x)$ be the interpolating

sine polynomial of $f(x)$ at the collocation points, and consider the associated problem

$$\frac{d^2\tilde{u}(x)}{dx^2} = \tilde{f}(x), \quad \tilde{u}(0) = a, \;\; \tilde{u}(1) = b, \tag{2.2}$$

where $\tilde{u}$ is the numerical solution based on the interpolation polynomial, $\tilde{f}$. The L^2 norm of the error between the exact solution and the numerical solution $||u - \tilde{u}||_2$ is of order $O(\Delta x^p)$ where p depends on the degree of smoothness of the periodic extension of $f(x)$. Sköllermo showed, for the case $u(0) = 0$, $u(1) = 0$, that $p = 4$ provided $f'(x)$ exists and is of bounded variation and showed that p = 2 for the case in which u(0) and u(1) are unequal and nonzero (Theorem 1, [19]). Our goal, to improve the continuity of $f(x)$, is achieved by constructing appropriate correction functions.

2.1 Fourier Transforms

The Fourier Transform is the process of transforming a function in physical space into Fourier space. For a function $f(x)$, defined on the interval $[0, 1]$ the Fourier sine coefficients are defined by

$$F_k = 2 \int_0^1 f(x) \sin(k\,\pi x)\, dx, \tag{2.3}$$

and $f(x)$ can be represented as the sum of its Fourier series

$$f(x) = \sum_{k=1}^{\infty} F_k \, \sin(k\,\pi x) \tag{2.4}$$

if $f(x)$ is sufficiently smooth. If the data are discrete then there is a discrete Fourier sine transform that is analogous, where the sine coefficients are given by

$$S_k = 2 \sum_{m=1}^{K-1} f(x_m) \, \sin\left(k\,\pi x_m\right) \qquad x_m = \frac{m}{K}. \tag{2.5}$$

This yields a finite set of Fourier coefficients. That finite set of coefficients represents the interpolating polynomial

$$\tilde{f}(x) = \sum_{k=1}^{K-1} S_k \sin\left(\frac{k\,\pi x}{K}\right) \tag{2.6}$$

which is equal to $f(x)$ at the interpolating points, where K is the number of collocation points and S_k represents the amplitude of the k^{th} sine coefficient. The Fast Fourier Transform can be used to calculate S_k and reduce computational load. Now representing $\tilde{f}(x)$ in sine terms, the computation of the solution $\tilde{u}(x)$ is straightforward as shown in the next section.

2.2 Solving the Poisson Equation in Fourier Space

Finding the solution of equation (2.2) involves scaling the sine coefficients. The discrete Fourier sine representation of $\tilde{u}(x)$ is denoted by the coefficients R_k. Inserting the discrete Fourier sine series $\tilde{u}(x)$ and $\tilde{f}(x)$ from equation (2.6) into equation (2.1), we obtain

$$\frac{d^2}{dx^2} \sum_{k=1}^{K-1} R_k \sin\left(\frac{k\,\pi x}{K}\right) = \sum_{k=1}^{K-1} S_k \sin\left(\frac{k\,\pi x}{K}\right). \tag{2.7}$$

Because the functions $\sin\left(\frac{k\,\pi x}{K}\right)$ in the Fourier sine series are orthogonal to each other on $[0, 1]$ we equate coefficients of the same wave number. Note that

$$\frac{d^2}{dx^2} R_k \sin\left(\frac{k\,\pi x}{K}\right) = \frac{-\pi^2\,k^2}{K^2} R_k \sin\left(\frac{k\,\pi x}{K}\right),$$

thus from (2.7)

$$R_k \frac{-k^2\pi^2}{K^2} \sin\left(\frac{k\,\pi x}{K}\right) = S_k \sin\left(\frac{k\,\pi x}{K}\right).$$

Then algebraically we can solve for the coefficients in $\tilde{u}(x)$

$$R_k = \frac{-K^2}{\pi^2\,k^2} S_k; \tag{2.8}$$

thus,

$$\tilde{u}(x) = \sum_{k=1}^{K-1} \frac{-K^2}{\pi^2\, k^2}\, S_k\, \sin\left(\frac{k\,\pi x}{K}\right). \tag{2.9}$$

This constant $\frac{-K^2}{\pi^2\, k^2}$ relates the Fourier sine series representation of $f(x)$ to the Fourier sine series representation of $u(x)$. This allows us to scale the coefficients S_k to solve the Poisson equation.

2.3 Gibbs Phenomenon

When $f(0)$ is not equal to $f(1)$ the periodic extension of f is discontinuous. When the Poisson problem is solved with a spectral technique there is an error in the solution for a right hand side that has a discontinuous periodic extension. This error is related to the Gibbs phenomenon. This error will also exist if the derivatives of the periodic extension are discontinuous.

To review, the Gibbs phenomenon occurs when a discontinuous function is represented by a sine or cosine series expansion. The Fourier series does not converge pointwise at the points of discontinuity. Where the discontinuities occur in f, in this case at non-periodic boundaries, an overshoot or undershoot will occur near the discontinuity. The error is only be mitigated by increasing the terms in the expansion, thus isolating the overshoot to only a location near the discontinuity. This overshoot or undershoot will never disappear int he vicinity of discontinuity. This allows the overall solution to converge reasonably well except at those regions.

Previous work has shown that minimizing the Gibbs phenomenon requires the addition of a filter in either Fourier space or in physical space [10]. Fourier space filters or spectral filters modify the data in Fourier space. These filters successfully remove most of the Gibbs phenomenon, the high frequency portions. However this compromises the function's representation, which is detrimental to the accuracy of the solution. Minimal computational cost is introduced in the application of Fourier space filters. Physical space filters involve modifying the original data to smooth or remove

the discontinuities. The application of the filter leads to increased computational cost.

In our alternate approach, a physical space subtraction method is used in this paper to reduce the effect of the Gibbs phenomenon. This method does not introduce the loss of accuracy associated with Fourier space filters. Thus our work is a method in which boundaries and derivatives are subtracted in physical space, via a cosine series representation.

2.4 Applying the Subtraction Function

If $f(x)$ does not have a periodic extension we apply a subtraction method [1, 2, 3]. Subtracting a function $E(x)$ from $f(x)$ results in a function $g(x)$. This function, $g(x)$, can be made more smooth by subtracting a function $E(x)$ that cancels discontinuities in the extension of $f(x)$ so that $g(0) = 0$ and $g(1) = 0$ and its derivatives, $g^{(p)}(0) = 0$ and $g^{(p)}(1) = 0$, where p is an even integer. We can achieve this by using a cosine series as the subtraction function. The use of a cosine polynomial is convenient because the same Fourier procedure can be used to determine the contribution to the solution as is used to solve equation (2.1). With $\tilde{g}(x)$ being the interpolating polynomial without discontinuities in the periodic extension we can solve the Poisson equation

$$\frac{d^2\tilde{v}(x)}{dx^2} = \tilde{g}(x). \tag{2.10}$$

The coefficients of the cosine interpolation polynomial are represented as C_k and the contribution to the solution involves scaling by the same scalar $\frac{-K^2}{\pi^2 k^2}$. The solution to $\frac{d^2\tilde{w}(x)}{dx^2} = E(x)$ is $\tilde{w}(x)$, which is given by

$$\tilde{w}(x) = \sum_{k=1}^{D} \frac{-K^2}{\pi^2 k^2} C_k \cos\left(\frac{k\pi x}{K}\right). \tag{2.11}$$

The number of cosine terms is represented by D which is different than the number of terms in the sine series representation. The physical space solutions $\tilde{w}(x)$ and $\tilde{v}(x)$

are now summed to represent the solution to equation (2.1). The solution is given by

$$\widehat{u}(x) = \tilde{v}(x) + \tilde{w}(x), \tag{2.12}$$

where $\widehat{u}(x)$ is the better approximation of $u(x)$. Based on Sköllermo's Theorem 1 [19], we need to find $E(x)$ that will subtract the non-zero boundaries of $f(x)$ so that the extension $\tilde{g}(x)$ is periodic. And representing $E(x)$ as a cosine polynomial is easier because it can be computed simultaneously with the sine component in use of the FFT. So we need to know $f(0)$, $f(1)$, $f^{(2)}(0)$, and $f^{(2)}(1)$ to set $E(x) = f(x)$ and $E^{(2)}(x) = f^{(2)}(x)$ for $x = 0$ and $x = 1$.

2.5 Numerical Approximation of Derivatives

Since the original data is given discretely, we approximate the various derivatives that we need by a Taylor approximation. (Other methods based on the Taylor series can be used to increase accuracy and computational efficiency, but would have only a small impact.) We first look at the Taylor representation of a suitably smooth function $f(x)$ near a point x_0, given by

$$f(x) = \sum_{n=0}^{M-1} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \frac{1}{M!} f^{(M)}(\zeta)(x - x_0)^M. \tag{2.13}$$

This equation holds for all x sufficiently close to x_0, with ζ depending on x and lying between x and x_0. This Taylor approximation can be used to generate a collection of linear equations by varying the number of collocation points K. Each new collocation point adds another equation, so for K collocation points, derivatives up to the K^{th} are approximated. This set of equations essentially forms a linear system for the derivatives. The example below shows the process for finding the numerical

approximation up to the fourth derivative at a boundary.

$$f(x_k)-f(x_0) \approx f^{(1)}(x_0)k\Delta x+\frac{1}{2}f^{(2)}(x_0)(k\Delta x)^2+\frac{1}{6}f^{(3)}(x_0)(k\Delta x)^3+\frac{1}{24}f^{(4)}(x_0)(k\Delta x)^4, \tag{2.14}$$

where k is incremented ($k = 1, 2, 3, 4$) to generate a set of four equations. These equations can be written as a linear system and solved as shown below:

$$\widehat{b} = A\widehat{d}, \qquad \widehat{d} = A^{-1}\widehat{b},$$

$$\widehat{b} = \begin{bmatrix} f(x_1) - f(x_0) \\ f(x_2) - f(x_0) \\ f(x_3) - f(x_0) \\ f(x_4) - f(x_0) \end{bmatrix}, \quad \widehat{d} = \begin{bmatrix} f^{(1)}(x_0) \\ f^{(2)}(x_0) \\ f^{(3)}(x_0) \\ f^{(4)}(x_0) \end{bmatrix}, \quad A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{6} & \frac{1}{24} \\ 2 & \frac{4}{2} & \frac{8}{6} & \frac{16}{24} \\ 3 & \frac{9}{2} & \frac{27}{6} & \frac{81}{24} \\ 4 & \frac{16}{2} & \frac{64}{6} & \frac{256}{24} \end{bmatrix}.$$

Arbitrary order approximate derivatives can be generated by including additional terms. The matrix coefficients can be generated using the following expression

$$A_{mn} = \frac{m^n}{n!}. \tag{2.15}$$

The accuracy of the approximated derivatives depends on the number of collocation points used. The derivative approximation method eventually experiences numerical round-off error and is limited in total accuracy. In fact matrix A has a large condition number thus inverting and solving the linear system is slightly more difficult. The accuracy of the approximation can possibly limit the accuracy of solution via $v(x)$.

The total number of calculated derivatives cannot exceed the number of collocation of points when using this method.

2.6 Cosine Series Generation

A cosine polynomial, as shown in section 2.4 can be generated, by using the approximate derivatives calculated in section 2.5 that represents $E(x)$. The cosine series when matched with a FFT can be computationally efficient. When the cosine series is subtracted from $f(x)$ the remaining $g(x)$ can be represented as a more quickly converging series of sine polynomials. For each boundary pair, $f^{(2p)}(x_0)$ and $f^{(2p)}(x_1)$ for $p = 1, 2, ..., D$, a pair of cosine terms is required to represent those conditions. Choose an even positive integer D, the number of cosine terms to be used. The order of the highest order derivatives approximated at the boundary points 0 and 1 is then $2D - 2$. Thus the following cosine series is generated,

$$E(x) = \sum_{k=1}^{D} C_n \cos(k\pi x), \tag{2.16}$$

where C_k is the approximate cosine interpolating polynomial coefficient for $E(x)$. Note C_0, the constant function, is not used since this would require an additional algebraic polynomial to the Poisson equation.

2.6.1 Example: Generating Cosine Coefficients for Six Data Points

The following example is the calculation based on six cosine terms and approximated derivatives of order 0, 2, and 4. We apply equation (2.16) to obtain the six equations

$$f(0) = \sum_{j=1}^{6} C_j,$$

$$f(1) = \sum_{j=1}^{6} C_j(-1)^j,$$

$$f^{(2)}(0) = -\sum_{j=1}^{6} j^2\pi^2 C_j,$$

$$f^{(2)}(1) = \sum_{j=1}^{6} j^2\pi^2 C_j(-1)^{j+1},$$

$$f^{(4)}(0) = \sum_{j=1}^{6} j^4 \pi^4 C_j,$$

$$f^{(4)}(1) = \sum_{j=1}^{6} j^4 \pi^4 C_j (-1)^j,$$

$$\widehat{b} = \begin{bmatrix} f^{(0)}(0) \\ f^{(0)}(1) \\ \frac{f^{(2)}(0)}{\pi^2} \\ \frac{f^{(2)}(1)}{\pi^2} \\ \frac{f^{(4)}(0)}{\pi^4} \\ \frac{f^{(4)}(1)}{\pi^4} \end{bmatrix}, \quad \widehat{d} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 \\ -1^2 & -2^2 & -3^2 & -4^2 & -5^2 & -6^2 \\ 1^2 & -2^2 & 3^2 & -4^2 & 5^2 & -6^2 \\ 1^4 & 2^4 & 3^4 & 4^4 & 5^4 & 6^4 \\ -1^4 & 2^4 & -3^4 & 4^4 & -5^4 & 6^4 \end{bmatrix}.$$

Solving for $\widehat{d}$ results in the coefficients of the cosine series. With the cosine series constructed an Inverse Discrete Cosine Transform can be performed to extract the subtraction function for modifying $f(x)$.

2.6.2 Inclusion of Boundary Conditions

Boundary values required for the solution $u(x)$ can also be matched with the cosine series. Typically, the boundary value conditions could be added as a linear function and a constant to the solution $u(x)$; however, producing a solution numerically via our method can be done by adding $f^{(-2)}(0)$ and $f^{(-2)}(1)$ to the cosine series generation procedure. We look back at the cosine series calculation method. Instead of starting at $f(0)$ and $f(1)$, the system of equations would introduce two new equations for the boundary value conditions at $u(0)$ and $u(1)$. This increases the number of equations

by two while properly introducing the boundary value conditions.

2.6.3 Iterative Refinement for Matrices with Poor Condition Numbers

The matrix used to solve the cosine coefficients has a poor condition number. The condition number of the matrix increases as the number of derivatives increases. To compensate for this high condition number, the numerical method of iterative refinement is used to solve for the correct coefficients [20](pages 454 to 461). We describe the method for improving the accuracy of the solution to a system with a poor condition number. We first obtain $\widehat{d}_1$, as an approximate solution of

$$A\widehat{d} = \widehat{b}.$$

Putting

$$\Delta\widehat{b}_1 = \widehat{b} - A\widehat{d}_1,$$

we obtain $\Delta\widehat{d}_1$, as an approximate solution of

$$A\widehat{d} = \Delta\widehat{b}_1.$$

Then put

$$\widehat{d}_2 = \Delta d_1 + \widehat{d}_1.$$

If $\widehat{d}_2 - \widehat{d}_1$ is small enough in norm, we stop and declare $\widehat{d}_2$ as a good enough solution; otherwise we put

$$\Delta\widehat{b}_2 = \widehat{b} - A\widehat{d}_2$$

and perform another iteration. Eventually either $\widehat{d}_{k+1} - \widehat{d}_k$ is small enough in norm, or we reach the maximum number of iterations that we care to perform.

Chapter 3

Computational Load

The Fast Fourier Transform with a cosine polynomial subtraction has significantly lighter computational load than inverting a large full matrix. Analyzing the computational load requires the breakdown of each component based on the number of divisions and multiplications in that piece. Looking first at the Fast Fourier Transform the computational load is much smaller. While numerous papers have been published on the accuracy of the Fast Fourier Transform, we use the conservative computational load $5\,K\,\log_2(K)$Averbuch *et al.* [1]. This is for a single transform of $f(x)$ where K is the number of collocation points. To recover the solution takes two transforms. Finding the derivatives and generation of the cosine series improves the computational load. Denote by $D-4$ the number of derivatives that need to be calculated to produce a subtraction function where D is the number of terms in the cosine polynomials in the subtraction function. The basic method for solving a system of equations, Gaussian Elimination, requires computational load K^3. In this case for n collocation points the initial point has to be considered and thus for each boundary the computational load for finding the desired derivatives is $(D-1)^3$, also through Gaussian Elimination While using $(D-1)^3$ is conservative, the amount of computational load added to the overall process is minimal when compared to the Fast Fourier Transform.

Again looking at the number of derivatives, $D-4$, we see that the computational

load for creating the cosine polynomials using the final boundary conditions is roughly D^3. Other sources of computational load are present. These sources of load are small and as such do not affect the load much. The total computational load for the Fast Fourier Transform with cosine polynomial subtraction is roughly

$$5\,K\,\log_2(K) + 2(D-1)^3 + D^3. \tag{3.1}$$

With the subtraction method the accuracy of the problem is variable and can be set to maximize computational efficiency. These computational calculations only hold true when K is a positive integer power of 2.

Chapter 4

Theoretical Accuracy

Determining the order of accuracy of our method requires examining the Fast Fourier Transform, and the accuracy of the calculated derivatives. According to Sköllermo [19], the order of accuracy of an unmodified zero endpoint Discrete Sine Transform was found to be $O(\Delta x^2)$, where Δx is the difference between discrete points in $f(x)$. With the base order of accuracy established, Averbuch [1] improved upon Sköllermo's subtraction method to increase the order of accuracy of the overall scheme. Subtracting a second order accurate function from $f(x)$ yields accuracy of $O(\Delta x^4)$.

If we know $f^{(j)}(0)$ and $f^{(j)}(1)$ exactly for $j = 0, 2, 4..., p$ the resulting limiting accuracy would be $O(\Delta x^{4+p})$. Finding the exact derivatives may not be possible when only discrete data is available. In our case numerical approximations of the derivatives at the boundaries are substituted. Thus $O(\Delta x^4)$ is the order of accuracy if $f(0) = f(1) = 0$ (we can use a shift and a subtractive corrective function to ensure that these conditions are satisfied for a modification of the original function f). When we use a subtraction function we only approximate the solution. Hence, $g(0) = O(\Delta x^q)$ and $g(1) = O(\Delta x^q)$ where q is the appropriate number of terms in the Taylor series subtraction. Thus, we only need the boundary values approximately satisfied; *i.e.*, $g(0) = O(\Delta x^4)$ by Sköllermo [19]. The base order of accuracy is $O(\Delta x^{4+p})$ plus the error in the boundary value $O(\Delta x^q)$ multiplied by the lower order accuracy from the Sköllermo theorem for non-zero boundary values, $O(\Delta x^2)$. Therefore the order is

$O(\Delta x^{4+p}) + O(\Delta x^{2+q})$.

For this example the solution is as accurate as if we know the boundary values exactly. Subsequently, we can extend this technique to higher order accuracy as long as the derivatives are calculated to the needed order of accuracy, $q = 2$. For 6^{th} order of accuracy we use $q = 4$ and $p = 2$ when second order accurate derivatives are calculated. The calculation for the order of accuracy would be $O(\Delta x^6) \approx O(\Delta x^{4+p}) + O(\Delta x^{2+q})$. Use of the Taylor series should provide the appropriate order of accuracy for the chosen higher order derivatives.

Chapter 5

Numerical Accuracy Results

To determine the accuracy a Matlab code is written to test the cosine series subtraction method. Some sample cases for a variety of functions are tested. These functions are provided in the middle column of the table below.

The initial two cases are designed to test the accuracy of the spectral solver and

Case	$f(x)$	$u(x)$
1	$-4\pi^2 \sin(2\pi x)$	$\sin(2\pi x)$
2	$-4\pi^2 \cos(2\pi x)$	$\cos(2\pi x)$
3	$-4\pi^2 \sin(2\pi x) - 4\pi^2 \cos(2\pi x)$	$\sin(2\pi x) + \cos(2\pi x)$
4	$\frac{6x^2-2}{x^6+3x^4+3x^2+1}$	$\frac{1}{1+x^2}$
5	$2\sec^2(x)\tan(x)$	$\tan(x)$
6	$\exp(x)$	$\exp(x)$
7	$(4x^2+2)\exp(x^2)$	$\exp(x^2)$
8	$(25x^8+20x^3)\exp(x^5)$	$\exp(x^5)$
9	$(100x^{18}+90x^8)\exp(x^{10})$	$\exp(x^{10})$
10	$(2500x^{98}+2450x^{48})\exp(x^{50})$	$\exp(x^{50})$

Table 5.1: Cases 1 through 10 with analytical solutions

the subtraction method: the cosine subtraction polynomials and the Fourier sine integration. Case 3 tests the combination of these two pieces; this would be the ideal case and shows that the predicted order indeed occurs. Cases 4 through 10 involve functions, non-periodic on $[0, 1]$ with nonzero derivatives. Cases 6 through 10 and are similar, but increase in stiffness.

All the cases have simple analytical solutions for easy comparisons with numerical

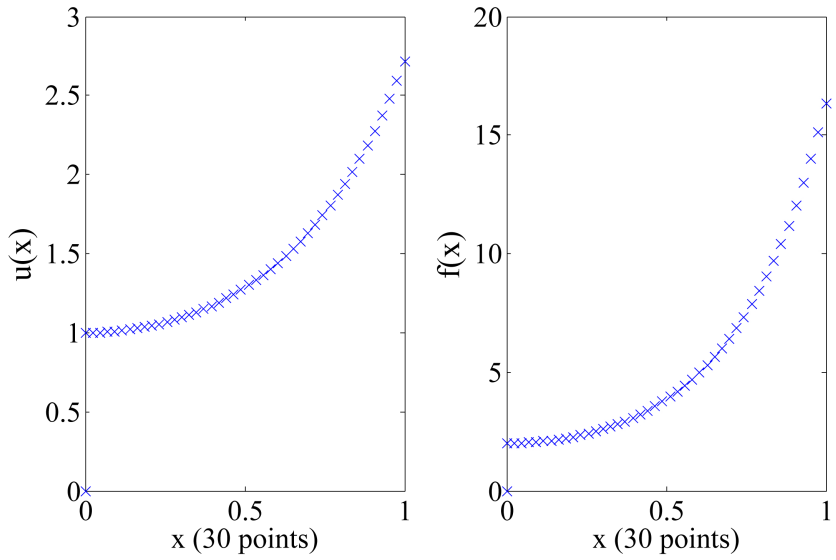

Figure 5.1: Case 7. Function $f(x)$ and the numerical solution to the Poisson equation $u(x)$

results. Here u satisfies the equation $\frac{d^2}{dx^2}u(x) = f(x)$ exactly for all x in the interval $[0, 1]$. Examining the order of accuracy numerically involves plotting the total error based on evaluation of the numerical solution at discrete values, Figure 5.3. A set of cases to test the method are run by varying the number of cosine polynomials used and the number of collocation points. In each case the slope of the curve corresponds to the order of accuracy of the method. Notice that the curve's slope increases as the number of cosine terms increases. Trying to match more derivatives, we see that the round off error generated by the Taylor polynomial approximation becomes evident. The error for 10^{-14} is equivalent to machine precision. This round-off error for higher-order derivatives is the limiting factor to the overall scheme. Figures for all the test cases are available in appendix B.

Tables 5.2, 5.3, 5.4, and 5.5 show the calculated order of accuracy for several cases. Each additional pair of derivatives, of even order, at the boundary regions adds two cosine terms to the cosine polynomial subtraction function. Observing cases 1, 2, and 3 we see that the order of the cosine polynomial subtraction governs the accuracy of the combined solution.

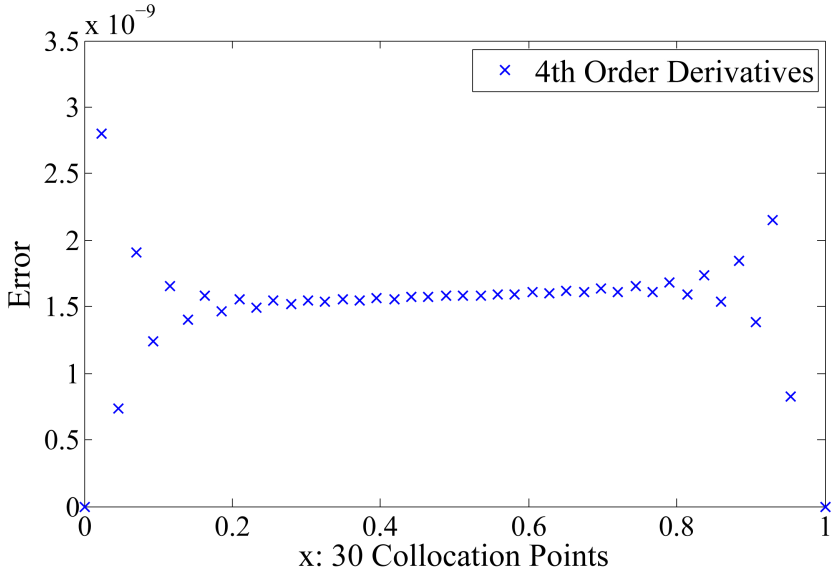

Figure 5.2: Case 7. Error between analytical and numerical solutions to $f(x)$ when using 4th order accurate derivatives at 30 collocation points

Cases 4 through 6 show improved order of accuracy for matching of higher derivatives. These functions are not periodic and have derivatives of every even order; however, the ability to capture the derivatives is more challenging. This can cause numerical values that hinder or help the accuracy of the solution. In these particular cases the magnitude of the function's derivatives increases the order of accuracy.

Observing cases 6 through 10, as the stiffness increases the order of accuracy decreases with higher order cosine polynomials. This decrease in accuracy is expected: as the function becomes increasingly stiff the Taylor series is directly affected. As shown here the Taylor series has trouble resolving the discrete function after case 8. As we increase the number of collocation points, the Taylor series representation becomes more accurate. This further demonstrates that the total accuracy is governed by the Taylor polynomial accuracy.

Highest-Order Derivative for Matching	Cases					Theoretical Order of Accuracy
	1	2	3	4	5	
2	6.653	5.624	5.635	5.677	5.723	6
4	8.629	7.507	7.563	7.793	7.537	8
6	10.63	9.367	9.532	10.00	8.990	10
8	12.57	11.26	11.38	12.74	10.87	12
10	14.61	12.66	13.38	15.52	12.51	14
12	15.92	15.46	15.45	18.50	15.20	16
14	15.54	16.66	15.74	20.57	18.12	18

Table 5.2: Results for Cases 1 through 5 at 32 collocation points

Highest-Order Derivative for Matching	Cases					Theoretical Order of Accuracy
	6	7	8	9	10	
2	5.145	5.678	4.892	4.729	3.126	6
4	7.239	7.794	6.847	5.958	3.426	8
6	9.409	10.03	8.208	7.028	3.678	10
8	12.51	12.68	9.739	8.044	3.902	12
10	15.40	15.55	11.09	8.985	4.100	14
12	18.02	18.29	12.59	9.885	4.280	16
14	22.34	22.65	14.62	10.81	4.446	18

Table 5.3: Results for Cases 6 through 10 at 32 collocation points

Highest-Order Derivative for Matching	Cases					Theoretical Order of Accuracy
	1	2	3	4	5	
2	6.595	5.573	5.584	5.592	5.626	6
4	8.601	7.559	7.576	7.647	7.445	8
6	10.66	9.557	9.547	9.722	8.646	10
8	0	0	0	12.74	11.78	12
10	0	0	0	0	0	14
12	0	0	0	0	0	16
14	0	0	0	0	0	18

Table 5.4: Results for Cases 1 through 5 at 64 collocation points

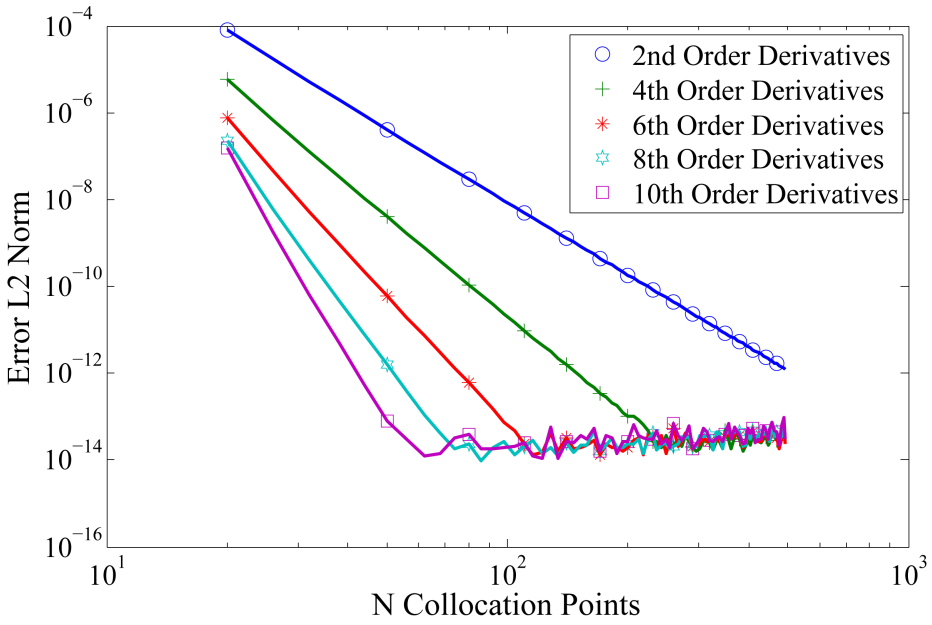

Figure 5.3: Case 7. L^2 norm between analytical and numerical solutions using up to tenth order accurate derivatives.

Highest-Order Derivative for Matching	Cases					Theoretical Order of Accuracy
	6	7	8	9	10	
2	5.091	5.602	5.148	5.006	3.669	6
4	7.146	7.663	7.033	6.459	4.147	8
6	9.218	9.684	8.639	7.802	4.559	10
8	14.74	0	10.26	9.076	4.922	12
10	0	0	11.75	10.28	5.246	14
12	0	0	13.08	11.49	5.547	16
14	0	0	0	13.05	5.839	18

Table 5.5: Results for Cases 6 through 10 at 64 collocation points

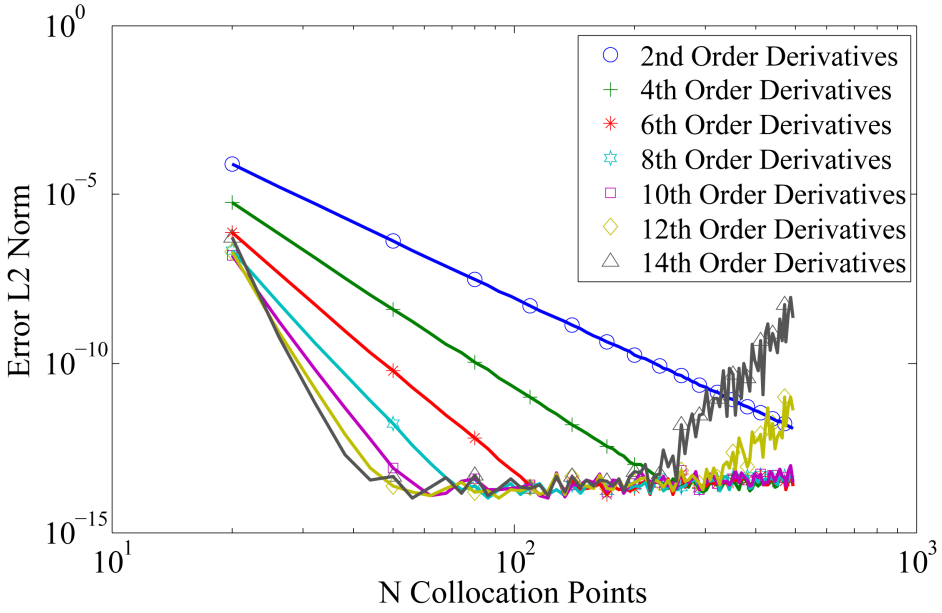

Figure 5.4: Case 7. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

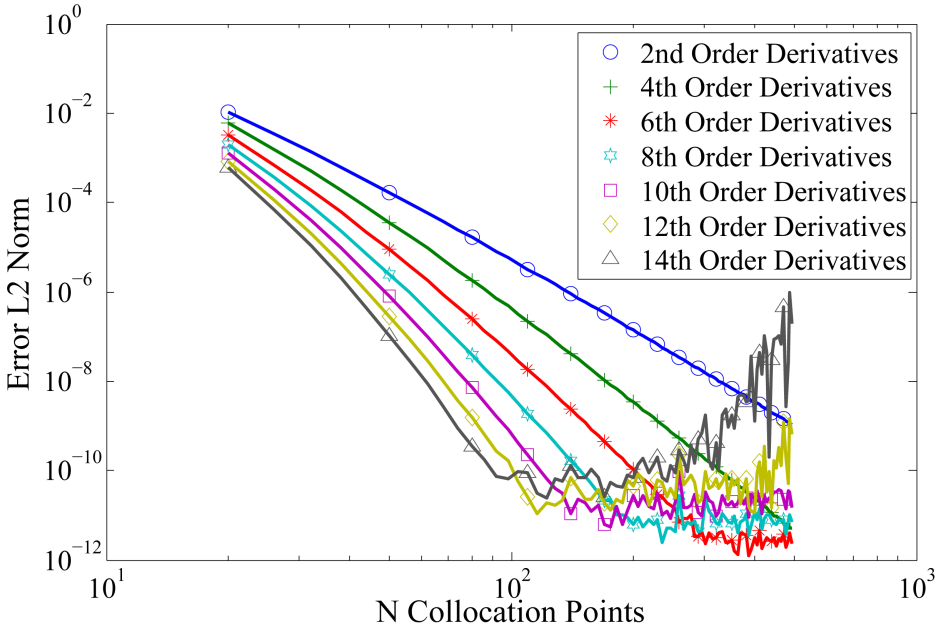

Figure 5.5: Case 9. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

Chapter 6

Conclusion

In this work a subtraction function is used to precondition a Poisson boundary value problem's right hand side, f. Sköllermo's theorem shows the order of accuracy of the solution to the Poisson problem is based on the smoothness of the periodic extension [19]. The resulting modified periodic extension, g, is smoother than f. Thus the order of accuracy obtained by means of the Fourier collocation approach using g is of higher order than using f. Derivatives at the boundaries need to be used to generate a subtraction function to smooth the right hand side. We do not know the derivatives at the boundaries exactly, but these derivatives can be calculated numerically. Cosine polynomials are constructed from these derivatives, to generate the subtraction function. The use of cosine polynomials matches well with the Fourier collocation approach. In particular, their construction is much simpler than that of algebraic polynomials to subtract discontinuities.

As shown, the order of accuracy of the solution can be modified based on the accuracy of even order derivatives calculated at the boundaries. In certain conditions the numerical accuracy is limited due to round off error at large numbers of computations. Furthermore in computing derivatives, the matrix is poorly conditioned and iterative refinement is used to improve accuracy. Thus the number of collocation points can be coupled with the order of accuracy of the cosine polynomials. In essence we could use the best ratio of solution accuracy to computational load for the solution to the

Poisson equation. The resulting outcome would be low computational load for a high accuracy solution using minimal collocation points.

This work builds the ground for future research. The expansion of this technique into higher dimensions is relatively straightforward and can be used in multidimensional problems. Also due to the nature of the Fourier collocation solution, multi dimensions problems can be solved implicitly without the need of iteration between dimensions. Transitioning solutions between grids with different resolutions could easily be extended with the numerical derivative approach outlined here. This technique can also be incorporated into a numerical solution of a partial differential equation, such as the Navier-Stokes, vorticity, and elasticity equations.

Bibliography

[1] A. Averbuch, M. Israeli, and L. Vozovoi, *On a fast direct elliptic solver by a modified Fourier method*, Numerical Algorithms **15**, no. 3–4 (1997), 287–313.

[2] A. Averbuch, M. Israeli, and L. Vozovoi, *A fast Poisson solver of arbitrary order accuracy in rectangular regions*, SIAM J. Sci. Comput. **19** (1998), 933–952.

[3] E. Braverman, B. Epstein, M. Israeli, and A. Averbuch, *A fast spectral subtractional solver for elliptic equations*, SIAM J. Sci. Comput. **21**, no. 1 (Aug. 2004), 91–128.

[4] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods in fluid dynamics*, Springer-Verlag, Berlin, 1988.

[5] H. S. Carslaw, *Introduction to the Theory of Fourier's Series and Integrals*, third ed., Dover, New York, 1950.

[6] H. S. Carslaw and J. C. Jaeger, *Conduction of Heat in Solids*, second ed., Clarendon Press, Oxford, 1986.

[7] Paul DuChateau and David Zachmann, *Applied Partial Differential Equations*, Dover, New York, 2002.

[8] Knut S. Eckhoff, *On a high order numerical method for functions with singularities*, Math. Comput. **67**, no. 223 (July 1998), 1063–1087.

[9] D. Gottlieb, *Issues in the application of high order schemes*, Algorithmic trends in computational fluid dynamics, ICASE/NASA Series, Springer, New York (1993), 195–218.

[10] David Gottlieb and Chi-Wang Shu, *On the Gibbs Phenomenon and Its Resolution*, SIAM Rev. **39**, no. 4 (Dec. 1997), 644–668.

[11] L. Greengard and J.-Y. Lee, *A direct adaptive Poisson solver of arbitrary order accuracy*, J. Comput. Phys. **125** (1996), 415–424.

[12] R. W. Hockney, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach. **12**, no. 1 (Jan. 1965), 95–113.

[13] L. V. Kantorovich and V. I. Krylov, *Approximate Methods of Higher Analysis*, trans. C. D. Benster, Noordhoff, Groningen, 1964.

[14] C. Lanczos, *Discourse on Fourier Series*, Oliver and Boyd, Edinburgh and London, 1966.

[15] Mark A. Pinsky, *Introduction to Fourier Analysis and Wavelets*, Brooks/Cole, Pacific Grove, CA, 2002.

[16] P. J. Roache, *A pseudo-spectral FFT technique for nonperiodic problems*, J. Comput. Phys. **21** (1978), 204–220.

[17] J. B. Rosser, *Fourier series in the computer age*, Mathematics Research Center, Technical Summary Report #1401, U. Wisc., Madison, WI, 1974.

[18] Z. Shi and B. Hassard, *Precise solution of Laplace's equation*, Math. Comput. **64** (1995), 515–536.

[19] Gunilla Sköllermo, *A Fourier method for the numerical solution of Poisson's equation*, Math. Comput. **29**, no. 131 (July 1975), 697–711.

[20] R. L. Burden and J. D. Faires, *Numerical analysis*, Brooks/Cole, Belmont, CA, 2005.

Appendix A

Matlab Code

```
%1-D solution to the Poisson equation
%Matthew Green (2011) Drazen Fabris (2007)

clc;
clear all;
n=32; %number of points
N=6; %order of derivatives calculated
u0=1; %d^2u/dx^2 boundary at zero
u1=1;%d^2u/dx^2 boundary at one
x=0:(1/(n-1)):1;
f=sin(2*pi*x)+cos(2*pi*x); %function f
u=IDposSolver(f,n,N,u0,u1); %solution to d^2u/dx^2=f

%-----------------------------------------------
%1-D Poisson solver using FFT and cosine subtraction
%Drazen Fabris (2007) Matthew Green (2011)

function [u]=IDposSolver(f,n,N,u0,u1)
%Calculate derivatives at endpoints
[d0,d1]=IDdervcalc(f,n,N);
%produce cosine subtraction polynomial based on derivatives
[cosRM]=IDcosCreate_IR(d0,d1,N,f(1),f(n),u0,u1);
%construct subtraction polynomial
cosIM=zeros(1,n);
cosIM(2:N+5)=cosRM'/(1/(n-1));
cosRMF=idctcopy(cosIM');
coscoef=cosIM;
%modifty f and take FFT
f2=f(2:end-1)-cosRMF(2:end-1)';
sincoef = dstcopy(f2')';
%Modify coefficients to produce solution to Poisson equation
for i=1:size(sincoef,2)
```

```
    iisc(i)=-1*sincoef(i)/(pi*i)^2;
end
iicc(1) = 0;
for i=2:size(coscoef,2)
    iicc(i)=-1*coscoef(i)/(pi*(i-1))^2;
end
u=idctcopy(iicc')'+[0 idstcopy(iisc')' 0];
return

%-----------------------------------------------------------
%Derivatives at function f endpoints based on Taylor series 1-D
%Drazen Fabris (2007)  Matthew Green (2011)

function [d0,d1]=IDdervcalc(f,n,N)
d0 = zeros(1,N);
d1 = zeros(1,N);
%set up matrix for solving derivatives at boundaries
for i=1:N+1
    for j=1:N+1
        A(i,j) = i^(j)/prod(1:j);
    end
end
%setting end conditions removing false negatives
E = inv(A);
for i=1:2:N+1
    D(i,:) = -1*E(i,:);
end
for i=2:2:N+1
    D(i,:) = E(i,:);
end
%solve for the end derivatives
for j=2:2:N
    d0(j) = E(j,:)*(f(2:N+2)-f(1))'/(1/(n-1))^j;
    d1(j) = D(j,:)*(f(n-1:-1:n-N-1)-f(n))'/(1/(n-1))^j;
end
return

%-------------------------------------------------------------
%Construction of cosine series to modify function f for Fourier processing.
%Drazen Fabris (2007)  Matthew Green (2011)

function [cosRM]=IDcosCreate_IR(d0,d1,N,f1,fn,u0,u1)
%creating matrix of cosine derivatives
for i=1:N/2+2
    for j=1:N+4
```

```
        A(i*2-1,j) =  j^(2*(i-2))*(-1)^(i); %first boundary coefficients
        A(i*2,j) = A(i*2-1,j)*(-1)^j;  %second boundary coefficients
    end
end
D=inv(A);
%Add final boundary conditions and derivatives
R(1)=u0*pi^2;R(2)=u1*pi^2;
R(3)=f1;R(4)=fn;
for i=4:2:N+2
    R(i+1)=d0(i-2)/(pi)^(i-2);
    R(i+2)=d1(i-2)/(pi)^(i-2);
end
%final cosine coefficients that are going to be subtracted from f
L=D*R';
%Iterative Refinement for poor condition numbers  Matthew Green (2011)
for i=1:6 %more itterations are needed for stiff problems
db=R'-A*L;
Rem=D*db;
L=L+Rem;
end
cosRM=L;
return

%---------------------------------------------------------------
% inverse sine transform, assume that size(z)=[nx,ny]=[2^k-1,ny]
% C. T. Kelley, May 1994
% D. Fabris, July 2007 scaled appropriately
% This code comes with no guarantee or warranty of any kind.

function u=isintv(z)
[nx,ny]=size(z);
ww=2*ifft([zeros(ny,1), z']',2*nx+2);
u=imag(ww(2:nx+1,:));

%---------------------------------------------------------------
% sine transform, assume that size(z)=[nx,ny]=[2^k-1,ny]
% C. T. Kelley, May 1994
% D. Fabris, July 2007 (scaled appropriately)
% result  = sine transform

function u=sintv(z)
[nx,ny]=size(z);
ww=-2*fft([zeros(ny,1), z']',2*nx+2);
u=imag(ww(2:nx+1,:));
```

```
%---------------------------------------------------------------
% inverse cosine transform, assume that size(z)=[nx,ny]=[2^k-1,ny]
% based on inverse sine transform of C. T. Kelley, May 1994
% D. Fabris, Sept. 2004
% This code comes with no guarantee or warranty of any kind.

function u=icostv(z)
[nx,ny]=size(z);
q = [z',z(end-1:-1:2,:)'];
ww=ifft(q',2*nx-2);
u=real(ww(1:nx,:));
```

Appendix B

Case Figures

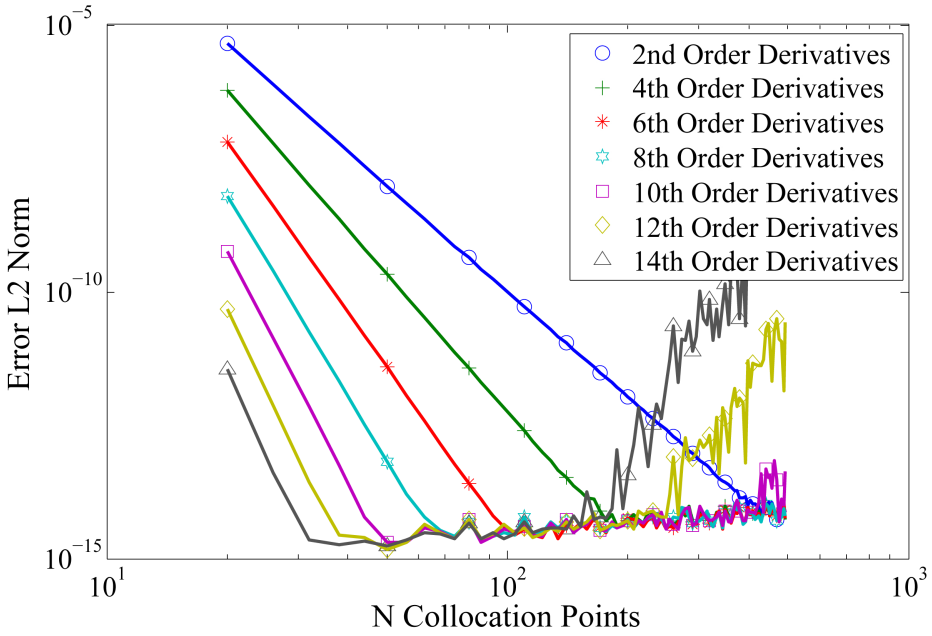

Figure B.1: Case 1. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

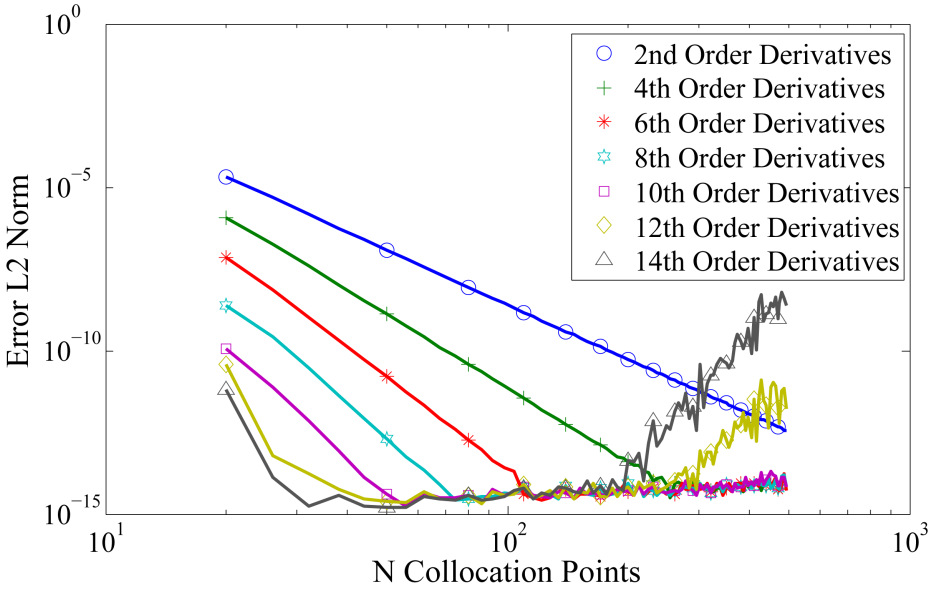

Figure B.2: Case 2. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

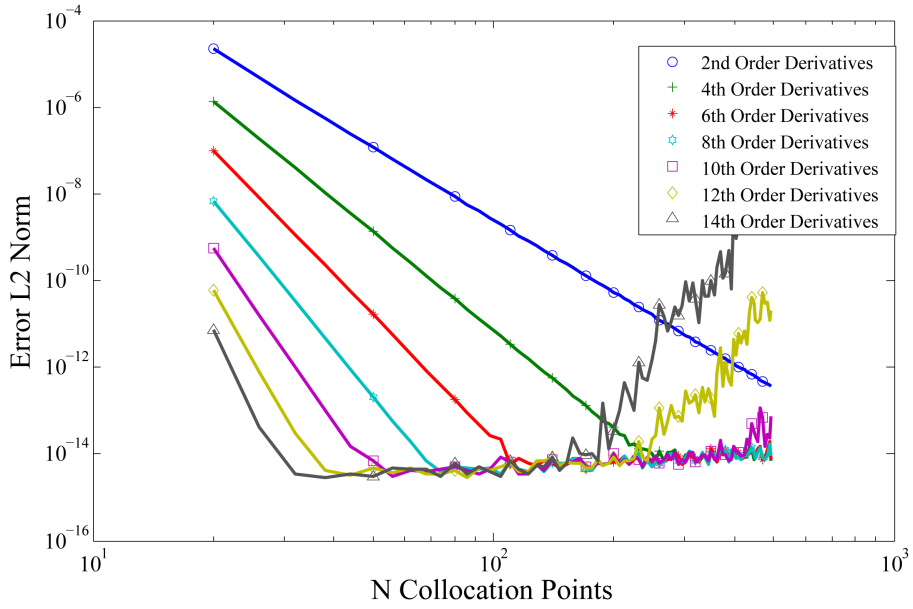

Figure B.3: Case 3. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

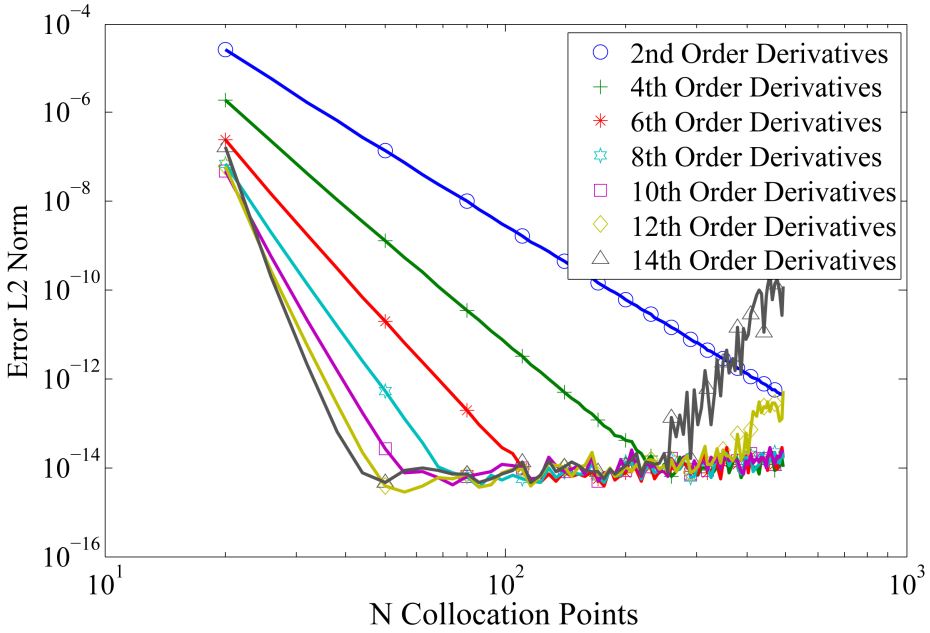

Figure B.4: Case 4. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

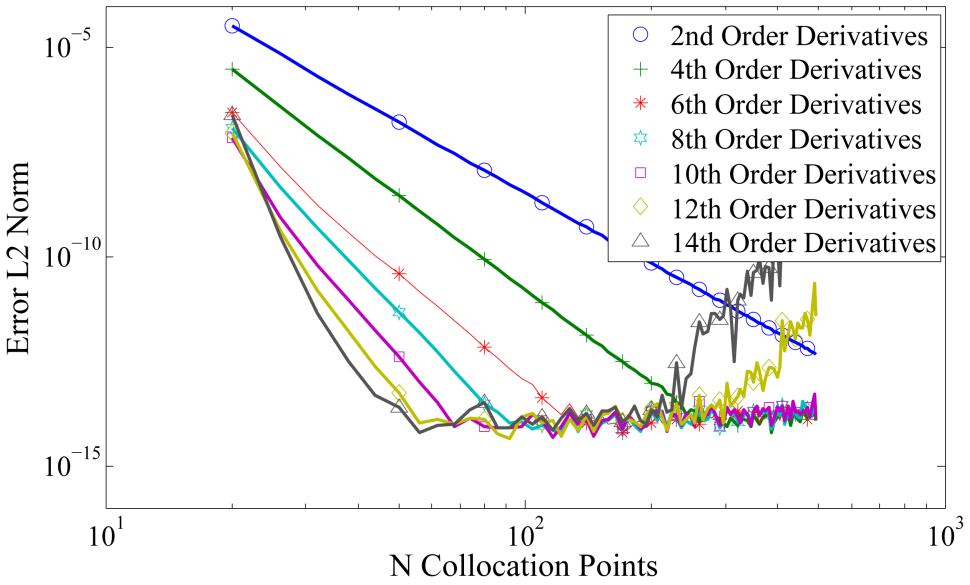

Figure B.5: Case 5. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

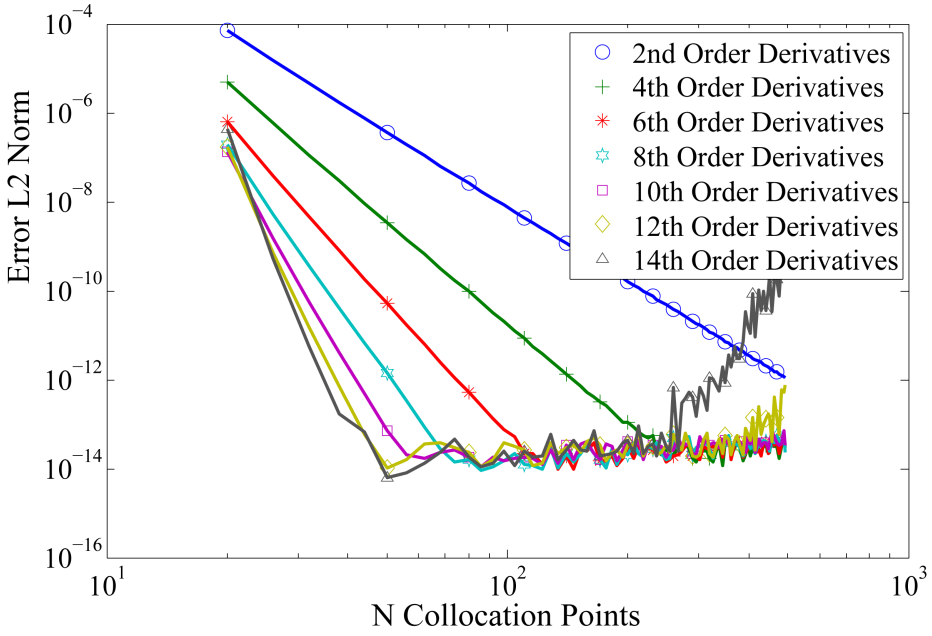

Figure B.6: Case 6. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

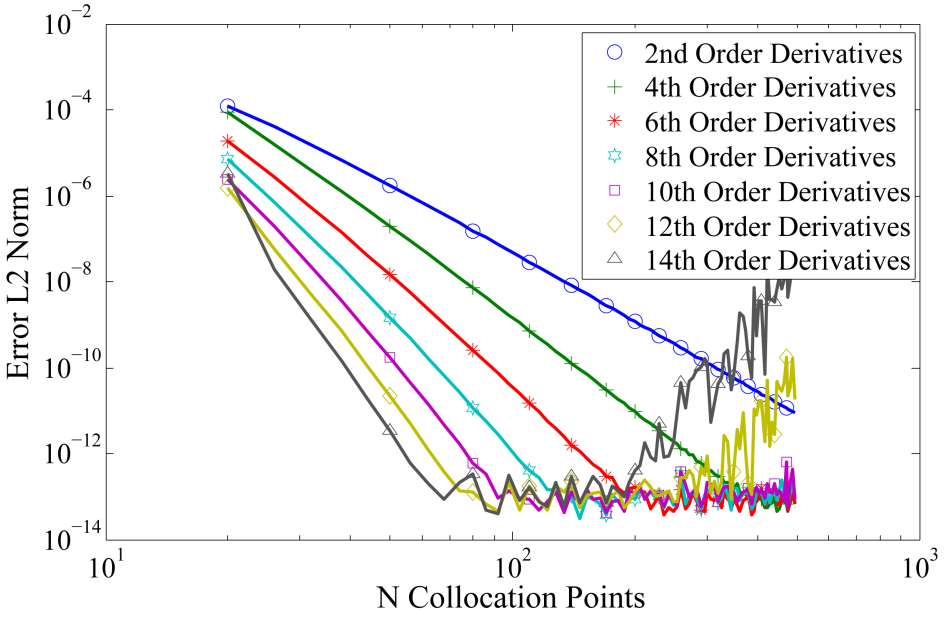

Figure B.7: Case 8. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.

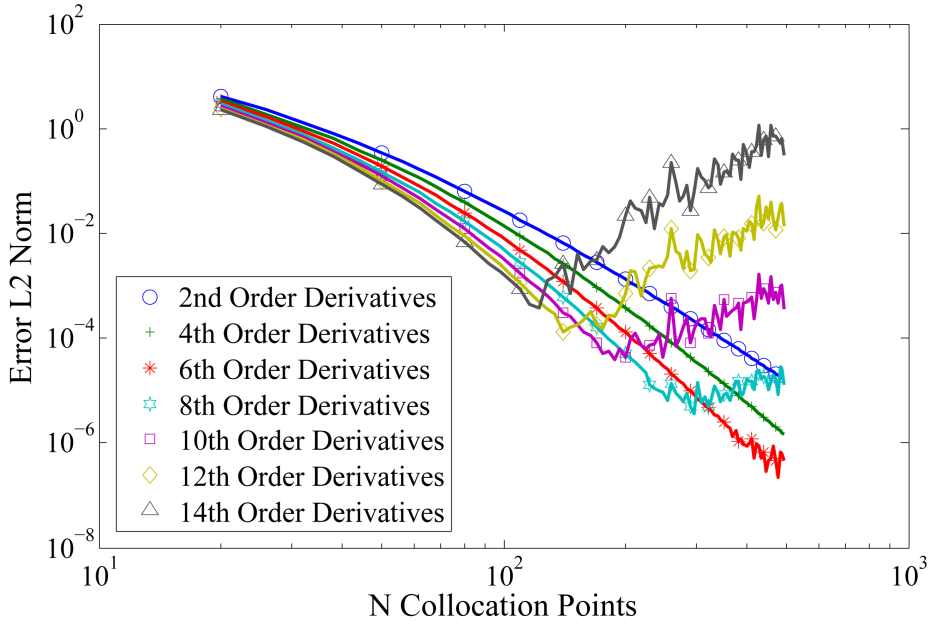

Figure B.8: Case 10. L^2 norm between analytical and numerical solutions using up to fourteenth order accurate derivatives.