A Modified Clenshaw-Curtis Quadrature Algorithm

by

Jeffrey Michael Barden

A project

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Applied Mathematics

April 24, 2013

Approved:

Professor Mayer Humi, Advisor

Professor Bogdan Vernescu, Department Head

Abstract

This project presents a modified method of numerical integration for a "well behaved" function over the finite interval $[-1, 1]$. Similar to the Clenshaw-Curtis quadrature rule, this new algorithm relies on expressing the integrand as an expansion of Chebyshev polynomials of the second kind. The truncated series is integrated term-by-term yielding an approximation for the integral of which we wish to compute. The modified method is then contrasted with its predecessor Clenshaw-Curtis, as well as the classical method of Gauss-Legendre in terms of convergence behavior, error analysis and computational efficiency. Lastly, illustrative examples are shown which demonstrate the dependence that the convergence has on the given function to be integrated.

Acknowledgments

I would like to sincerely thank my advisor, Professor Mayer Humi, for all of the guidance and inspiration he has offered me over the past two years. His teaching and wisdom have provided me with a strong education and understanding of many mathematical concepts. Second I would like to thank Professors Homer Walker, Sarah Olson and Marcus Sarkis for each supplying me with rigorous courses spanning across three different disciplines of numerical analysis. Their instruction has forever invoked my interest in this field and offered me the intellectual challenge and desire to investigate numerical mathematics. Lastly I am in a debt of gratitude to my family and friends who have supported me throughout graduate school and with each and every endeavor I have sought out to pursue. Without them none of this would be possible.

I dedicate this project to Sasha Lynn Neal, a very special woman in my life. She has always been there for me and sacrificed so much throughout my years of schooling, for that I am forever grateful.

Contents

Abstract i

Acknowledgments ii

Introduction 1

Classical Theory of Orthogonal Polynomials as a Basis for Series Expansions 2
- Orthogonal Polynomials . . . 2
- The Rodrigues Formula . . . 3
- Chebyshev Polynomials of the First Kind . . . 3
- Chebyshev Abscissas of the First Kind . . . 4
- Chebyshev Series Expansion of the First Kind . . . 4
- Remainder Approximation and Error Bound . . . 5
- A Simple Demonstration . . . 5
- Chebyshev Polynomials of the Second Kind . . . 6
- The Filippi Abscissas . . . 7
- Chebyshev Series Expansion of the Second Kind . . . 7
- The Legendre Polynomials . . . 8
- An Approximation of the Abscissas of the Legendre Polynomials . . . 8

Gauss-Legendre and Clenshaw-Curtis Quadrature 9
- The Idea of Numerical Quadrature . . . 9
- Gauss-Legendre Quadrature . . . 9
- Gauss-Legendre Error Estimate . . . 9
- A Derivation of Clenshaw-Curtis Quadrature . . . 9
- The $(n+1)$-point Clenshaw-Curtis Rule . . . 10
- Approximation of the Coefficients . . . 11
- An Error Bound for the Clenshaw-Curtis Rule . . . 11

Modified Clenshaw-Curtis Quadrature 13
- Derivation of the New Method . . . 13
- The $(n+1)$-point Modified Clenshaw-Curtis Rule . . . 14
- Gauss-Lobatto Quadrature . . . 14
- Gauss-Lobatto Error Estimate . . . 15
- An Error Bound for the Modified Clenshaw-Curtis Rule . . . 15

Findings and Results 16
- Error Convergence and Computational Efficiency Comparisons . . . 17
- Direct Comparison of CCGL and MCCGL . . . 22

Future Work 23

Appendix A: MATLAB Code 26

References 29

List of Figures

Figure 1: Chebyshev Polynomials of the First Kind 3
Figure 2: Chebyshev Polynomials of the Second Kind 7
Figure 3: The Legendre Polynomials 8

Error Convergence and Computational Efficiency Comparisons **17**
Figure 4: An Oscillatory Function 17
Figure 5: Absolute Value of an Oscillatory Function 18
Figure 6: Product of a Polynomial, Gaussian, Trigonometric and Inverse Trigonometric Function 18
Figure 7: Product of a Logarithmic Function and the Error Function 19
Figure 8: A Decaying Highly Oscillatory Function 19
Figure 9: Absolute Value of a Decaying Highly Oscillatory Function 20
Figure 10: An Oscillatory Exponential Function 20
Figure 11: Composition of a Polynomial and an Inverse Trigonometric Function 21

Direct Comparison of CCGL and MCCGL **22**
Figure 12: Product of an oscillatory function, the Gamma function and the Error Function . . 22
Figure 13: Composition of three Hyperbolic Functions 22

List of Tables

Table 1: Actual Error vs. Error Bound for the Clenshaw-Curtis Rule 12
Table 2: Actual Error vs. Error Bound for the Modified Clenshaw-Curtis Rule 16
Table 3: 15 Sample Integrals Computed by MCCGL 21
Table 4: Error Comparison of CCGL and MCCGL for figure 12 22
Table 5: Error Comparison of CCGL and MCCGL for figure 13 23

Introduction

One of the most common topics administered in a numerical analysis course or in standard textbooks is the idea of quadrature. Quadrature is the classical term reserved for numerically approximating the integral of a function $f(x)$ over some range of values $a \leq x \leq b$ within a specified error tolerance. The most familiar techniques are those that fall under the Newton-Cotes classification, such as the trapezoidal rule or Simpson's rule. These methods are easily implemented but do not converge in general unless the specified integrand f is analytic in a large region surrounding $[a, b]$ [15]. In addition, the Newton-Cotes formulas can become unstable due to their high vulnerability to accumulations in rounding error [2]. Usually once analysis of these integration techniques has been exhausted a more powerful method is proposed.

The general class of Gaussian quadrature rules is much less restricted in terms of convergence. These methods will always approach the true value for a given integral as long as the function f is continuous throughout the interval $[a, b]$. Moreover, the techniques attributed to Gauss lack vulnerability to accumulations in error and occurrences such as the Runge's phenomenon [11]. The reason behind this lies in the unequal spacing (or optimal spacing) of the sampled nodes which depend on the specified member of this quadrature family. Because the Gauss-like methods and the Newton-Cotes rules are interpolatory in nature, for specific degree polynomials they should exactly produce the result of performing integration on such a function. More specifically, for an $(n+1)$-node sampling the Newton-Cotes formulas will exactly integrate polynomials up to degree n, whereas Gauss-Legendre quadrature will exactly integrate polynomials up to degree $2n + 1$ [15] and Gauss-Lobatto up to degree $2n - 1$ [16]. The Gauss formulas require the computation of the abscissas of various orthogonal polynomials and their corresponding weights. The most efficient way of finding these values is by means of a tridiagonal Eigenvalue problem in which $\mathcal{O}(n^2)$ operations are sufficient [11].

One classical type of quadrature that seems to be overlooked in most entry level textbooks is Clenshaw-Curtis. This method is constructed based on the Chebyshev expansion of a continuous function. The given series is integrated term-by-term and its coefficients are approximated using the trapezoidal rule [1]. The original algorithm presented by its authors C. W. Clenshaw and A. R. Curtis outlined a way of pre-computing the weights and Chebyshev nodes which relies on the use of a discrete cosine transform. For large values of n this method proves to be just as computationally expensive as Gauss, requiring $\mathcal{O}(n^2)$ operations. After the advent of the fast Fourier transform (FFT) by Cooley and Tukey in 1965 [12], W. M. Gentleman of the University of Waterloo suggested using this new method for computing the nodes and weights needed in Clenshaw-Curtis. His idea was to convert the (DCT) to a discrete Fourier transform (DFT), once in this form the (FFT) can be directly applied [4]. Cooley and Tukey were able to prove that with their algorithm only $\mathcal{O}\big(n \log(n)\big)$ operations are required for computing the (DFT) [12]. This big $\mathcal{O}$ notation for the computational expense is titled linearithmic and resembles the number of floating point operations needed in some sorting algorithms found in computer languages such as Java and C++. Unlike the (DCT), the (FFT) is an extremely stable algorithm [5] and this attribute is always preferable in numerical analysis. Another useful feature noted by Gentleman was for a node sampling chosen such that $n = 2^p$ for some positive integer p, the already computed function evaluations at those nodes could be stored and reused. This resembles a property that is also true of the Newton-Cotes methods with equally spaced nodes.

Similar to Gauss and the Newton-Cotes rules, Clenshaw-Curtis is also an interpolatory quadrature technique. With this in mind, it should exactly integrate polynomials of degree at most n and can accomplish this feat in $\mathcal{O}\big(n \log(n)\big)$ operations. Depending on the analyticity of the integrand f, Gauss-Legendre may display a factor of 2 convergence advantage over Clenshaw-Curtis. If the function is not analytic in a large region containing $[a, b]$ then the convergence of both methods is more or less the

same [15]. This behavior was first observed by O'Hara and Smith in 1968 [3]. Thus, for most functions, Clenshaw-Curtis is just as effective as its counterpart Gauss-Legendre, but computationally it is orders of magnitude cheaper when using large values of n.

Expanding upon the Clenshaw-Curtis algorithm, an area of interest that this project explores is the use of Chebyshev polynomials of the second kind to derive an analogous automatic quadrature routine. In a similar fashion, we can express any continuous function as a series of Chebyshev polynomials of the second kind and integrate the series term-by-term producing a weighted sum as the approximation for the integral. Once again computing the coefficients for such an expansion proves to be a formidable task. Nonetheless we have some freedom to choose whichever routine best suits our needs. In general we want to designate a method for such an approximation that is guaranteed to converge. Clenshaw and Curtis adopted the use of the trapezoidal rule, which in general may not converge, throughout this paper we will commit ourselves to using the Gauss-Lobatto rule. It will be shown that for certain functions this modification to Clenshaw-Curtis will demonstrate improved convergence with the trade off of requiring more floating point operations. In other situations the convergence will be more or less the same as Clenshaw-Curtis and Gauss-Legendre. We will only examine the ideas of convergence, error analysis and computational efficiency for this new algorithm. Analysis of numerical stability, preconditioning, acceleration, adaptivity, multi-dimensions and endpoint or interior singularities are left for future work.

Classical Theory of Orthogonal Polynomials as a Basis for Series Expansions

It is the interest of this project to examine the convergence behavior of three interpolatory quadrature methods, two of which rely on an order n expansion of the integrand $f(x)$. Before we derive the Clenshaw-Curtis algorithm and its modification we will need some background information pertinent to the field of orthogonal polynomials.

Definition 1: Orthogonal Polynomials
The class of functions $\{P_k(x)\}$ defined over the interval $[a, b]$ are said to be orthogonal if they satisfy the inner product relation

$$\int_a^b P_k(x)P_\ell(x)w(x)dx = \delta_{k\ell}c_k, \tag{1}$$

where $w(x)$ is a weighting function, $\delta_{k\ell}$ is the Kronecker delta and

$$c_k = \int_a^b w(x)P_k^2(x)dx \quad [6]. \tag{2}$$

These special functions arise in the solutions of many mathematical and physical problems. They posses numerous useful properties which we will apply throughout the exhibition of this project. Each of the classical orthogonal polynomials that we will consider satisfy the general differential equation

$$Q(x)y'' + L(x)y' + \lambda_k y = 0 \tag{3}$$

where $Q(x)$ is at most a quadratic polynomial, $L(x)$ is a linear polynomial and λ_k is a real constant chosen so that the solutions of this equation bear no singularities. With these assumptions we can accordingly define

$$\lambda_k = -k\left(\frac{k-1}{2}Q'' + L'\right) \quad [8]. \tag{4}$$

The weight function under which the polynomial solutions to (3) are orthogonal is defined as the following ratio

$$w(x) = \frac{R(x)}{Q(x)} \tag{5}$$

where

$$R(x) = \exp\left(\int \frac{L(x)}{Q(x)}dx\right) \quad [8]. \tag{6}$$

This leads us to the generalized solution of (3)

Definition 2: The Rodrigues Formula
The polynomial solutions to equation (3) can be found using the following formula

$$p_k(x) = \frac{1}{e_k w(x)} \frac{d^k}{dx^k}\left(w(x)[Q(x)]^k\right) \quad [8] \tag{7}$$

where e_k is a standardization constant.
With this we will construct three sets of orthogonal polynomials for later use in this paper.

Definition 3: Chebyshev Polynomials of the First Kind
The Chebyshev polynomials of the first kind are defined by the trigonometric formula

$$T_k(x) = \cos\big(k \arccos(x)\big), \quad x \in [-1, 1], \quad k \geq 0 \tag{8}$$

and can be generated using the three-term recurrence relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \tag{9}$$

with $T_0(x) = 1$ and $T_1(x) = x$. This set of polynomials is orthogonal with respect to the weight function $w(x) = (1 - x^2)^{-1/2}$ and forms a complete basis in $L^2[-1, 1]$ [7].
These polynomials appear as the solution of the differential equation

$$(1 - x^2)y'' - xy' + k^2 y = 0 \tag{10}$$

and from the general Rodrigues formula as defined in (7) we have an alternative way of constructing the k^{th} polynomial, namely

$$T_k(x) = \frac{(-1)^k \sqrt{\pi}\sqrt{1 - x^2}}{2^k\big(k - \frac{1}{2}\big)!} \frac{d^k}{dx^k}\left[(1 - x^2)^{k-1/2}\right] \quad [14]. \tag{11}$$

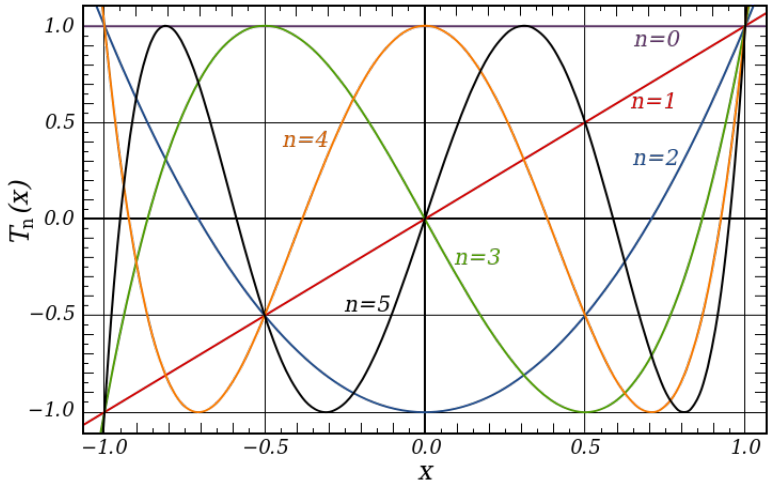


Fig. 1: The first six Chebyshev polynomials $T_k(x)$

Definition 4: Chebyshev Abscissas of the First Kind
The Chebyshev polynomial $T_k(x)$ of degree $k \geq 1$ has k simple roots in $[-1, 1]$ at the points

$$x_{k,j} = \cos\left(\frac{(2j-1)\pi}{2k}\right), \quad \text{for each } j = 1, 2, ..., k. \tag{12}$$

Furthermore, $T_k(x)$ assumes its absolute extrema at the points

$$x'_{k,j} = \cos\left(\frac{j\pi}{k}\right) \quad \text{with} \quad T_k(x'_{k,j}) = (-1)^j, \quad \text{for each} \quad j = 0, 1, ..., k \quad [10]. \tag{13}$$

With this information we can now introduce the desired series expansion upon which the Clenshaw-Curtis algorithm is formulated.

Theorem 1: The Chebyshev Series Expansion of the First Kind
For any function $f(x) \in C^1[-1, 1]$, there exists a unique series expansion expressed in terms of Chebyshev polynomials of the first kind which converges uniformly on $[-1, 1]$ to f. This expansion is written in the form

$$f(x) = {\sum_{k=0}^{\infty}}' c_k T_k(x) = \frac{c_0}{2} T_0(x) + \sum_{k=1}^{\infty} c_k T_k(x), \quad -1 \leq x \leq 1 \tag{14}$$

where the prime notation implies the first term of the sum is to be halved. Taking into account the orthogonality of $T_k(x)$, the coefficients are uniquely determined by means of the inner product

$$c_k = \frac{\displaystyle\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\, dx}{\displaystyle\int_{-1}^{1} \frac{T_k^2(x)}{\sqrt{1-x^2}}\, dx} \quad [2]. \tag{15}$$

Making a change of variable, $x = \cos\theta$ and using the identity that $T_k(\cos\theta) = \cos(k\theta)$, the above formula reduces to

$$c_k = \frac{2}{\pi} \int_0^{\pi} f(\cos\theta)\cos(k\theta) d\theta \quad [1]. \tag{16}$$

The Chebyshev series is known as a generalized Fourier expansion and this is easily seen using the previously stated change of variable, which transforms (14) into an equivalent form

$$f(x) = \sum_{k=0}^{\infty}{}' c_k \cos(k\theta)$$

of which we define as the Fourier cosine series for f.

From a computational standpoint, the upper value of the summation is limited by the processing capability of the machine we are operating with. This in turn leads to a truncation error of the expansion for f. It was observed by Lanczos that as a result of truncating the series in (14), the remainder will look a lot like the first $T_k(x)$ that is neglected, that is

$$R_n^T(x) = \sum_{k=n+1}^{\infty} c_k T_k(x) \approx c_{n+1} T_{n+1}(x). \tag{17}$$

This approximation of the tail-end of the series holds true if $c_{n+1} \neq 0$ and if the coefficients c_k are rapidly converging to zero. For a proof of this conjecture the reader is referred to [13]. Lanczos was also able to show that the error E_n^T that results from dropping terms beyond $T_n(x)$ is bounded in the following way

$$\left|E_n^T\right| = \left|f(x) - \left(\frac{c_0}{2} + c_1 T_1(x) + \cdots + \frac{c_n}{2} T_n(x)\right)\right| = \left|\sum_{k=n+1}^{\infty} c_k T_k(x)\right| \leq \sum_{k=n+1}^{\infty} |c_k| \quad [6] \tag{18}$$

where the last inequality is a consequence of the fact that $|T_k(x)| \leq 1$ for all $x \in [-1, 1]$. By this construction one can see that the n^{th} order Chebyshev expansion of f should be exact for polynomials of degree less than or equal to n, since each $c_k = 0$ for all $k \geq n+1$. Now consider a truncated power series representation of our function f. By definition, this function approximation is expressed as an order n polynomial which must agree with some linear combination of Chebyshev polynomials T_0 through T_n. This implies the following equivalence relationship

$$\sum_{k=0}^{n} a_k x^k = \sum_{k=0}^{n}{}'' c_k T_k(x) \quad \text{for all} \quad x \in [-1, 1].$$

It has been shown by Hamming [6] that at the n^{th} stage of such an expansion

$$a_n x^n \to \frac{a_n}{2^{n-1}} T_n(x) + \text{lower order terms.}$$

It is for this reason that in general, the Chebyshev series converges much more rapidly than the corresponding power series. Essentially, a power series, or for that matter an n^{th} order Taylor expansion only provides local convergence surrounding a single point; whereas the Chebyshev series will converge to f locally to a given interval $[a, b]$.

A Simple Demonstration:

Let us now choose a convenient function, say $f = \arccos(x)$, to demonstrate how the Chebyshev expansion works. Using (16) the necessary coefficients are

$$c_0 = \frac{2}{\pi} \int_0^{\pi} \theta d\theta = \pi$$

and for $k \geq 1$

$$\begin{aligned} c_k &= \frac{2}{\pi}\int_0^\pi \theta\cos(k\theta)d\theta \\ &= \frac{2}{\pi}\left[\frac{\theta\sin(k\theta)}{k}\Big|_0^\pi - \int_0^\pi \frac{\sin(k\theta)}{k}d\theta\right] \\ &= \frac{2}{\pi}\left[\frac{\cos(k\theta)}{k^2}\right]_0^\pi = \frac{2}{\pi}\left(\frac{(-1)^k-1}{k^2}\right) \quad \text{*Vanishes for even } k \end{aligned}$$

which implies

$$c_{2k} = 0 \;\text{ and }\; c_{2k-1} = \frac{-4}{\pi(2k-1)^2}.$$

Therefore by (14) the resultant Chebyshev expansion is given by

$$\arccos(x) = \frac{\pi}{2} - \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{T_{2k-1}(x)}{(2k-1)^2}.$$

Now say we wish to approximate the given function f to order 10, the remainder will then be

$$R_{10}^T(x) \approx \frac{-4T_{21}(x)}{441\pi}$$

and the truncation error can be bounded by

$$|E_{10}^T| \leq \sum_{k=11}^{\infty}|c_{2k-1}| = \frac{4}{\pi}\sum_{11}^{\infty}\frac{1}{(2k-1)^2} \doteq 0.0318045549110114.$$

As described in the abstract, the goal of this project is to formulate an alternative series expansion for the function f which can be integrated term-by-term. We will do so using an expansion of Chebyshev polynomials of the second kind.

Definition 5: Chebyshev Polynomials of the Second Kind
The Chebyshev polynomials of the second kind are defined by the formula

$$U_k(x) = \frac{\left(x+\sqrt{1-x^2}\right)^{k+1} - \left(x-\sqrt{1-x^2}\right)^{k+1}}{2\sqrt{1-x^2}} = \frac{\sin\left((k+1)\arccos(x)\right)}{\sqrt{1-x^2}}, \quad x\in[-1,1], \quad k\geq 0 \tag{19}$$

and can be generated using the three-term recurrence relation

$$U_{k+1}(x) = 2xU_k(x) - U_{k-1}(x) \tag{20}$$

with $U_0(x) = 1$ and $U_1(x) = 2x$. This set of polynomials is orthogonal with respect to the weight function $w(x) = \sqrt{1-x^2}$ and also forms a complete basis in $L^2[-1,1]$ [14].

These polynomials emerge as the solution of the differential equation

$$(1-x^2)y'' - 3xy' + k(k+2)y = 0 \tag{21}$$

and posses a Rodrigues representation of

$$U_k(x) = \frac{(-1)^k(k+1)\sqrt{\pi}}{2^{k+1}\left(k+\frac{1}{2}\right)!\sqrt{1-x^2}} \frac{d^k}{dx^k}\left[(1-x^2)^{k+1/2}\right] \quad [14]. \tag{22}$$

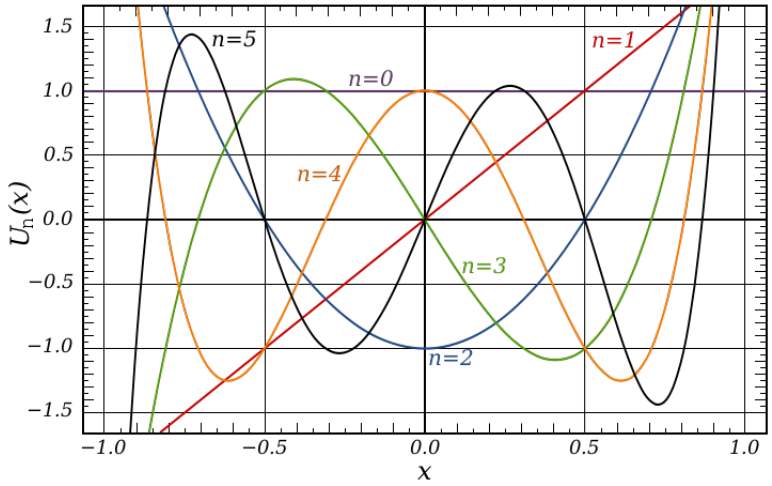


Fig. 2: The first six Chebyshev polynomials $U_k(x)$

Definition 6: The Filippi Abscissas
The Chebyshev polynomial $U_k(x)$ of degree $k \geq 1$ has k simple roots in $[-1,1]$ at the points

$$x_{k,j} = \cos\left(\frac{j\pi}{k+1}\right), \quad \text{for each } j = 1, 2, ..., k \quad [14]. \tag{23}$$

With this we are now able to state the following theorem which establishes the desired series expansion that our modified Clenshaw-Curtis quadrature technique will rely on.

Theorem 2: The Chebyshev Series Expansion of the Second Kind
For any function $f(x) \in C^1[-1,1]$, there exists a unique series expansion expressed in terms of Chebyshev polynomials of the second kind which converges uniformly on $[-1,1]$ to f. This expansion is written in the form

$$f(x) = {\sum_{k=0}^{\infty}}' \psi_k U_k(x) = \frac{\psi_0}{2}U_0(x) + \sum_{k=1}^{\infty} \psi_k U_k(x), \quad -1 \leq x \leq 1 \tag{24}$$

where the coefficients ψ_k are uniquely determined using the inner product

$$\psi_k = \frac{\int_{-1}^{1} f(x)U_k(x)\sqrt{1-x^2}\,dx}{\int_{-1}^{1} U_k^2(x)\sqrt{1-x^2}\,dx} \quad [2]. \tag{25}$$

Part of our modified Clenshaw-Curtis method will utilize the Gauss-Lobatto quadrature technique. This algorithm relies on the Legendre polynomials and the corresponding roots of their derivatives. We will also be closely comparing both methods to Gauss-Legendre quadrature which also uses these same polynomials and their respective roots. To proceed we will need the following two definitions.

Definition 7: Legendre Polynomials
The Legendre polynomials are a set of orthogonal functions over the interval $[-1, 1]$ *with respect to the weight function* $w(x) = 1$. *They are ascertained by the Rodrigues formula*

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k}{dx^k} \left[(x^2 - 1)^k \right] \quad [14]. \tag{26}$$

Similar to the Chebyshev polynomials, the Legendre polynomials form a complete basis set in $L^2[-1, 1]$. These functions can also be generated by means of Bonnet's recursion formula

$$(k + 1)P_{k+1}(x) = (2k + 1)xP_k(x) - kP_{k-1}(x) \tag{27}$$

in which we must multiply each $P_k(x)$ by the normalizing factor $\frac{2}{2k+1}$ to allow the coefficients to match the polynomials given by (26).

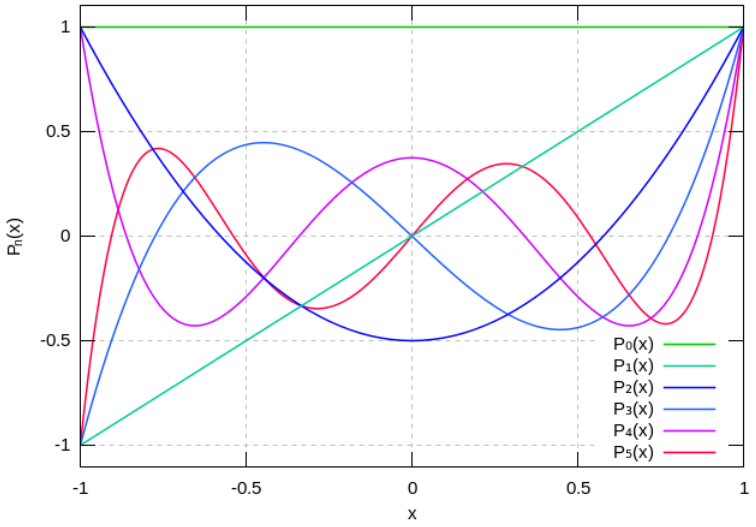


Fig. 3: The first six Legendre polynomials $P_k(x)$

Although there is no closed expression for the roots of the Legendre polynomials we can approximate there value in the following way

Definition 8: Abscissas of the Legendre Polynomials
The Legendre polynomial $P_k(x)$ *of degree* $k \geq 1$ *has* k *simple roots* $x_{k,1} > x_{k,2} > \cdots > x_{k,k}$ *where*

$$x_{k,j} = \left(1 - \frac{1}{8k^2} + \frac{1}{8k^3}\right) \cos\left(\frac{4j - 1}{4k + 2}\pi\right) + \mathcal{O}(k^{-4}) \quad [14]. \tag{28}$$

This concludes the necessary framework upon which the remainder of the analysis in this project is based.

Gauss-Legendre and Clenshaw-Curtis Quadrature

Throughout the analysis presented in this project we will make use of two numerical integration techniques, Gauss-Legendre and Clenshaw-Curtis, we shall examine the former first. In general a quadrature formula is an approximation of a definite integral for a given function. It is expressed as a weighted sum of function evaluations at some conveniently chosen set of points that lie in the interval of integration. An $n-$point quadrature rule is designed to yield the exact value of which we would obtain from direct integration of a polynomial of degree at most n. To achieve this we require a set of nodes to be sampled, denoted by x_i and the corresponding weights w_i for $i = 1, 2, ..., n$. The most generic quadrature formula is then written as

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i). \tag{29}$$

For Gauss-Legendre quadrature we choose the conventional interval of integration $[-1, 1]$ and define each Gauss node x_i as the i^{th} root of the n^{th} Legendre polynomial $P_n(x)$ (See Definition 7). The weights that correspond to these nodes are given by

$$w_i = \frac{2}{(1-x_i^2)[P_n'(x_i)]^2} = \frac{2(1-x_i^2)}{(n+1)^2[P_{n+1}(x_i)]^2} \quad [16] \tag{30}$$

and for the $n-$point Gauss-Legendre rule this will allow for exact integration of any polynomial up to degree $2n-1$. The error estimate for this method of integration is well known and we will refer the reader to Abramowitz & Stegun [9] for a derivation and proof.

Theorem 3: Gauss-Legendre Error Estimate
For any function $f \in C^{2n}[-1, 1]$, the error as a result of performing Gauss-Legendre quadrature to approximate the integral of f over $[-1, 1]$ can be estimated by

$$\int_{-1}^{1} f(x)dx - \sum_{i=1}^{n} w_i f(x_i) \approx \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi), \quad \text{for some } \xi \in (-1, 1) \quad [16]. \tag{31}$$

A Derivation of Clenshaw-Curtis Quadrature:
The second, and more important type of quadrature that we consider is that of Clenshaw-Curtis. As presented in their original 1960 paper, Clenshaw and Curtis came up with a fundamental method that relies on direct integration of the n^{th} order Chebyshev expansion of the first kind for $f(x)$ and makes use of some unique properties of the Chebyshev polynomials. Let us work through a derivation of this algorithm beginning with the integral

$$\begin{aligned}\int_{-1}^{1} f(x)dx &= \int_{-1}^{1} {\sum_{k=0}^{\infty}}' c_k T_k(x)dx \\ &= \int_{-1}^{1} \lim_{n\to\infty} {\sum_{k=0}^{n}}' c_k T_k(x)dx.\end{aligned}$$

Because the series converges uniformly we can interchange summation and integration giving us

$$= \int_{-1}^{1} \left(\frac{c_0 T_0(x)}{2} + \lim_{n\to\infty} \sum_{k=1}^{n} c_k T_k(x) \right) dx = c_0 + \lim_{n\to\infty} \sum_{k=1}^{n} c_k \int_{-1}^{1} T_k(x)dx. \tag{32}$$

To simplify the expression in (32) we can use the following three properties

$$\int T_k(x)dx = \frac{1}{2}\left(\frac{T_{k+1}(x)}{k+1} - \frac{T_{k-1}(x)}{k-1}\right) + const. = \frac{kT_{k+1}(x)}{k^2-1} - \frac{xT_k(x)}{k-1} + const. \tag{33}$$

$$T_k(1) = 1 \quad \text{and} \quad T_k(-1) = (-1)^k \quad [14]. \tag{34}$$

This leads to

$$\begin{aligned}
\int_{-1}^{1} f(x)dx &= c_0 + \sum_{k=1}^{\infty} c_k \int_{-1}^{1} T_k(x)dx \\
\text{*Application of (33)} \quad &= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{kT_{k+1}(1)}{k^2-1} - \frac{T_k(1)}{k-1} - \frac{kT_{k+1}(-1)}{k^2-1} - \frac{T_k(-1)}{k-1}\right) \\
\text{*Application of (34)} \quad &= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{k}{k^2-1} - \frac{1}{k-1} - \frac{k(-1)^{k+1}}{k^2-1} - \frac{(-1)^k}{k-1}\right) \\
&= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{k(k-1) - k(k-1)(-1)^{k+1} - (k^2-1) - (k^2-1)(-1)^k}{(k^2-1)(k-1)}\right) \\
&= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{k(k-1)\left(1+(-1)^k\right) - (k^2-1)\left(1+(-1)^k\right)}{(k^2-1)(k-1)}\right) \\
&= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{\left(1+(-1)^k\right)\left(k(k-1) - (k^2-1)\right)}{(k^2-1)(k-1)}\right) \\
&= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{\left(1+(-1)^k\right)(1-k)}{(k^2-1)(k-1)}\right) \\
\text{*Vanishes for odd } k \quad &= c_0 + \sum_{k=1}^{\infty} c_k \left(\frac{1+(-1)^k}{1-k^2}\right) \\
&= c_0 + \sum_{k=1}^{\infty} \frac{2c_{2k}}{(1-4k^2)}.
\end{aligned}$$

Therefore the $(n+1)$-point Clenshaw-Curtis rule is

$$\int_{-1}^{1} f(x)dx \approx \int_{-1}^{1} {\sum_{k=0}^{n}}'' c_k T_k(x)dx = c_0 + \sum_{k=1}^{n/2-1} \frac{2c_{2k}}{1-4k^2} + \frac{c_n}{1-n^2} \quad [1] \tag{35}$$

where n must be an even integer greater than or equal to 4. If we insert the coefficients c_{2k} as defined in (16) the formula becomes

$$\boxed{\int_{-1}^{1} f(x)dx \approx \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)d\theta + \frac{4}{\pi}\sum_{k=1}^{n/2-1}\left(\frac{\int_0^{\pi} f(\cos\theta)\cos(2k\theta)d\theta}{1-4k^2}\right) + \frac{2}{\pi}\,\frac{\int_0^{\pi} f(\cos\theta)\cos(n\theta)d\theta}{1-n^2}}. \tag{36}$$

Based on observation of the coefficients in (16), it may be cumbersome to compute such integrals, so we rely on the trapezoidal rule to approximate their value. We will use the points of extrema for the n^{th} Chebyshev polynomial $T_n(x)$, given in (13), as the sampling points for this approximation.

$$\begin{aligned}\int_0^\pi f(\cos\theta)\cos(k\theta)d\theta &\approx \frac{\pi}{2n}\left[f(x'_{n,0})\cos(kx'_{n,0}) + 2\sum_{j=1}^{n-1} f(x'_{n,j})\cos(kx'_{n,j}) + f(x'_{n,n})\cos(kx'_{n,n})\right] \\ &= \frac{\pi}{2n}\left[f(1) + (-1)^k f(-1) + 2\sum_{j=1}^{n-1} f\left(\cos\left(\frac{\pi j}{n}\right)\right)\cos\left(\frac{\pi kj}{n}\right)\right]\end{aligned}$$

which implies

$$\frac{2}{\pi}\int_0^\pi f(\cos\theta)\cos(k\theta)d\theta \approx \frac{2}{n}{\sum_{j=0}^{n}}'' f\left(\cos\left(\frac{\pi j}{n}\right)\right)\cos\left(\frac{\pi kj}{n}\right) \quad [1] \tag{37}$$

where the double prime notation signifies the first and last term of the sum are to be halved. It was originally suggested by Clenshaw and Curtis [1] that this summation be computed using a discrete cosine transform (DCT). In 1972, twelve years after the publication of their paper, Gentleman [5] suggested the use of Cooley and Tukey's algorithm, the fast Fourier transform (FFT). Today it is customary to pre-compute the weights

$$w_{n,j} = \frac{2}{n}\cos\left(\frac{\pi kj}{n}\right)$$

and evaluate the summation in (37) using this technique. By doing so, the computation that would have otherwise required $\mathcal{O}(n^2)$ operations now can be performed in only $\mathcal{O}(n\log n)$ operations. We will omit further explanation of both these transform methods and refer the reader to [12]. Long before the Chebyshev series was being considered in the use of numerical integration, Hungarian mathematician Lipót Fejér suggested the use of the Chebyshev Abscissas and extrema on the open interval $(-1, 1)$ for approximating the integrals found in (16) [15]. Incorporating this idea in the procedure to find the expression of (37) would have lead to slightly different results. In any case, let it be known that there are two variations of Clenshaw-Curtis quadrature known as Fejér's first and second rules.

An Error Bound for the Clenshaw-Curtis Rule:

We will next present a remainder term for the truncated Chebyshev series, similar to that of (17). Upon interpolating our given function at the zeros of $T_{n+1}(x)$ we can express f in the form

$$f(x) = {\sum_{k=0}^{n}}'' c_k T_k(x) + \frac{T_{n+1}(x)}{2^n(n+1)!}f^{n+1}(\xi) \quad [7] \tag{38}$$

where as usual $|\xi| < 1$. Let us define a bound for the $(n+1)^{\text{st}}$ derivative of f evaluated at some $\xi(x)$

$$\left|f^{n+1}(\xi)\right| \le M, \quad \text{where} \quad M = \max_{-1\le x\le 1}\left|f^{n+1}(x)\right|. \tag{39}$$

Now using (38) we can generate an approximate error bound for the integral of f, assuming that $f \in C^{n+1}[-1, 1]$. We begin by integrating both sides of the expression

$$\int_{-1}^{1} f(x)dx = {\sum_{k=0}^{n}}'' c_k \int_{-1}^{1} T_k(x)dx + \frac{1}{2^n(n+1)!}\int_{-1}^{1} T_{n+1}(x)f^{n+1}(\xi)dx$$

appealing to (35) this becomes

$$\int_{-1}^{1} f(x)dx - \left(c_0 + \sum_{k=1}^{n/2-1} \frac{2c_{2k}}{1-4k^2} + \frac{c_n}{1-n^2} \right) = \frac{1}{2^n(n+1)!} \int_{-1}^{1} T_{n+1}(x) f^{n+1}(\xi)dx.$$

Now taking absolute value to both sides we obtain

$$\begin{aligned}
\left| \int_{-1}^{1} f(x)dx - \left(c_0 + \sum_{k=1}^{n/2-1} \frac{2c_{2k}}{1-4k^2} + \frac{c_n}{1-n^2} \right) \right| &= \left| \frac{1}{2^n(n+1)!} \int_{-1}^{1} T_{n+1}(x) f^{n+1}(\xi)dx \right| \\
&\leq \frac{1}{2^n(n+1)!} \int_{-1}^{1} \left| T_{n+1}(x) f^{n+1}(\xi) \right| dx \\
\text{*Use of Cauchy-Schwarz inequality} \qquad &\leq \frac{1}{2^n(n+1)!} \left(\int_{-1}^{1} \left| T_{n+1}(x) \right|^2 dx \right)^{1/2} \left(\int_{-1}^{1} \left| f^{n+1}(\xi) \right|^2 dx \right)^{1/2} \\
\text{*Use of (39) and } |T_{n+1}(x)| \leq 1 \qquad &\leq \frac{\sqrt{2}}{2^n(n+1)!} \left(\int_{-1}^{1} M^2 dx \right)^{1/2} \\
&= \frac{M}{2^{n-1}(n+1)!}.
\end{aligned}$$

Therefore an error bound for the Clenshaw-Curtis rule is

$$\boxed{\left| I(f) - I_{cc}(f,n) \right| \leq \frac{M}{2^{n-1}(n+1)!}}. \tag{40}$$

Let us examine the integral of a simple function and test the validity of this error bound. Choose $f(x) = e^x$, this implies

$$M = \max_{-1 \leq x \leq 1} |e^x| = e$$

and

$$I(f) = \int_{-1}^{1} e^x dx = e - \frac{1}{e} \doteq 2.350402387287603.$$

Table 1: Actual Error vs. Error Bound for the Clenshaw-Curtis Rule

n	Clenshaw-Curtis Approximation	Absolute Error	Error Bound
4	2.350375376931479	$2.701035612373559 \times 10^{-5}$	0.002831543571312
6	2.350402366696299	$2.059130332909831 \times 10^{-8}$	$1.685442601971135 \times 10^{-5}$
8	2.350402387267139	$2.046407487910074 \times 10^{-11}$	$5.852231256844216 \times 10^{-8}$
10	2.350402387287584	$1.865174681370263 \times 10^{-14}$	$1.330052558373686 \times 10^{-10}$

The inequality given in (40) may not be the best error bound, but nonetheless it is a bound which works for sufficiently differentiable integrands $f(x)$.

Modified Clenshaw-Curtis Quadrature

Now that we have successfully derived the original algorithm, we can proceed forward with a modification to this method. We shall begin with the Chebyshev series of the second kind as defined in (24),

$$f(x) = {\sum_{k=0}^{\infty}}' \psi_k U_k(x).$$

For sufficiently chosen functions which satisfy the conditions of Theorem 2, this series will converge uniformly to f. That being said we can integrate both sides yielding

$$\begin{aligned}\int_{-1}^{1} f(x)dx &= \int_{-1}^{1} {\sum_{k=0}^{\infty}}' \psi_k U_k(x)dx \\ &= \int_{-1}^{1} \lim_{n\to\infty} {\sum_{k=0}^{n}}' \psi_k U_k(x)dx\end{aligned}$$

and because of the uniform convergence of the series we are allowed to interchange summation and integration, thus reducing the above expression to

$$= \int_{-1}^{1} \left(\frac{\psi_0 U_0(x)}{2} + \lim_{n\to\infty} \sum_{k=1}^{n} \psi_k U_k(x) \right) dx = \psi_0 + \lim_{n\to\infty} \sum_{k=1}^{n} \psi_k \int_{-1}^{1} U_k(x)dx. \tag{41}$$

Let us note that there exists a convenient derivative relationship between Chebyshev polynomials of the first and second kind which states

$$U_k(x) = \frac{1}{k+1} \frac{d}{dx}\Big(T_{k+1}(x)\Big) \quad [14], \tag{42}$$

applying this we can re-express (41) in the following way.

$$\begin{aligned}\int_{-1}^{1} f(x)dx &= \psi_0 + \sum_{k=1}^{\infty} \frac{\psi_k}{k+1} \int_{-1}^{1} \frac{d}{dx}\Big(T_{k+1}(x)\Big)dx \\ \text{*By the Fundamental Theorem of Calculus} \quad &= \psi_0 + \sum_{k=1}^{\infty} \frac{\psi_k}{k+1}\Big(T_{k+1}(1) - T_{k+1}(-1)\Big) \\ \text{*Application of (34)} \quad &= \psi_0 + \sum_{k=1}^{\infty} \frac{\psi_k}{k+1}\Big(1 - (-1)^{k+1}\Big) \\ \text{*Vanishes for odd } k \quad &= \psi_0 + \sum_{k=1}^{\infty} \frac{\psi_k\left(1 + (-1)^k\right)}{k+1} \\ &= \psi_0 + \sum_{k=1}^{\infty} \frac{2\psi_{2k}}{2k+1}.\end{aligned}$$

Therefore the modified quadrature rule in its $(n+1)$-point truncated form is expressed as

$$\int_{-1}^{1} f(x)dx = \psi_0 + \sum_{k=1}^{n/2-1} \frac{2\psi_{2k}}{2k+1} + \frac{\psi_n}{n+1} \tag{43}$$

for any even integer $n \geq 4$. Now using (25) we can determine the coefficients ψ_k by using the change of variable $x = \cos\theta$ and the trigonometric definition of U_k as defined in (19). The inner product (U_k, U_k) with respect to the weight function will become

$$\begin{aligned}
\int_{-1}^{1} U_k^2(x)\sqrt{1-x^2}\,dx &= -\int_{\pi}^{0} \frac{\sin^2\Big((k+1)\arccos\big(\cos\theta\big)\Big)}{1-\cos^2\theta}\sin^2\theta\,d\theta \\
&= \int_0^{\pi} \sin^2\big((k+1)\theta\big)d\theta \\
&= \frac{1}{2}\int_0^{\pi} 1-\cos\big((2k+2)\theta\big)d\theta \\
&= \frac{1}{2}\left[\theta - \frac{1}{2k+2}\sin\big((2k+2)\theta\big)\right]_{\theta=0}^{\theta=\pi} = \frac{\pi}{2}.
\end{aligned}$$

Similarly the inner product (f, U_k) with respect to the weight function will become

$$\begin{aligned}
\int_{-1}^{1} f(x)U_k(x)\sqrt{1-x^2}\,dx &= -\int_{\pi}^{0} f(\cos\theta)\frac{\sin\Big((k+1)\arccos\big(\cos\theta\big)\Big)}{\sqrt{1-\cos^2\theta}}\sqrt{1-\cos^2\theta}\sin\theta\,d\theta \\
&= \int_0^{\pi} f(\cos\theta)\sin\big((k+1)\theta\big)\sin\theta\,d\theta.
\end{aligned}$$

Therefore the coefficients from (25) simplify to

$$\psi_k = \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)\sin\big((k+1)\theta\big)\sin\theta\,d\theta. \tag{44}$$

Plugging the result of (44) into the right hand side of (43) we obtain our final form of the modified Clenshaw-Curtis method. For an $n+1$ node sampling within the interval $[-1, 1]$, we can approximate the integral of f over this same interval using the following formula

$$\boxed{\int_{-1}^{1} f(x)dx \approx \frac{2}{\pi}\int_0^{\pi} f(\cos\theta)\sin^2\theta\,d\theta + \frac{4}{\pi}\sum_{k=1}^{n/2-1}\left(\frac{\int_0^{\pi} f(\cos\theta)\sin\big((2k+1)\theta\big)\sin\theta\,d\theta}{2k+1}\right) + \frac{2}{\pi}\,\frac{\int_0^{\pi} f(\cos\theta)\sin\big((n+1)\theta\big)\sin\theta\,d\theta}{n+1}}. \tag{45}$$

In general, it will be required that the coefficients ψ_k be numerically approximated for efficiency purposes. Instead of using the trapezoidal rule as seen in the method of Clenshaw and Curtis, we will now approximate the coefficients to higher precision using the Gauss-Lobatto rule.

Definition 9: Gauss-Lobatto Quadrature:
The $n-$point Gauss-Lobatto rule is defined as

$$\int_{-1}^{1} f(x)dx \approx \frac{2}{n(n-1)}\big[f(1)+f(-1)\big] + \sum_{i=2}^{n-1} w_i f(x_i) \quad [16] \tag{46}$$

where x_i is the $(i-1)^{st}$ root of $P'_{n-1}(x)$, the derivative of the $(n-1)^{st}$ Legendre polynomial. The corresponding weights are given by

$$w_i = \begin{cases} \dfrac{2}{n(n-1)(P_{n-1}(x_i))^2}, & \text{for } \; i = 2, ..., n-1 \\ \dfrac{2}{n(n-1)}, & \text{for } \; i = 1, \;\; \text{or } \;\; i = n \end{cases} . \tag{47}$$

Theorem 4: Gauss-Lobatto Error Estimate
For any function $f \in C^{2n-2}[-1,1]$, the error as a result of performing Gauss-Lobatto quadrature to approximate the integral of f over $[-1,1]$ can be estimated by

$$E_n \approx -\frac{n(n-1)^3\ 2^{2n-1}\ \big[(n-2)!\big]^4}{(2n-1)\big[(2n-2)!\big]^3} f^{(2n-2)}(\xi), \quad \text{for some } \xi \in (-1,1) \;\; [9]. \tag{48}$$

A Proof of this theorem can be found in Abramowitz & Stegun's book. Because the Gauss-Lobatto rule is only established for integration of a function over the interval $[-1,1]$, when computing the ψ_k's we must make the following change of interval

$$\int_a^b f(\bar{x})d\bar{x} = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right)dx \;\; [10]. \tag{49}$$

This implies that the coefficients ψ_k will now take on the form

$$\frac{2}{\pi}\int_0^{\pi} f(\cos\bar{\theta})\sin\big((k+1)\bar{\theta}\big)\sin\bar{\theta}\,d\bar{\theta} = \int_{-1}^{1} f\left(\cos\frac{\pi}{2}(\theta+1)\right)\sin\left(\frac{\pi}{2}(k+1)(\theta+1)\right)\sin\left(\frac{\pi}{2}(\theta+1)\right)d\theta. \tag{50}$$

An Error Bound for the Modified Clenshaw-Curtis Rule:

Let us now derive an upper bound for the error that results from using the modified Clenshaw-Curtis method to approximate the integral of f. Upon interpolating our given function at the zeros of $U_{n+1}(x)$ we can express f in a similar way as done in (38),

$$f(x) = \sum_{k=0}^{n}{}'' \psi_k U_k(x) + \frac{U_{n+1}(x)}{2^{n+1}(n+1)!} f^{n+1}(\xi) \;\; [7] \tag{51}$$

where again $|\xi| < 1$. First let's assume that the condition $f \in C^{n+1}[-1,1]$ holds, then by integrating both sides of the expression we find

$$\int_{-1}^{1} f(x)dx = \sum_{k=0}^{n}{}'' \psi_k \int_{-1}^{1} U_k(x)dx + \frac{1}{2^{n+1}(n+1)!}\int_{-1}^{1} U_{n+1}(x) f^{n+1}(\xi)dx;$$

applying (43) this becomes

$$\int_{-1}^{1} f(x)dx - \left(\psi_0 + \sum_{k=1}^{n/2-1}\frac{2\psi_{2k}}{2k+1} + \frac{\psi_n}{n+1}\right) = \frac{1}{2^{n+1}(n+1)!}\int_{-1}^{1} U_{n+1}(x) f^{n+1}(\xi)dx.$$

Taking absolute value to both sides we obtain

$$\left|\int_{-1}^{1} f(x)dx - \left(\psi_0 + \sum_{k=1}^{n/2-1} \frac{2\psi_{2k}}{2k+1} + \frac{\psi_n}{n+1}\right)\right| = \left|\frac{1}{2^{n+1}(n+1)!}\int_{-1}^{1} U_{n+1}(x) f^{n+1}(\xi)dx\right|$$

$$\leq \frac{1}{2^{n+1}(n+1)!}\int_{-1}^{1} \left|U_{n+1}(x) f^{n+1}(\xi)\right|dx$$

*Use of Cauchy-Schwarz inequality

$$\leq \frac{1}{2^{n+1}(n+1)!}\left(\int_{-1}^{1} \left|U_{n+1}(x)\right|^2 dx\right)^{1/2}\left(\int_{-1}^{1} \left|f^{n+1}(\xi)\right|^2 dx\right)^{1/2}$$

*Use of (39) and $|U_{n+1}(x)| \leq n+1$

$$\leq \frac{1}{2^{n+1}(n+1)!}\left(\int_{-1}^{1} (n+1)^2 dx\right)^{1/2}\left(\int_{-1}^{1} M^2 dx\right)^{1/2}$$

$$= \frac{\sqrt{2}(n+1)\sqrt{2}M}{2^{n+1}(n+1)!} = \frac{M}{2^n n!}.$$

Therefore an error bound for the modified Clenshaw-Curtis rule is

$$\boxed{\left|I(f) - I_{mcc}(f,n)\right| \leq \frac{M}{2^n n!}}. \tag{52}$$

Now consider another simple integral to verify the validity of (52). Let $f(x) = \cos(2x)$, this implies

$$M = \max_{-1\leq x\leq 1}\left|\frac{d^{n+1}}{dx^{n+1}}\big(\cos(2x)\big)\right| = 2^{n+1}$$

and

$$I(f) = \int_{-1}^{1} \cos(2x)dx = \frac{\sin(2) - \sin(-2)}{2} = \sin(2) \doteq 0.909297426825682.$$

Table 2: Actual Error vs. Error Bound for the Modified Clenshaw-Curtis Rule

n	Clenshaw-Curtis Approximation	Absolute Error	Error Bound
4	0.909642376076686	$3.449492510044783 \times 10^{-4}$	0.083333333333333
6	0.909292487926950	$4.938898732098629 \times 10^{-6}$	0.002777777777778
8	0.909297472613820	$4.578813816991101 \times 10^{-8}$	$4.960317460317460 \times 10^{-5}$
10	0.909297426528125	$2.975569790564236 \times 10^{-10}$	$5.511463844797178 \times 10^{-7}$
12	0.909297426827113	$1.431410545649214 \times 10^{-12}$	$4.175351397573620 \times 10^{-9}$
14	0.909297426825676	$5.218048215738236 \times 10^{-15}$	$2.294149119545945 \times 10^{-11}$

It is evident from this example that the given inequality in (52) will suffice to bound the error for sufficiently differentiable functions.

Findings and Results

The next eight figures show various examples which compare the convergence behavior and computational efficiency of Gauss-Legendre, Clenshaw-Curtis and the modified Clenshaw-Curtis routine (MC-CGL) as it will be denoted. The "GL" in this acronym stands for Gauss-Lobatto which is being applied

to compute the coefficients defined in (44). The error convergence graphs are measuring the ratio of the log of the absolute error to the number of nodes sampled. The log of the error is chosen because as n grows large the errors can be on the order of 10^{-15} which is difficult to see on a graph in comparison to error of order 10^{-2}, a common value for small node samplings. The computational efficiency graphs are each utilizing MATLAB's built in "tic toc" command to measure the amount of processing time that each algorithm requires for specific values of n. A third graph is shown in each example displaying the given function f and the shaded region of area in which the integral approximation is attempting to compute. Lastly, the numerical value of the integral computed to 16 digits of precision is given for each function.

One first observation made, is when examining these examples, Clenshaw-Curtis in all cases remains computationally superior to Gauss-Legendre and MCCGL. This is merely a confirmation of the behavior that we predicted from the beginning and is a consequence of the (FFT) versus a tridiagonal eigenvalue problem. For decaying and non-decaying oscillatory functions as shown in figures 4 and 8 Gauss-Legendre converges faster than Clenshaw-Curtis and MCCGL. Looking directly at the later two, Clenshaw-Curtis slightly outperforms MCCGL. In contrast, for nonnegative decaying and non-decaying oscillatory functions as depicted in figures 5 and 9, the new quadrature algorithm MCCGL converges faster than both Gauss-Legendre and Clenshaw-Curtis. For products of several functions such as the example shown in figure 6, the convergence of all three methods is almost identical. In cases where special functions are used such as the error function $\text{erf}(x)$ found in figure 7, MCCGL mirrors the behavior of Clenshaw-Curtis and Gauss slightly outperforms each. There are many other examples that can be tested and in general the behavior of convergence will vary from situation to situation. Figure 10 shows one of these cases where the new quadrature rule does significantly worse than its opponents. To examine the performance of MCCGL alone, table 3 has been included, which provides numerical approximations within an error tolerance of 10^{-15} for 15 sample integrals.

Error Convergence and Computational Efficiency Comparisons:

Fig. 4: $f(x) = \cos\left(\sqrt{377}x\right) + \sin\left(\sqrt{135}x\right)$, $I(f) \doteq 0.055318603004214$

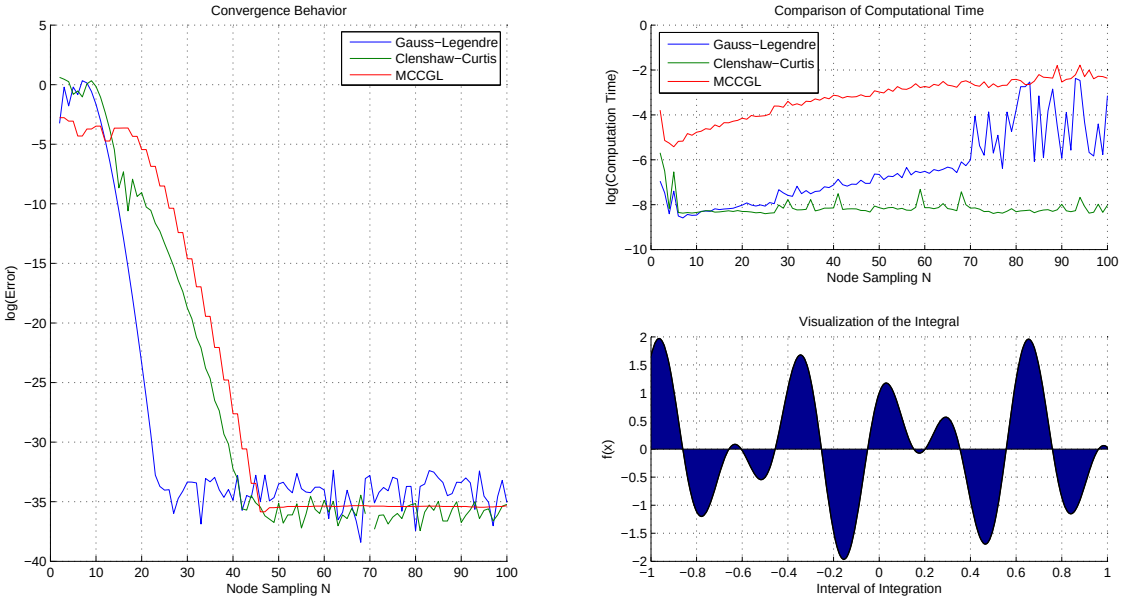


Fig. 5: $f(x) = \left|\cos\left(\sqrt{377}x\right) + \sin\left(\sqrt{135}x\right)\right|$, $I(f) \doteq 1.646690417700640$

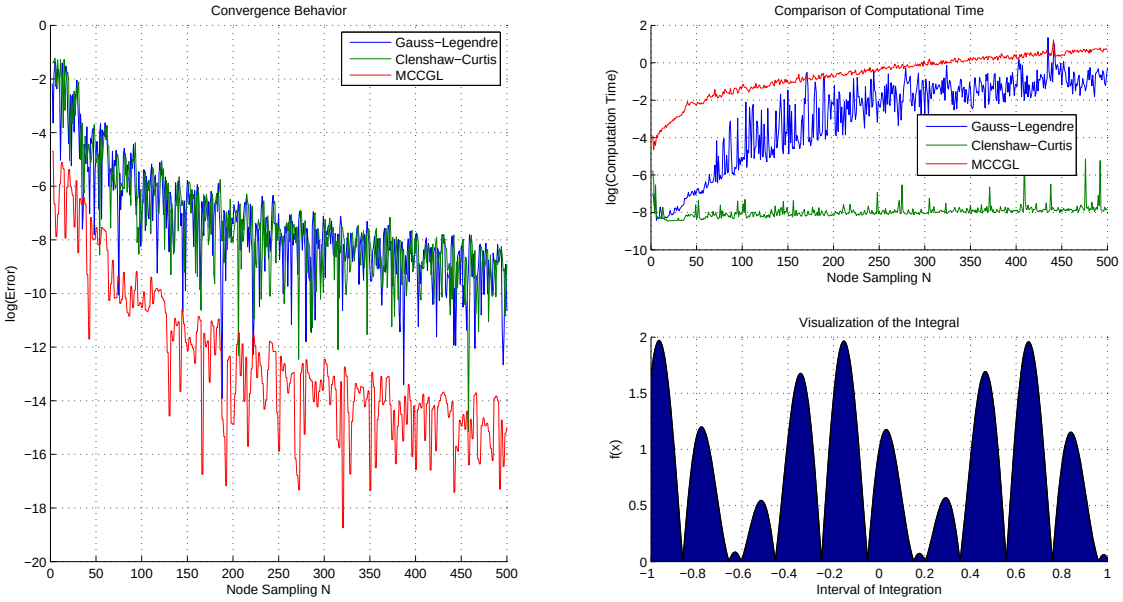


Fig. 6: $f(x) = x^2 e^{-x^2} \tan(x) \arccos(x)$, $I(f) \doteq -0.321556002594905$

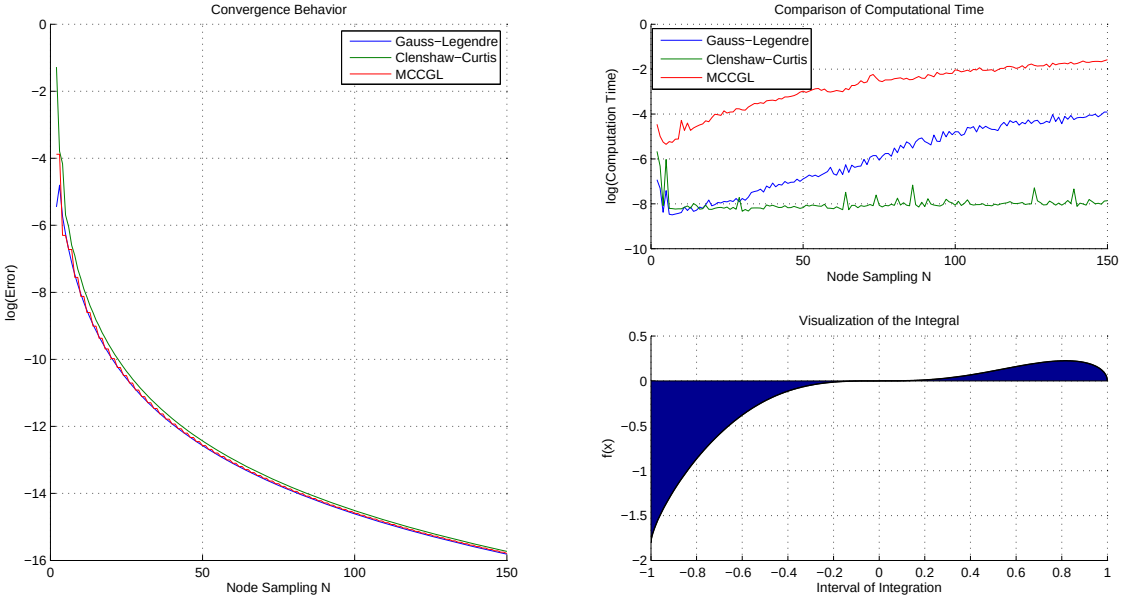
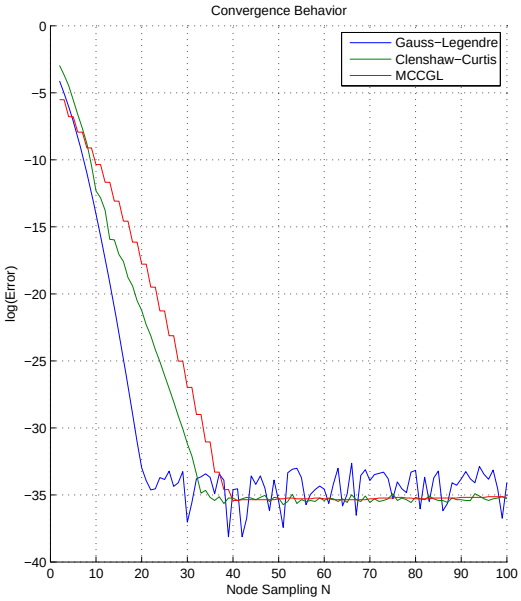
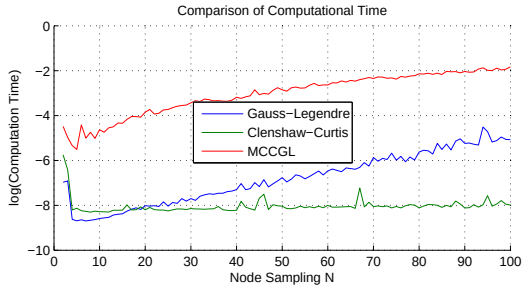


Fig. 7: $f(x) = \ln(x + 2e)\mathrm{erf}(\pi x)$, $I(f) \doteq 0.175664900305971$





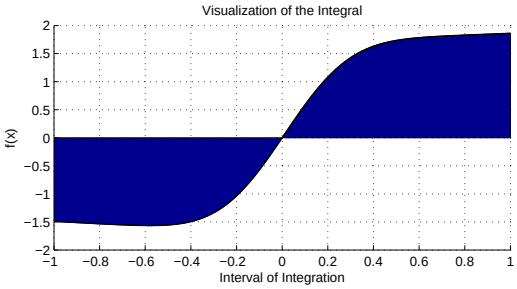
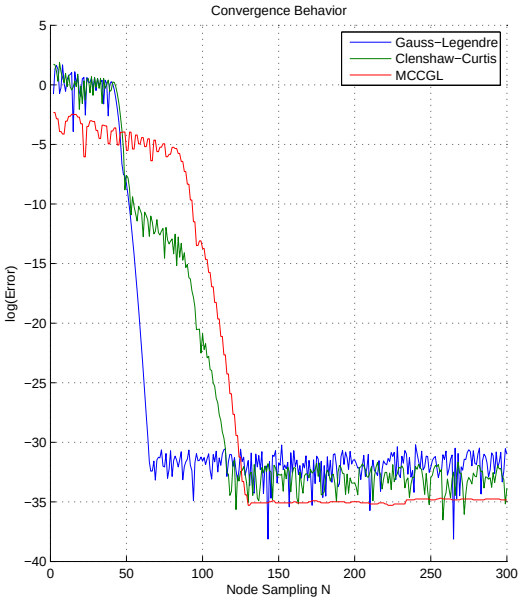
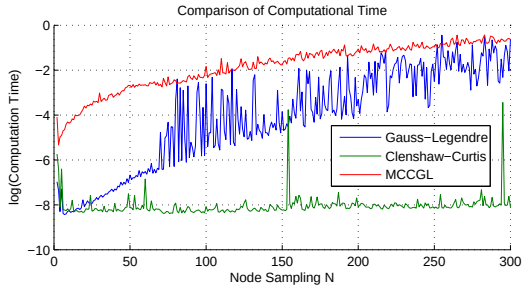


Fig. 8: $f(x) = e^{-3x}\cos\left(16\sqrt{3}\pi x\right)$, $I(f) \doteq -0.176358246030565$





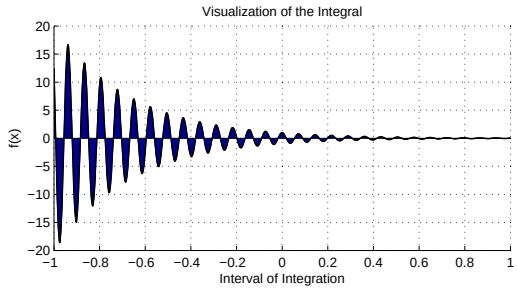
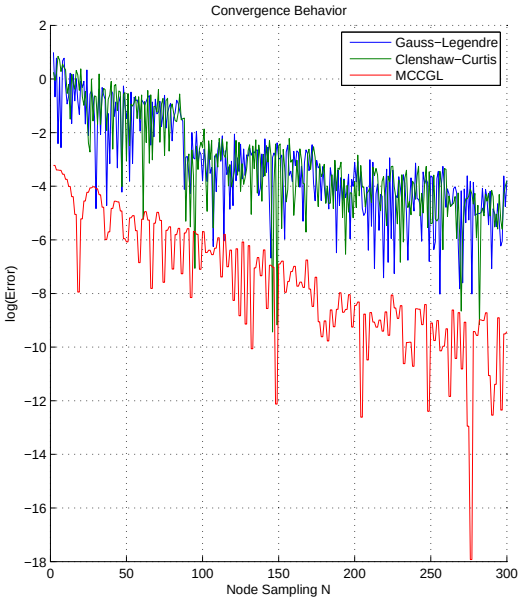
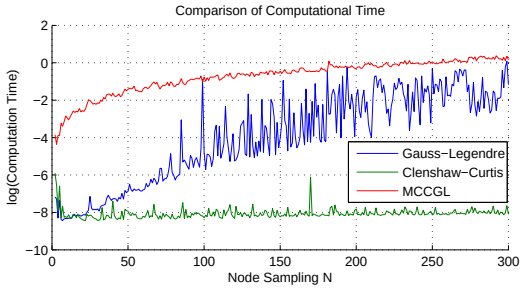


Fig. 9: $f(x) = \left| e^{-3x} \cos\left(16\sqrt{3}\pi x\right) \right|$, $I(f) \doteq 4.202933872637208$





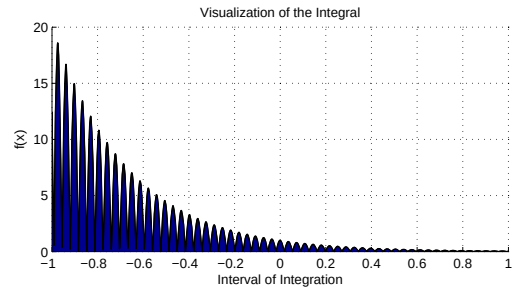
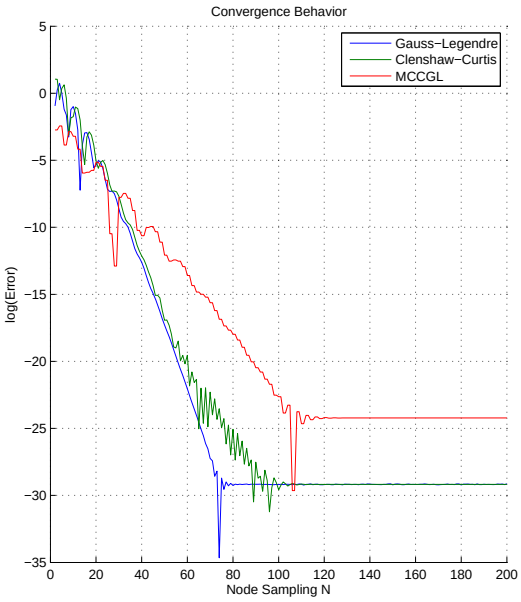
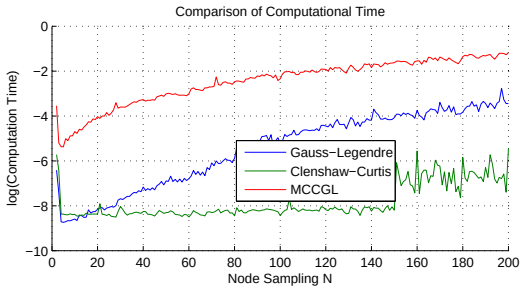


Fig. 10: $f(x) = e^{\cos\left(\sqrt{47}\pi x\right)}$, $I(f) \doteq 2.438081482203559$





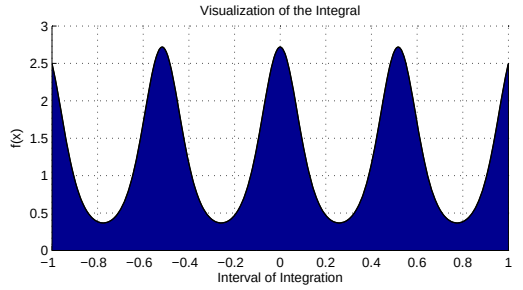


Fig. 11: $f(x) = \arctan\left(x^2\right)$, $I(f) \doteq 0.595805337996175$

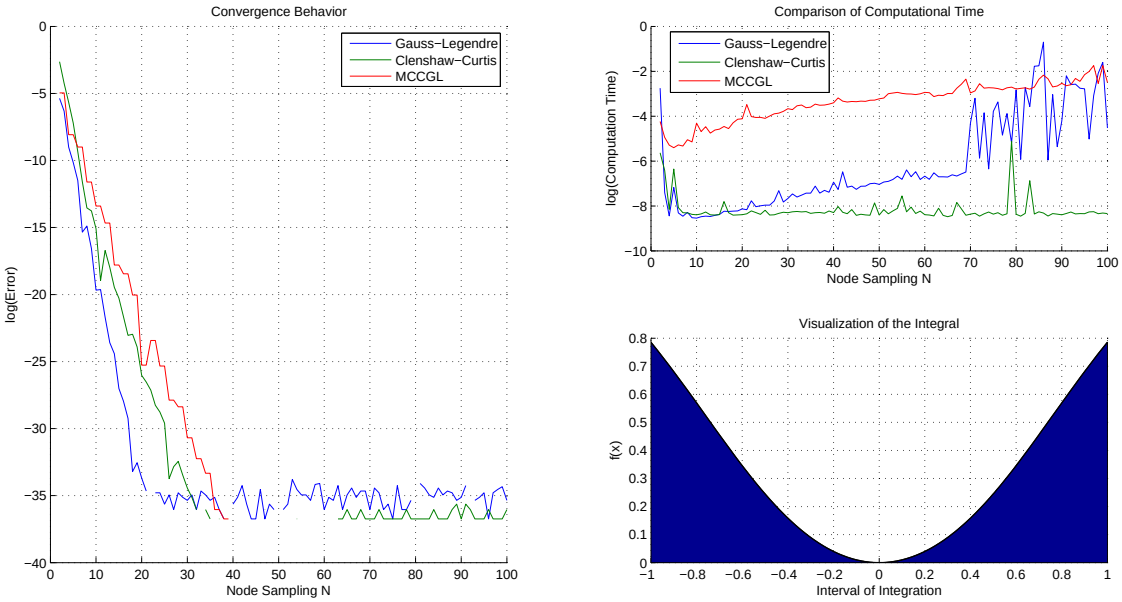


Table 3: 15 sample integrals computed by MCCGL within an error tolerance of 10^{-15}

Integrand $f(x)$	MCCGL Approximation	Nodes Required	Computation Time (Seconds)
e^x	2.350402387287603	12	0.009287600000000
$\sqrt{(100\pi)^2 - x^2}$	628.3174696833920	4	0.000079444000000
$x/\left(e^x + 1\right)$	−0.158885300995123	16	0.005830175521202
$1/\left(1 + x^2\right)$	1.570796326794896	34	0.013707653551110
$\frac{23}{25}\cosh(x) - \cos(x)$	0.479428226688802	12	0.004472379038743
$\cos\left(\sqrt{521}x\right) + \sin\left(\sqrt{273}x\right)$	−0.064910975381289	50	0.025488112094860
$\ln\left(x + 2e^2\right)\operatorname{erf}(2\pi x)$	0.066862331558334	64	0.038738946516917
$e^{-2x}\cos\left(16\sqrt{2}x\right)$	−0.218673123892560	50	0.026050324700878
$x\arctan\left(x^3\right)$	0.355120831053971	42	0.045733800000000
$e^x\arctan\left(x^3\right)$	0.398130064822845	42	0.035174700000000
$x\sin(30x)\Big/\sqrt{1 - x^2/\left(4\pi^2\right)}$	−0.012696821645673	60	0.048116700000000
$x\sin(30x)\cos(50x)\Big/\sqrt{1 - x^2/\left(4\pi^2\right)}$	0.019528272812124	120	0.123265800000000
$x\sin(50x)\cos(75x)$	0.033518732588154	170	0.286400100000000
$1/\left(x^4 + x^2 + e\right)$	0.631299652055891	28	0.016825800000000
$\tan(x)/\left(1 + e^x\sin(\pi x)\right)$	−0.719818067507943	90	0.121847500000000

Direct Comparison of CCGL and MCCGL:

One further consideration that was made, was to apply the Gauss-Lobatto rule to the original Clenshaw-Curtis algorithm in order to compute the coefficients given in (16). Doing so we see that this new modification, denoted by CCGL, along with MCCGL, both display nearly identical convergence patterns and require the same order of floating point operations. This behavior is interesting and can be attributed to the similar nature of both kinds of Chebyshev polynomials. Although for certain functions the expansions in (14) and (24) will vary slightly in how fast they converge on $[-1, 1]$, both approaches should display the same rate of asymptotic convergence. It is for this reason that the observed error as a result of integrating f bears close resemblance for both methods.

Fig. 12: $f(x) = \cos(10x)\Gamma(x+2)\mathrm{erf}(\sqrt{1+x})$, $I(f) \doteq -0.115420768826884$

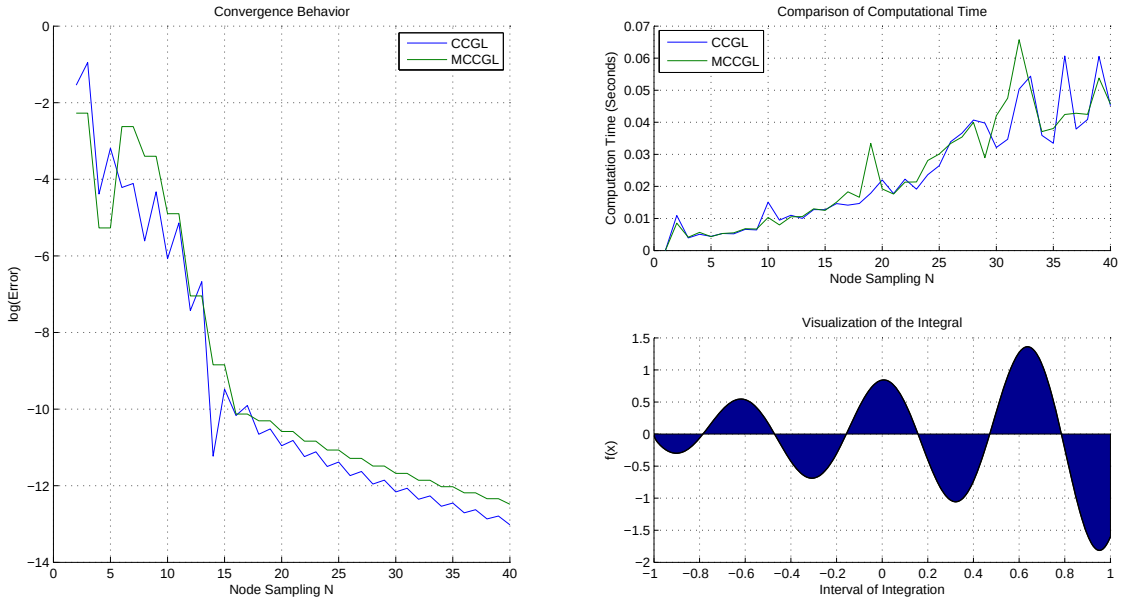


Table 4: A comparison of the errors for $f(x) = \cos(10x)\Gamma(x+2)\mathrm{erf}(\sqrt{1+x})$

n	CCGL Approx.	CCGL Error	MCCGL Approx.	MCCGL Error
4	−0.127880902110724	0.012460133283840	−0.110269423215667	0.005151345611217
8	−0.111758657474522	0.003662111352363	−0.082034833703284	0.033385935123600
12	−0.114825707423031	0.000595061403853	−0.114549294656312	0.000871474170572
16	−0.115382301684895	0.000038467141989	−0.115460693253348	0.000039924426463
20	−0.115403326736661	0.000017442090223	−0.115446088057898	0.000025319231014
24	−0.115410611797485	0.000010157029400	−0.115436370720138	0.000015601893254
28	−0.115414349392029	0.000006419434855	−0.115431025001805	0.000010256174921
32	−0.115416459037443	0.000004309789441	−0.115427858133590	0.000007089306705

Fig. 13: $f(x) = \cosh(\tanh(\sinh(x)))$, $I(f) \doteq 2.278006221315598$

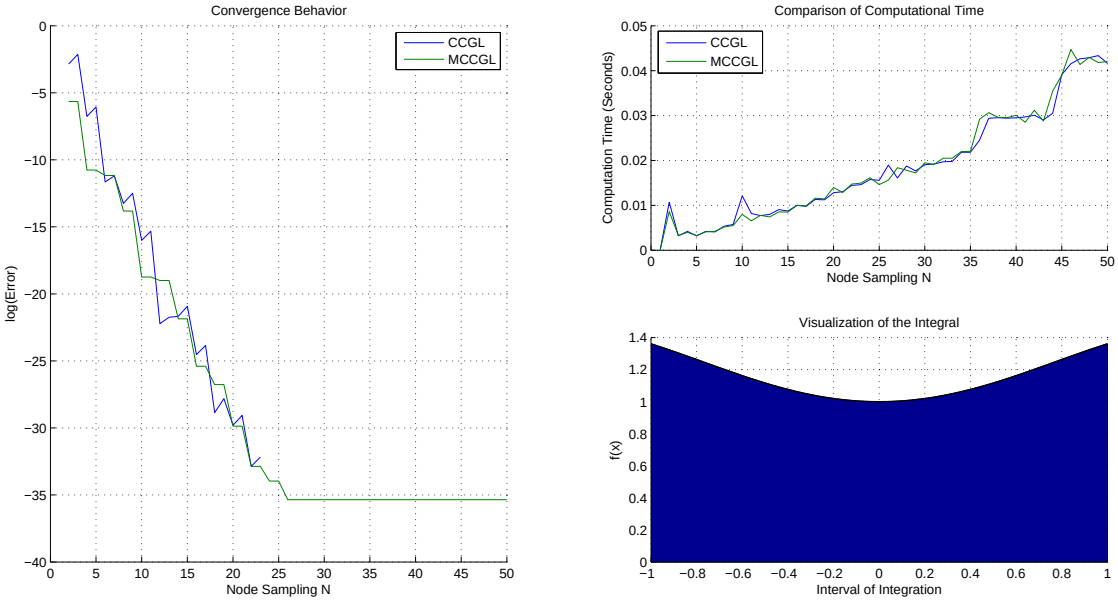


Table 5: A comparison of the errors for $f(x) = \cosh(\tanh(\sinh(x)))$

n	CCGL Approx.	CCGL Error	MCCGL Approx.	MCCGL Error
4	2.276846498382461	0.001159722933137	2.277985124995150	0.000021096320449
6	2.278014932115627	0.000008710800029	2.277992122558898	0.000014098756700
8	2.278007981850735	0.000001760535136	2.278007217776638	0.000000996461039
10	2.278006110197199	0.000000111118399	2.278006228565790	0.000000007250192
12	2.278006221540577	0.000000000224979	2.278006215719572	0.000000005596026
14	2.278006221698916	0.000000000383317	2.278006221636900	0.000000000321302
16	2.278006221293295	0.000000000022303	2.278006221324897	0.000000000009299

Future Work

Beyond the scope of what has been presented in this project, there are many other possible avenues of research that can be pursued related to Clenshaw-Curtis quadrature. It is natural for one to hold interest in investigating the use of other sets of orthogonal functions for the basis of an expansion. Such research was attempted for this project using the Legendre and Hermite polynomials, but to no avail a suitable algorithm for integration was not found. In both cases, Legendre integration over the interval $[-1, 1]$ and Hermite integration over the interval $(-\infty, \infty)$ lead to problems when attempting to integrate the series term-by-term. Similar to (14), if we expand the integrand $f(x)$ in a Legendre Series

$$f(x) = \sum_{k=0}^{\infty} \ell_k P_k(x), \quad -1 \leq x \leq 1 \tag{53}$$

it is easily verified that integrating this sum term-by-term will yield a trivial result due to the fact that

$$\int_{-1}^{1} P_k(x)dx = 0, \quad \text{for all } k \geq 1$$

and

$$\ell_k = \frac{2k+1}{2} \int_{-1}^{1} f(x)P_k(x)dx \;\Rightarrow\; \ell_0 = \frac{1}{2} \int_{-1}^{1} f(x)dx.$$

All we are left with is

$$\int_{-1}^{1} \ell_0 \, dx = \int_{-1}^{1} f(x)dx$$

which is an uninteresting finding. To the same end, if we expand the integrand as a series of Hermite polynomials the aforementioned trivial result will occur upon term-by-term integration. Some other possible considerations for this research area are the use of the Laguerre polynomials over the interval $[0, \infty)$ or the Jacobi polynomials, a generalized form of the Chebyshev polynomials. Nevertheless, this would require more extensive analysis to achieve an algorithm similar to that of Clenshaw-Curtis. In terms of computational efficiency it would also be interesting to mimic the work of Gentleman [5] and apply the fast Fourier transform to the modified method presented in (45) and to any future algorithms derived from the use of other sets of orthogonal functions.

Apart from the integration of a function of one variable a compelling idea would be to make use of what are called the multivariate Chebyshev polynomials to construct higher dimensional expansions of a function of several variables. Quite a bit of work has been pursued in this field relating to the development of recursion formulas for multivariate Chebyshev polynomials of the first kind. It is rather uncharted territory when it comes to generating the second kind of these polynomials, let alone using them as means for an expansion of some function. It would be useful to develop an efficient Clenshaw-Curtis algorithm which can compute double and triple integrals within some error tolerance, over various domains. Comparison of such an algorithm with Gauss-Quadrature for multiple integrals may yield valuable results in terms of convergence behavior and error bounds. To pursue such an idea we will require assorted elements from group theory such as the concept of Dynkin diagrams, which pertain to Weyl groups (finite reflection groups) and the overall theory of Lie groups. We will also need a keen understanding of root systems or root lattices within a Lie group. With this we will be able to develop a suitable algorithm for multiple integration over appropriate domains.

Some other topics relating to MCCGL that can be analyzed are

- stability of the algorithm,
- the possibility of preconditioning the eigenvalue problem which is solved each time Gauss-Lobatto attempts to find the coefficients ψ_k in the series shown in (24),
- incorporating an adaptivity technique which will allow for user specified error tolerance rather than the choice of n,
- and lastly, find a suitable way of handling endpoint and interior singularities of the kernel $f(x)$.

Appendix A: MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%           The (n+1)-point Gauss-Legendre Quadrature Rule                  %%
%%  Input: The function f(x) and the number of nodes to be sampled N         %%
%%        Output: The approximate integral and computation time              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [I time] = gauss(f,N)
tic
beta = .5./sqrt(1-(2*(1:N)).^(-2));      % 3-Term Recurrence Coefficients
T = diag(beta,1) + diag(beta,-1);        % Jacobi Matrix
[V,D] = eig(T);                          % Eigenvalue Decomposition
x = diag(D); [x,i] = sort(x);            % Legendre Abscissas
w = 2*V(1,i).^2;                         % Corresponding Weights
I = w*feval(f,x);                        % Integral Approximation
time = toc;                              % Computation time
```

[15]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%           The (n+1)-point Clenshaw-Curtis Quadrature Rule                 %%
%%  Input: The function f(x) and the number of nodes to be sampled N         %%
%%        Output: The approximate integral and computation time              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [I time] = clenshaw_curtis(f,N)
tic
fa = f(-1);
assignin('caller','fa',0);
x = cos(pi*(0:N)'/N);                       % Chebyshev Abscissas
fx = feval(f,x)/(2*N);                      % f evaluated at each point
g = real(fft(fx([1:N+1 N:-1:2])));          % Fast Fourier Transform
a = [g(1); g(2:N)+g(2*N:-1:N+2); g(N+1)];   % Chebyshev Coefficients
w = 0*a'; w(1:2:end) = 2./(1-(0:2:N).^2);   % Weight Vector
I = w*a;                                    % Integral Approximation
time = toc;                                 % Computation time
```

[15]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%         The (n+1)-point Clenshaw-Curtis/Gauss-Lobatto Rule                %%
%%  Input: The function f(x) and the number of nodes to be sampled N         %%
%%        Output: The approximate integral and computation time              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [approx time] = CCGL(f,N)
format long
tic
```

```
c = zeros(N,1);
g = @(x) cos((pi/2).*x+(pi/2));                 % Change of variable
h = @(x) f(g(x));
c0 = quadl(h,-1,1);                             % Leading Chebyshev coefficient
for k = 2:2:N                                   % Integrand of the c_k's
    h = @(x) f(g(x)).*cos(k.*((pi/2).*x+(pi/2)));
    c(k) = quadl(h,-1,1);                       % Computation of the coefficients
end                                             % using Gauss-Lobatto quadrature
intSum = 0;
for m = 1:N/2-1
    intSum = intSum + 2*c(2*m)/(1-(2*m)^2); % Weighted sum
end
approx = c0 + intSum + c(N)/(1-N^2);            % Integral approximation
time = toc;                                     % Computation time
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     The (n+1)-point Modified Clenshaw-Curtis/Gauss-Lobatto Rule     %%
%%  Input: The function f(x) and the number of nodes to be sampled N   %%
%%       Output: The approximate integral and computation time         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [approx time] = MCCGL(f,N)
tic
format long
psi = zeros(N,1);
g = @(x) cos((pi/2).*x+(pi/2));                 % Change of variable
h = @(x) f(g(x)).*(sin((pi/2).*x+(pi/2))).^2;
psi0 = quadl(h,-1,1);                           % Leading Chebyshev coefficient
                                                % computed using Gauss-Lobatto
for k = 2:2:N                                   % Integrand of the psi_k's
    h = @(x) f(g(x)).*sin((k+1).*((pi/2)...
        .*x+(pi/2))).*sin((pi/2).*x+(pi/2));
    psi(k) = quadl(h,-1,1);                     % Computation of the coefficients
end
intSum = 0;
for m = 1:N/2
    intSum = intSum + 2*psi(2*m)/(2*m+1);       % Weighted sum
end
approx = 2*psi0 + intSum;                       % Integral approximation
time = toc;                                     % Computation time
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  A function to compare Gauss Quadrature, Clenshaw-Curtis, & MCCGL   %%
%%       Input: The function f(x) and the number of iterations         %%
%%   Ouput: Error convergence and computation time tables and graphs   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = quadratureComparison(f,itmax)
clf
```

```
close all
format longG
I_exact = double(integral(f,-1,1));
for N = 2:itmax
    [intG(N) timeG(N)] = gauss(f,N);
    errorG(N) = abs(I_exact-intG(N));
    [intCC(N) timeCC(N)] = clenshaw_curtis(f,N);
    errorCC(N) = abs(I_exact-intCC(N));
    [intMCCGL(N) timeMCCGL(N)] = MCCGL(f,N);
    errorMCCGL(N) = abs(I_exact-intMCCGL(N));
end
disp('          Gauss                ErrorG          Clenshaw-Curtis          ErrorCC')
disp([intG' errorG' intCC' errorCC'])
disp('          MCCGL                ErrorMCCGL')
disp([intMCCGL' errorMCCGL'])
disp('The exact value of the integral is')
disp(I_exact)
vec = 1:itmax; X = -1:.001:1;
subplot(2,2,4)
hold all
area(X,f(X))
p = plot(X,f(X));
set(p,'Color','black','LineWidth',1)
grid on
xlabel('Interval of Integration')
ylabel('f(x)')
title('Visualization of the Integral')
subplot(2,2,[1 3])
hold all
plot(vec,log(errorG))
plot(vec,log(errorCC))
plot(vec,log(errorMCCGL))
grid on
xlabel('Node Sampling N')
ylabel('log(Error)')
title('Convergence Behavior')
legend('Gauss-Legendre','Clenshaw-Curtis','MCCGL')
subplot(2,2,2)
hold all
plot(vec,log(timeG))
plot(vec,log(timeCC))
plot(vec,log(timeMCCGL))
grid on
xlabel('Node Sampling N')
ylabel('log(Computation Time)')
title('Comparison of Computational Time')
legend('Gauss-Legendre','Clenshaw-Curtis','MCCGL')
```

References

[1] C.W. Clenshaw and A.R. Curtis. A Method for Numerical Integration on an Automatic Computer. *Numerische Mathematik*, 2:197–205, 1960.

[2] K.E. Atkinson. *An Introduction to Numerical Analysis, Second Edition*. John Wiley & Sons, Inc., Malden, Massachusetts, 1989.

[3] H. O'Hara and F.J. Smith. Error Estimation in the Clenshaw-Curtis Quadrature Formula. *The Computer Journal*, 11:213–219, 1968.

[4] W.M. Gentleman. Implementing Clenshaw-Curtis Quadrature i. *Communications of the ACM*, 15:337–342, 1972.

[5] W.M. Gentleman. Implementing Clenshaw-Curtis Quadrature ii. *Communications of the ACM*, 15:343–346, 1972.

[6] R.W. Hamming. *Numerical Methods for Scientists and Engineers, Second Edition*. Dover Publications, Inc., New York, New York, 1986.

[7] F.B. Hildebrand. *Introduction to Numerical Analysis, Second Edition*. McGraw-Hill, Inc., New York, New York, 1974.

[8] J.M. Horner. Generalized Rodrigues Formula Solutions For Certain Linear Differential Equations. *Transactions of the American Mathematical Society*, 115:31–42, 1965.

[9] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., New York, New York, 1965.

[10] R.L. Burden and J.D. Faires. *Numerical Analysis, Ninth Edition*. Brooks/Cole, Cengage Learning, Boston, Massachusetts, 2011.

[11] G.H. Golub and J.H. Welsch. Calculation of Gauss Quadrature Rules. *Communications of the ACM*, 23:221–230, 1969.

[12] J.W. Cooley and J.W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19:297–301, 1965.

[13] C. Lanczos. *Discourse on Fourier Series*. Hafner Publishing Company, Inc., New York, New York, 1966.

[14] P.J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, Inc., New York, New York, 1975.

[15] L.N. Trefethen. Is Gauss Quadrature better than Clenshaw-Curtis? *SIAM Review*, 50:67–87, 2008.

[16] D. Zwillinger. *Handbook of Integration*. Jones and Bartlett Publishers, Inc., Boston, Massachusetts, 1992.