On the Numerical Solution of Fourth-Order Linear Two-Point Boundary Value Problems

William Leeb and Vladimir Rokhlin

Abstract

This paper introduces a fast and numerically stable algorithm for the solution of fourth-order linear boundary value problems on an interval. This type of equation arises in a variety of settings in physics and signal processing. However, current methods of solution involve discretizing the differential equation directly by finite elements or finite differences, and consequently suffer from the poor conditioning introduced by such schemes. Our new method instead reformulates the equation as a collection of second-kind integral equations defined on local subdomains. Each such equation can be stably discretized. The boundary values of these local solutions are matched by solving a banded linear system. The method of deferred corrections is then used to increase the accuracy of the scheme. Deferred corrections requires applying the integral operator to a function on the entire domain, for which we provide an algorithm with linear cost. We illustrate the performance of our method on several numerical examples.

1 Introduction

This paper describes the numerical solution to differential equations of the form

$$\sum_{j=0}^{4} a_j(x) \frac{d^j \phi}{dx^j}(x) = f(x) \tag{1}$$

for x in an interval $[a, b]$, with specified boundary conditions

$$\phi(a) = \alpha_{l,0} \tag{2}$$
$$\phi(b) = \alpha_{r,0} \tag{3}$$
$$\phi'(a) = \alpha_{l,1} \tag{4}$$
$$\phi'(b) = \alpha_{r,1} \tag{5}$$

and given coefficients $a_j(x)$ and right hand side $f(x)$.

Fourth-order equations of this kind arise in a variety of physical problems. In the small-bending regime, the shape of a beam under an external force is described as the solution to an equation of the form

$$\frac{d^2}{dx^2}\left(c(x)\frac{d^2\phi}{dx^2}(x)\right) = f(x) \tag{6}$$

where $c(x)$ is the stiffness of the beam, and $f(x)$ is an external force applied to the beam [8]. One-dimensional equations also arise from separation-of-variables for higher-dimensional problems, such as vibration of plates [5]. A notable fourth-order operator on the unbounded domain $[0, \infty)$ is defined by:

$$-\frac{d^2}{dx^2}\left(x^2\frac{d^2\phi}{dx^2}(x)\right) + (a^2+b^2)\frac{d}{dx}\left(x^2\frac{d\phi}{dx}(x)\right) - (a^2b^2x^2 - 2a^2)\phi(x). \tag{7}$$

This operator arises in many applications because it commutes with the truncated Laplace transform composed with its adjoint [2, 3]. It is useful when working with the family of decaying exponential functions, and has been the subject of recent investigation [17, 18, 19].

Existing methods for solving fourth-order boundary value problems of the form (1)–(5) employ finite difference or finite element schemes. To obtain the solution on m nodes, these methods involve solving an $O(m)$-by-$O(m)$ banded system of linear equations. While the solution can be obtained with asymptotic cost $O(m)$, the discretizations used introduce a condition number of size $O(m^4)$ [23]. The resulting loss of accuracy in the solution is entirely due to the choice of discretization, and is not a result of the conditioning inherent to the problem (1)–(5).

This paper proposes an algorithm for the solution of (1)–(5) that maintains the $O(m)$ running time of finite element and finite difference schemes, but is as accurate as the problem allows. Our method centers on expressing the solution ϕ in the form

$$\phi(x) = \int_a^b G_0(x,t)\sigma(t)dt + \psi_\alpha(x), \tag{8}$$

where $G_0(x,t)$ is the Green's function for the biharmonic equation $\phi^{(4)} = f$ with zero boundary values; ψ_α is a third degree polynomial with the desired boundary values; and $\sigma = \phi^{(4)}$ is the new function to be solved for.

The key observation is that the function σ can be expressed as the solution to a second-kind integral equation. While direct discretizations of the differential equation (1)–(5) are ill-conditioned, second-kind integral equations can be stably discretized. More precisely, the values of σ on a grid of points is expressible as the solution to a linear system whose condition number does not markedly exceed the condition number of the original continuous problem.

The challenge with using second-kind integral equations is that while their discretized linear systems are well-conditioned, they are dense; consequently, a naive solver will have cubic cost $O(m^3)$. It has been observed that for many physical problems, these dense linear systems can nevertheless be solved in linear or nearly linear time. This is the observation underpinning a variety of methods devised for the solution of second-order two-point boundary value problems [22, 13, 20].

This paper employs a more direct approach to solving for σ. Briefly, our method contains three steps. First, we use the integral equation form of the problem to produce m local solutions of the equation (1) with homogeneous boundary values. Next, we solve a banded linear system to match the boundary values of the local solutions. While

this step introduces extraneous loss of accuracy similar to a finite element scheme, it is corrected by the method of deferred corrections, in which we recursively solve for the residual solutions on the entire interval $[a, b]$. Computing the right hand side of the residual equation requires applying the integral operators defined by the biharmonic Green's function and its derivatives to an arbitrary function on the entire interval $[a, b]$, for which we provide a linear time algorithm. This linear time algorithm is modeled after the fast multipole method [12] for applying certain dense matrices to vectors.

The asymptotic CPU time of our algorithm is $O(mn^3 \log(1/\epsilon))$, where ϵ is machine precision, n is a user-selected integer (the number of points for each local solution), and m is the number of local solutions, or discretization nodes. The complexity of the algorithm is linear in the number of discretization nodes, but nevertheless achieves full machine accuracy. In this sense, our algorithm realizes the advantages of finite element methods' small CPU time while maintaining the numerical accuracy afforded by second-kind integral equations.

The rest of the paper is structured as follows. In Section 2, we review the mathematical and numerical tools we will be using in our algorithm. In Section 3, we describe in detail the algorithm for solving (1)–(5). In Section 4, we provide the results of numerical experiments.

2 Mathematical and numerical preliminaries

In this section, we review the mathematical and numerical tools that we will be using throughout the paper. In particular, we will show how to express the function ϕ which solves (1)–(5) in terms of the solution to a second-kind integral equation; review the properties of Gaussian quadrature, which we will use to discretize this integral equation; and review the method of deferred corrections.

2.1 Rescaling the problem domain

It will be convenient to rescale the problem (1)–(5) to convert the interval $[a, b]$ into $[-1, 1]$. Define:

$$\tilde{\phi}(y) = \phi\left(\frac{(b-a)(y+1)}{2} + a\right) \tag{9}$$

$$\tilde{a}_j(y) = \left(\frac{2}{b-a}\right)^j a_j\left(\frac{(b-a)(y+1)}{2} + a\right) \tag{10}$$

$$\tilde{f}(y) = f\left(\frac{(b-a)(y+1)}{2} + a\right). \tag{11}$$

The equations (1)–(5) for ϕ are then equivalent to the equation

$$\sum_{i=0}^{4} \tilde{a}_j(y) \frac{d^j \tilde{\phi}}{dy^j}(y) = \tilde{f}(y) \tag{12}$$

with modified boundary values

$$\tilde{\phi}(-1) = \alpha_{l,0} \tag{13}$$

$$\tilde{\phi}(1) = \alpha_{r,0} \tag{14}$$

$$\tilde{\phi}'(-1) = \left(\frac{b-a}{2}\right)\alpha_{l,1} \tag{15}$$

$$\tilde{\phi}'(1) = \left(\frac{b-a}{2}\right)\alpha_{r,1}. \tag{16}$$

After solving for $\tilde{\phi}$ and its derivatives, we can perform the inverse change of variables to arrive at the solution ϕ and its derivatives, as follows:

$$\phi^{(j)}(x) = \left(\frac{2}{b-a}\right)^j \tilde{\phi}^{(j)}\left(\frac{2(x-a)}{b-a} - 1\right). \tag{17}$$

2.2 The biharmonic Green's function on $[-1, 1]$

We will make central use of the Green's function $G_0(x, t)$ for the biharmonic equation $\phi^{(4)} = f$ with homogeneous boundary conditions ($\alpha_{l,0} = \alpha_{r,0} = \alpha_{l,1} = \alpha_{r,1} = 0$ in (2)–(5)), on the interval $[-1, 1]$. The biharmonic Green's function is given by the formula:

$$G_0(x,t) = \begin{cases} (1-t)^2(1+x)^2(1+2t-2x-tx)/24, & \text{if } t > x \\ (1-x)^2(1+t)^2(1+2x-2t-tx)/24, & \text{if } t < x \end{cases} \tag{18}$$

It can be checked by direct calculation that $G_0(x, t)$ satisfies the defining properties of the Green's function (see [5]). Specifically, the following properties hold:

$$\frac{\partial^4 G}{\partial x^4}(x,t) = 0, \quad t \in [-1,1], \tag{19}$$

$$G_0(-1,t) = G_0(1,t) = \frac{\partial G_0}{\partial x}(-1,t) = \frac{\partial G_0}{\partial x}(1,t) = 0, \quad t \in [-1,1], \tag{20}$$

$$\lim_{t\to x^+} \frac{\partial^j G}{\partial x^j}(x,t) = \lim_{t\to x^-} \frac{\partial^j G}{\partial x^j}(x,t), \quad x \in [-1,1], \quad 0 \le j \le 2, \tag{21}$$

and

$$\lim_{h\to 0}\left[\frac{\partial^3 G}{\partial x^3}(x+h,t) - \frac{\partial^3 G}{\partial x^3}(x-h,t)\right] = -1, \quad t \in [-1,1]. \tag{22}$$

We will use the notation G_j to denote the j^{th} partial derivative of the Green's function, $0 \le j \le 3$; that is, we define:

$$G_j(x,t) = \frac{\partial^j G_0}{\partial x^j}(x,t). \tag{23}$$

We will use the boldface letter $\mathbf{G}_j$ to denote the corresponding integral operator. In this notation, for a function f on $[-1,1]$ we will write:

$$(\mathbf{G}_j f)(x) = \int_{-1}^{1} G_j(x,t) f(t) dt = \int_{-1}^{1} \frac{\partial^j G_0}{\partial t^j}(x,t) f(t) dt. \tag{24}$$

The significance of the Green's function and its derivatives is that the solution to the equation $\phi^{(4)} = f$ with zero boundary conditions is the function $\phi = \mathbf{G}_0 f$, with derivatives $\phi^{(j)} = \mathbf{G}_j f$, $1 \leq j \leq 3$. In Section 2.3, we will use the biharmonic Green's function to reformulate the problem (1)–(5) as a second-kind integral equation.

2.3 Integral form of the boundary value problem (1)–(5)

In this section, we will use the biharmonic Green's function to reformulate the differential equation (1)–(5) as a second-kind integral equation. For comprehensive background on second-kind integrals equations, see [21, 5, 4]. We let $\mathbf{L}$ denote the differential operator on the left side of (1); that is,

$$(\mathbf{L}\phi)(x) = \sum_{j=0}^{4} a_j(x) \frac{d^j \phi}{dx^j}(x) \tag{25}$$

The equation (1) can then be written in the more compact form $\mathbf{L}\phi = f$.

Any four-times differentiable function with vanishing values and first derivatives on $[-1,1]$ can be written in the form $\mathbf{G}_0\sigma$, for some function σ. Consequently, we can express the solution ϕ to (1)–(5) as being of the form $\phi = \mathbf{G}_0\sigma + \psi_\alpha$, where $\sigma = \phi^{(4)}$, and where ψ_α is a function satisfying the boundary conditions (2)–(5) and $\psi_\alpha^{(4)} = 0$ on $[-1,1]$.

It can be directly checked that the unique function ψ_α on $[-1,1]$ satisfying $\psi^{(4)} = 0$ and the boundary conditions (2)–(5) (with $a = -1$ and $b = 1$) is given by the following formula:

$$\psi_\alpha(x) = \alpha_{l,0}\psi_{l,0}(x) + \alpha_{r,0}\psi_r(x) + \alpha_{l,1}\psi_{l,1}(x) + \alpha_{r,1}\psi_{r,1}(x) \tag{26}$$

where the functions $\psi_{l,0}$, $\psi_{r,0}$, $\psi_{l,1}$ and $\psi_{r,1}$ are defined by the formulas:

$$\psi_{l,0}(x) = (1-x)^2(2+x)/4 \tag{27}$$
$$\psi_{r,0}(x) = (1+x)^2(2-x)/4 \tag{28}$$
$$\psi_{l,1}(x) = (1-x)^2(x+1)/4 \tag{29}$$
$$\psi_{r,1}(x) = (1+x)^2(x-1)/4. \tag{30}$$

The function $\sigma = \phi^{(4)}$ is expressible as the solution to a second-kind integral equation, as we now show. The equation $\mathbf{L}(\mathbf{G}_0\sigma + \psi_\alpha) = f$ can be written equivalently as

$$\mathbf{L}\mathbf{G}_0\sigma = f - \mathbf{L}\psi_\alpha. \tag{31}$$

Because $\frac{\partial^4 G}{\partial x^4}(x,t) = \delta(x-t)$, the operator $\mathbf{L}\mathbf{G}_0$ is of the form

$$(\mathbf{L}\mathbf{G}_0\sigma)(x) = a_4(x)\sigma(x) + \sum_{j=0}^{3} a_j(x) \int_{-1}^{1} G_j(x,t)\sigma(t)dt. \tag{32}$$

After solving (31) for σ, the solution ϕ to (1)–(5) and its derivatives are given by $\phi^{(j)} = \mathbf{G}_j\sigma + \psi_\alpha^{(j)}$, for $0 \le j \le 3$, and $\phi^{(4)} = \sigma$. The reformulation of the differential equation as a second-kind integral equation is beneficial, as the latter can be stably discretized.

Remark 1. The integral equation formulation of (1)–(5) makes use of the biharmonic Green's function G_0. However, in principle this can be replaced by the Green's function G_0 for any equation $\mathbf{L}_0\phi = f$ with a fourth-order differential operator $\mathbf{L}_0$. In this general setting, G_0 is known as the *background* Green's function [13, 20]. We choose to work with the biharmonic equation because it is so analytically tractable. We also suspect that the numerical performance of our method will not depend substantially on the choice of background Green's function, as was observed for second-order equations in the numerical experiments of [13].

2.4 Legendre polynomials and Gaussian quadrature

The n^{th} Legendre polynomial P_n is defined for $x \in [-1,1]$ by

$$P_0(x) = 1; \quad P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2-1)^n, \quad n \ge 1. \tag{33}$$

As is well known [1], the n roots $y_1, \ldots, y_n$ of P_n lie in $[-1,1]$. Together with a suitable choice of weights $w_i > 0$, they can be used to evaluate the integral on $[-1,1]$ of any polynomial f of degree less than or equal to $2n+1$ via the formula

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} f(y_i)w_i. \tag{34}$$

By rescaling the interval $[-1,1]$ to $[a,b]$, we obtain the quadrature

$$\int_a^b f(x)dx = \sum_{i=1}^{n} f\left(\frac{b-a}{2}(y_i+1)+a\right)\frac{b-a}{2}w_i. \tag{35}$$

The rescaled nodes $\frac{b-a}{2}(y_i+1)+a$ are the roots of the polynomials $P_n(2(x-a)/(b-a)-1)$ on $[a,b]$, which form an orthogonal basis for $L^2([a,b])$. The nodes y_i and weights w_i on $[-1,1]$ can be computed at cost $O(n)$ [9].

For an arbitrary $(2n+2)$-times continuously differentiable function f on $[a,b]$ (not necessarily a degree $2n+1$ polynomial), the same quadrature formula (35) has error

$$\left| \int_a^b f(x)dx - \sum_{i=1}^{n} f\left(\frac{b-a}{2}(y_i+1)+a\right)\frac{b-a}{2}w_i \right| \leq \frac{|f^{(2n+2)}(\xi)|}{(2n+2)!}\int_a^b \prod_{i=0}^{n}(x-x_i)^2 dx \tag{36}$$

for some $\xi \in [a,b]$; for a proof, see Theorem 7.4.5 in [6]. It follows immediately that the error can be bounded above by

$$\frac{(b-a)^{2n+3}}{(2n+2)!}\sup_{x\in[a,b]}|f^{(2n+2)}(x)|. \tag{37}$$

We also note that the Legendre polynomials P_n satisfy the three-term recurrence [6, 11]

$$P_{n+1}(x) = \frac{2n+1}{n+1}xP_n(x) - \frac{n}{n+1}P_{n-1}(x), \quad n \geq 1 \tag{38}$$

Using this recurrence, for any value of $x \in [-1,1]$ all values $P_0(x),\ldots,P_n(x)$ can be computed in $O(n)$ floating-point operations.

There is a one-to-one correspondence between the values of f on the n nodes $y_1,\ldots y_n$ and the first n Legendre coefficients of f. More precisely, given the values of f on n Gaussian nodes y_i of $[-1,1]$, the first n coefficients of the Legendre expansion of f can be computed at cost $O(n^2)$, by applying the matrix $[P_i(y_j)w_j]_{1\leq i,j\leq n}$ to the vector $(f(y_1),\ldots,f(y_n))^\top$. Conversely, from the first n Legendre coefficients of f we can compute the values of f on the n Gaussian nodes at cost $O(n^2)$, by applying the inverse matrix to the vector of coefficients.

2.5 Deferred corrections

In this section, we review the technique of deferred corrections for improving the accuracy of the solution to a linear system $\mathbf{A}\phi = f$. Informally, the method proceeds by solving a sequence of equations for the residuals of the previous solution. The setting where this method may be applied is when there already exists an inaccurate method for inverting $\mathbf{A}$ but a highly accurate method for applying $\mathbf{A}$ to an arbitrary vector. The algorithm is as follows:

1. Produce an initial solution $\hat{\phi}$.
2. Apply $\mathbf{A}$ to $\hat{\phi}$ and compute the residual right hand side $\Delta = f - \mathbf{A}\hat{\phi}$.
3. Compute a vector $\hat{r}$ that solves the residual system $\mathbf{A}r = \Delta$ for the residual $r = \phi - \hat{\phi}$. Update the solution: $\hat{\phi} \leftarrow \hat{\phi} + \hat{r}$.

4. Repeat steps 2 and 3 until convergence.

Algorithms based on this method have been employed for discretized second-kind integral operators $\mathbf{A}$ in, for example, [7, 10, 16]. Its convergence properties for certain differential equations have been studied [14]. When employed for general linear systems, the method also goes by the name "iterative refinement", where its numerical properites have been well-studied [15, 24].

At each step of the algorithm, the current solution $\hat{\phi}$ has a residual vector $r = \phi - \hat{\phi}$ that satisfies the residual linear system:

$$\mathbf{A}r = \mathbf{A}\phi - \mathbf{A}\hat{\phi} = f - \mathbf{A}\hat{\phi}. \tag{39}$$

Since $\mathbf{A}\hat{\phi}$ can be computed accurately, the right hand side $f - \mathbf{A}\hat{\phi}$ can be computed accurately as well (though see Remark 2 below). Consequently, the residual can be obtained to some relative accuracy ϵ, or in other words, we can produce a vector $\hat{r}$ with $\|\hat{r} - r\| \leq \epsilon\|r\|$. Then the updated solution $\hat{\phi} + \hat{r}$ satisfies:

$$\|(\hat{\phi} + \hat{r}) - \phi\| = \|\hat{r} - (\phi - \hat{\phi})\| = \|\hat{r} - r\| \leq \epsilon\|r\| = \epsilon\|\phi - \hat{\phi}\|. \tag{40}$$

In other words, the error of $\hat{\phi} + \hat{r}$ is smaller than the error of $\hat{\phi}$ by a factor of ϵ. If this factor is gained after every iteration, then only $\lceil \log(\epsilon^*)/\log(\epsilon) \rceil$ iterations are required to achieve machine precision ϵ^*.

Remark 2. We explain one subtlety with the method of deferred corrections. In order to solve for the residual r to relative error ϵ, the right hand side $\Delta = f - \mathbf{A}\hat{\phi}$ must be computed to relative error at most ϵ. Even if $\mathbf{A}\hat{\phi}$ is computed to full machine precision, if $\phi \approx \phi$ then $f \approx \mathbf{A}\hat{\phi}$, and the difference $f - \mathbf{A}\hat{\phi}$ may have large relative error. More precisely, when $\hat{\phi}$ is close enough to ϕ so that $\|f - \mathbf{A}\hat{\phi}\| \leq \delta\|f\|$, then the difference $f - \mathbf{A}\hat{\phi}$ will be computed to relative error ϵ^*/δ, due to loss of digits. So long as $\epsilon^*/\delta \leq \epsilon$, the next iteration of the algorithm will decrease the error by a factor of ϵ. However, when $\hat{\phi}$ is close enough to ϕ that $\delta < \epsilon^*/\epsilon$, the next iteration will only increase the accuracy by a factor of ϵ^*/δ.

3 The algorithm for solving (1)–(5)

In this section, we provide a detailed description of the algorithm for solving (1)–(5). The algorithm has three main components. First, the interval $[a, b]$ is broken into m equal-length subintervals $[x_i, x_{i+1}], 1 \leq i \leq m$, where $x_i = -1 + 2(i-1)/m$. A solution with zero boundary values is produced on each subinterval, employing the second-kind integral equation described in Section 2.3. In addition, four linearly independent solutions to the homogeneous equation are also produced on each subinterval.

Second, we form a linear combination of the solutions on each subinterval. The coefficients of this linear combination are chosen so that the resulting functions on adjacent intervals have matching boundary values. Solving for these coefficients is inexpensive but ill-conditioned, resulting in a solution that does not achieve machine precision.

Third, we remedy the ill-conditioning introduced in the second step by deferred corrections. To do so we must accurately apply the second-kind integral operator $\mathbf{L}\mathbf{G}_0$ to the right hand side of the equation; concretely, this entails integrating the right hand side against the functions G_j. We describe an algorithm for doing so with asymptotic cost $O(m)$. This step is then iterated until full machine accuracy is achieved.

The algorithm can be summarized as follows:

1. Solve the problem with zero boundary conditions on each subinterval $[x_i, x_{i+1}]$.
2. Adjust the boundary values on each subinterval $[x_i, x_{i+1}]$ so that the solutions match at the endpoints.
3. Compute the residual right hand side on the full domain $[a, b]$ and apply deferred corrections.

Steps 1–3 will produce the solution ϕ sampled at n Gaussian nodes of each subinterval $[x_i, x_{i+1}]$. These n nodes can be converted into the first n Legendre coefficients of the solution ϕ on each subinterval $[x_i, x_{i+1}]$. The solution can then be evaluated at any point in $[a, b]$ by evaluating this expansion on the subinterval $[x_i, x_{i+1}]$ containing that point, as described in Section 2.4.

Finally, we note that by applying the rescaling described in Section 2.1, we may assume that the problem domain is the interval $[-1, 1]$. After producing the solution, we can apply the inverse rescaling (17) to return to the original domain $[a, b]$.

3.1 The solution on n nodes in a single subinterval

In this section, we describe how to obtain a solution ϕ on n Gaussian nodes of a subinterval $[x_i, x_{i+1}]$. We will rescale the subinterval $[x_i, x_{i+1}]$ to be $[-1, 1]$, as described in Section 2.1; when the algorithm is completed, we will rescale back to the original domain $[x_i, x_{i+1}]$, as in equation (17). We explain how to produce the solution ϕ, with specified boundary values, to the equation $\mathbf{L}\phi = f$ on n Gaussian nodes $y_1, \ldots, y_n$ in $[-1, 1]$. We will also assume that we have divided the entire equation (1) by the leading coefficient $a_4(x)$, so that without any loss in generality we assume that the leading coefficient $a_4(x) = 1$.

As described in Section 2.3, we can write the solution ϕ as $\phi = \mathbf{G}_0\sigma + \psi_\alpha$, where ψ_α is defined by formulas (26)–(30), and the function σ satisfies the following second-kind integral equation:

$$\sigma(x) + \int_{-1}^{1} \sum_{j=0}^{3} a_j(x) G_j(x, t)\sigma(t)dt = f - \mathbf{L}\psi_\alpha \equiv f_\alpha. \tag{41}$$

We discretize the equation (41) by sampling the coefficients $a_j(x)$, the right hand side $f_\alpha(x)$, and the functions $G_j(x, t)$ in each variable on the n Gaussian nodes $y_1, \ldots, y_n$. We will produce the solution σ evaluated on the same nodes. We will write the integrals in (41) using the quadrature formula for Gaussian nodes and weights w_k described in

Section 2.4. The discrete system of equations resulting from discretizing (41) is given by:

$$\sigma(y_i) + \sum_{k=1}^{n} \left[\sum_{j=0}^{3} a_j(y_i) G_j(y_i, y_k) \right] \sigma(y_k) w_k = f_\alpha(y_i), \quad 1 \le i \le n. \tag{42}$$

We can compactly write this as a linear system $A\hat{\sigma} = \hat{f}_\alpha$, where $\hat{f}_\alpha = (f_\alpha(y_1), \ldots, f_\alpha(y_n))^\top$, $\hat{\sigma} = (\sigma(y_1), \ldots, \sigma(y_n))^\top$ and A is the n-by-n matrix with entries

$$A(i,k) = \delta_{i,k} + \sum_{j=0}^{3} a_j(y_i) G_j(y_i, y_k) w_k. \tag{43}$$

We invert the n-by-n matrix A using the QR factorization at a cost of $O(n^3)$ floating point operations [6]; other algorithms for solving linear systems may be used as well.

The solution ϕ to (1)–(5) and its first three derivatives on the n nodes $y_1, \ldots, y_n$ are now given by:

$$\phi^{(j)}(y_i) = \sum_{k=1}^{n} G_j(y_i, y_k) \hat{\sigma}_k w_k + \psi_\alpha^{(j)}(y_i), \quad 0 \le j \le 3 \tag{44}$$

and $\phi^{(4)}(y_i) = \hat{\sigma}_i$. We then perform the inverse rescaling in equation (17) to complete the computation of the solution on the subinterval $[x_i, x_{i+1}]$. Since the cost of solving (42) on each of the m subintervals $[x_i, x_{i+1}]$ is $O(n^3)$, the total cost is $O(mn^3)$.

3.2 Matching the boundary values

In Section 3.1 we detailed how to obtain a solution to (1)–(5) on n Gaussian nodes in each subinterval $[x_i, x_{i+1}]$ with specified boundary values at x_i and x_{i+1}. We will denote by $\tilde{\phi}_i$ the solution on $[x_i, x_{i+1}]$. In this section we explain how to adjust the boundary values of these functions so that the resulting functions ϕ_i and ϕ_{i+1} have matching values and three derivatives at the interface x_{i+1}, and so that ϕ_1 satisfies (2) and (4) at -1, and ϕ_m satisfies (3) and (5) at 1. We will assume that the $\tilde{\phi}_i$ were chosen to have zero boundary values at x_i and x_{i+1}.

In addition to constructing $\tilde{\phi}_i$, on each subinterval $[x_i, x_{i+1}]$, we use the method of Section 3.1 to solve the equation $\mathbf{L}g = 0$ four times to obtain functions $g_{i,j}, 1 \le j \le 4$, with a single non-zero boundary value; concretely, $g_{i,1}(x_i) = 1, g_{i,2}(x_{i+1}) = 1, g'_{i,3}(x_i) = 1$, and $g'_{i,4}(x_{i+1}) = 1$, and the other values and first derivatives at x_i and x_{i+1} are zero. We will find coefficients $\beta_{i,j}, 1 \le i \le m, 1 \le j \le 4$, such that the functions

$$\phi_i(x) = \tilde{\phi}_i(x) + \sum_{j=1}^{4} \beta_{i,j} g_{i,j}(x) \tag{45}$$

have matching boundary values, and the desired values at ± 1 given by (2)–(5). The derivatives of ϕ_i are found by adding the same linear combination of the derivatives of the $g_{i,j}$, namely:

$$\phi_i^{(k)}(x) = \tilde{\phi}_i^{(k)}(x) + \sum_{j=1}^{4} \beta_{i,j} g_{i,j}^{(k)}(x), \quad 1 \le k \le 4. \tag{46}$$

The function $\phi^{(k)}(x)$, $0 \le k \le 4$, is then defined by $\phi_i^{(k)}(x)$ when $x \in [x_i, x_{i+1}]$. In particular, $\sigma = \phi^{(4)}$ solves the integral equation (41) on the entire interval $[-1, 1]$.

The coefficients $\beta_{i,j}$ can be found as the solution to a banded linear system of size $4m$-by-$4m$. For every pair of adjacent intervals $[x_i, x_{i+1}]$ and $[x_{i+1}, x_{i+2}]$, we require that

$$\tilde{\phi}_i(x_{i+1}) + \beta_{i,2} = \tilde{\phi}_{i+1}(x_{i+1}) + \beta_{i,1} \tag{47}$$

$$\tilde{\phi}_i'(x_{i+1}) + \beta_{i,4} = \tilde{\phi}_{i+1}'(x_{i+1}) + \beta_{i,3} \tag{48}$$

$$\tilde{\phi}_i''(x_{i+1}) + \sum_{j=1}^{4} \beta_{i,j} g_{i,j}''(x_{i+1}) = \tilde{\phi}_{i+1}''(x_{i+1}) + \sum_{j=1}^{4} \beta_{i+1,j} g_{i+1,j}''(x_{i+1}) \tag{49}$$

$$\tilde{\phi}_i^{(3)}(x_{i+1}) + \sum_{j=1}^{4} \beta_{i,j} g_{i,j}^{(3)}(x_{i+1}) = \tilde{\phi}_{i+1}^{(3)}(x_{i+1}) + \sum_{j=1}^{4} \beta_{i+1,j} g_{i+1,j}^{(3)}(x_{i+1}) \tag{50}$$

These conditions ensure that the functions ϕ_i have four matching derivatives at the interfaces $x_{i+1}, 1 \le i \le m-1$. To ensure the boundary conditions (2)–(5) on $[a, b]$, we also require the following equations:

$$\tilde{\phi}_1(-1) + \beta_{1,1} = \alpha_{l,0} \tag{51}$$

$$\tilde{\phi}_1'(-1) + \beta_{1,3} = \alpha_{l,1} \tag{52}$$

$$\tilde{\phi}_m(1) + \beta_{m,2} = \alpha_{r,0} \tag{53}$$

$$\tilde{\phi}_m'(1) + \beta_{m,4} = \alpha_{r,1}. \tag{54}$$

Ordering the variables $\beta_{1,1}, \beta_{1,2}, \beta_{1,3}, \beta_{1,4}, \ldots\ldots, \beta_{m,1}, \beta_{m,2}, \beta_{m,3}, \beta_{m,4}$, and ordering the equations as above and by the intervals they involve, the linear system described by equations (47)–(54) is 9-diagonal and so can be solved in $O(m)$ floating-point operations. In Figure 1, we plot this matrix for $m = 8$ subintervals. After solving for the coefficients $\beta_{i,j}$, the solutions ϕ_i on each subinterval can then be obtained by equations (45) and (46), at an additional cost of $O(nm)$.

Remark 3. Because the initial local solutions $\tilde{\phi}_i$ on each subinterval $[x_i, x_{i+1}]$ are obtained via the second-kind integral formulation described in Section 2.3, they can be computed with high accuracy. However, the method we have described in this section for matching their boundary values departs from this integral equation formulation. In practice, we have observed that the banded linear system defined by equations (48)–(54) behaves like the matrices encountered in finite element or finite difference schemes, in that it has a condition number of size $O(m^4)$. To achieve greater accuracy, in Section 3.3 we show how to apply deferred corrections to this problem.

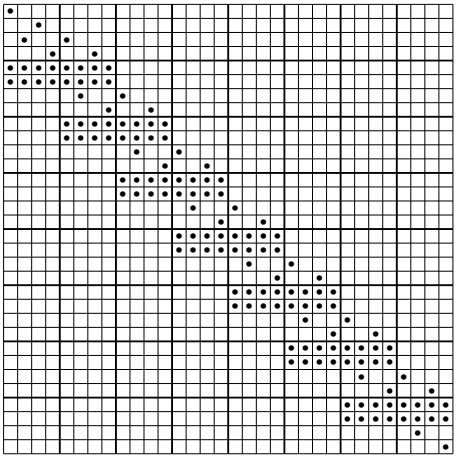

Figure 1: The banded matrix used to match the boundary values, shown here for $m = 8$ subintervals; non-zero values are marked with dots.

3.3 Applying deferred corrections

The solution we obtain from Sections 3.1 and 3.2 will not be accurate to the precision permitted by the problem, as explained in Remark 3. To improve the accuracy, we apply the method of deferred corrections, as described in Section 2.5. If $\hat{\sigma}$ is the vector of length mn with the estimated solution to the integral equation (41) evaluated on the mn nodes, then to apply deferred corrections we perform the following steps:

1. Apply the operator $\mathbf{LG}_0$ to $\hat{\sigma}$, evaluated on the mn nodes.
2. Solve the new system $\mathbf{LG}_0 r = f_\alpha - \mathbf{LG}_0\hat{\sigma}$, obtaining an estimated residual $\hat{r}$.
3. Update the solution: $\hat{\sigma} \leftarrow \hat{\sigma} + \hat{r}$.

Step 2 is performed by following Sections 3.1 and 3.2 at a cost of $O(mn^3)$ floating point operations, and Step 3 requires mn additions. We will now exhibit an algorithm for performing Step 1 with asymptotic cost $O(mn^2)$.

Because $a_4 \equiv 1$, from equation (32) we have

$$(\mathbf{LG}_0\hat{\sigma})(x) = \hat{\sigma}(x) + \sum_{j=0}^{3} a_j(x) \int_{-1}^{1} G_j(x,t)\hat{\sigma}(t)dt. \tag{55}$$

Therefore, we must integrate $\hat{\sigma}(x)$ against the functions $G_j(x,t)$. We now explain how to compute the integrals $\int_{-1}^{1} G_j(x,t)\hat{\sigma}(t)dt$ with $O(mn^2)$ floating-point operations. The function $G_j(x,t)$ can be written in the form

$$G_j(x,t) = \begin{cases} \sum_{i=0}^{3-j} x^i p_i(t), & \text{if } x < t \\ \sum_{i=0}^{3-j} x^i q_i(t), & \text{if } t < x \end{cases} \tag{56}$$

where p_i and q_i are polynomials of degree $3-j$ that can be explicitly computed from (18). (Of course, p_i and q_i depend on j, though we suppress this for notational simplicity.) Using (56), we obtain:

$$\int_{-1}^{1} G_j(x,t)\hat{\sigma}(t)dt = \sum_{i=0}^{3-j} x^i \left\{ \int_{-1}^{x} q_i(t)\hat{\sigma}(t)dt + \int_{x}^{1} p_i(t)\hat{\sigma}(t)dt \right\} \tag{57}$$

We must compute $\int_{-1}^{1} G_j(x,t)\hat{\sigma}(t)dt$ for each of the mn values of x on which we have discretized the problem – the n Gaussian nodes on each of the m subintervals $[x_i, x_{i+1}]$. For each such x, let us suppose $x \in [x_{i^*}, x_{i^*+1}]$. We can then write:

$$\int_{-1}^{x} q_i(t)\hat{\sigma}(t)dt = \sum_{k=1}^{i^*-1} \int_{x_k}^{x_{k+1}} q_i(t)\hat{\sigma}(t)dt + \int_{x_{i^*}}^{x} q_i(t)\hat{\sigma}(t)dt \tag{58}$$

and similarly

$$\int_{x}^{1} p_i(t)\hat{\sigma}(t)dt = \sum_{k=i^*+1}^{m} \int_{x_k}^{x_{k+1}} p_i(t)\hat{\sigma}(t)dt + \int_{x}^{x_{i^*+1}} p_i(t)\hat{\sigma}(t)dt. \tag{59}$$

Each of the $2m$ integrals $\int_{x_k}^{x_{k+1}} q_i(t)\hat{\sigma}(t)dt$ and $\int_{x_k}^{x_{k+1}} p_i(t)\hat{\sigma}(t)dt$ can be computed using an n-point Gaussian quadrature, at a total cost of $O(mn)$ floating-point operations. For each Gaussian node x in the interval $[x_{i^*}, x_{i^*+1}]$, the integrals $\int_{x_{i^*}}^{x} q_i(t)\hat{\sigma}(t)dt$ and $\int_{x}^{x_{i^*+1}} p_i(t)\hat{\sigma}(t)dt$ can also be computed using an n-point Gaussian quadrature, at a total cost of $O(mn^2)$ floating-point operations.

From (57), the integrals $\int_{-1}^{1} G_j(x,t)\hat{\sigma}(t)dt$, and hence the evaluation of $(\mathbf{L}\mathbf{G}_0\hat{\sigma})(x)$, can be computed at cost $O(mn^2)$. Consequently, each iteration of deferred corrections requires $O(mn^2)$ operations. As explained in Section 2.5, the number of iterations is $O(\log(1/\epsilon))$, where ϵ is the machine precision. This brings the asymptotic running time of the entire algorithm to $O(mn^3 \log(1/\epsilon))$.

4 Numerical results

The algorithm of Section 3 has been implemented in Fortran. In this section, we use the algorithm to obtain the numerical solution ϕ to several specific problems of the form (1)–(5). All experiments were performed on a Dell XPS-L521X laptop computer with an Intel Core i7-3632QM CPU at 2.20GHz with 15.6 GB RAM, running the 64-bit Ubuntu 14.04 LTS operating system. In our experiments, the code was compiled with the gfortran compiler using extended (16-bit) precision; this is called with the compilation flag `-freal-8-real-16`.

In the presentation of the CPU times of the experiments, we distinguish between the preprocessing steps – which depend only on the differential operator $\mathbf{L}$ and which we refer to as "factorizing" $\mathbf{L}$ – and the actual solution of (1)–(5) for a specified right side f and boundary values $\alpha_{l,0}$, $\alpha_{r,0}$, $\alpha_{l,1}$, $\alpha_{r,1}$. Factorizing $\mathbf{L}$ consists of constructing the

mn Gaussian nodes and weights on the interval $[a, b]$; computing the QR factorization of the discretized operators $\mathbf{G}_j$, $0 \leq j \leq 3$; and computing the values of the functions (27)–(30), and their derivatives, on the Gaussian nodes.

Five experiments are presented below, and their results are contained in Tables 1–15. In each experiment, we solve a specific boundary value problem on a specified interval $[a, b]$, and measure the relative error of the solution and its first four derivatives on a grid of $N = 10,000$ equispaced points $x_1, \ldots, x_N$ on $[a, b]$, including the endpoints a and b. If $\hat{\phi}^{(j)}$ is the estimated j^{th} derivative, then the relative error is defined as follows:

$$R(\phi^{(j)}) = \sqrt{\frac{\sum_{i=1}^{N} |\hat{\phi}^{(j)}(x_i) - \phi^{(j)}(x_i)|^2}{\sum_{i=1}^{N} |\phi^{(j)}(x_i)|^2}} \tag{60}$$

For each experiment, we display three tables. In each table, the first column contains the number m of subintervals of $[a, b]$ used for that experiment. In the first table, the remaining five columns display the errors of the solution and its first four derivatives. In the second table, the remaining three columns show the factorization time, the solution time, and the total CPU time. In the third table, the remaining columns display the relative size of the residual norm after each iteration of the algorithm, to demonstrate the convergence of deferred corrections.

Remark 4. For each example, the relative errors $R(\phi^{(j)})$ decrease by approximately a fixed number of digits whenever m is doubled. This behavior is expected from the error of the Gaussian quadrature, as described in Section 2.4, equation (37), since the quadrature error should scale like $m^{-(2n+2)}$ (the exponent is not $-(2n+3)$ because we multiply by m, the number of subintervals).

Remark 5. For each example, the CPU times – both for the factorization of $\mathbf{L}$ and the solution of the equation – scale approximately linearly with m. This behavior is expected from the $O(m)$ asymptotic running time of the algorithm.

Remark 6. For each example and for each value of m, the residual size decreases by an approximately constant number of digits after approximately two iterations of deferred corrections. This is likely because two linear systems are solved each step – the local linear systems on each subinterval, and the global linear system that matches the local solutions' boundary values. Up to taking two steps instead of one, this rate of decrease in the residual size is what we expect from Section 2.5. In particular, very few iterations of the algorithm are required until convergence is achieved.

4.1 The function $\phi(x) = \sin(5x)$

The first example is the following simple, well-conditioned equation:

$$\sum_{j=0}^{4} (1 + x^{4-j}) \phi^{(j)}(x) = \sum_{j=0}^{4} (1 + x^{4-j}) \frac{d^j}{dx^j} \sin(5x) \tag{61}$$

on the interval $[0, 2\pi]$, subject to the boundary conditions $\alpha_{l,0} = 0$, $\alpha_{r,0} = 0$, $\alpha_{l,1} = 5$ and $\alpha_{r,1} = 5$. The solution is apparently $\phi(x) = \sin(5x)$. This example is included to illustrate that for well-conditioned problems, our algorithm obtains the solution to full machine accuracy. The algorithm was run using $n = 10$ Gaussian nodes per subinterval. The results are displayed in Tables 1–3.

4.2 The function $\phi(x) = \sin(150x)$

For the second example, we solve the following variation on equation (61):

$$\sum_{j=0}^{4}(1 + x^{4-j})\phi^{(j)}(x) = \sum_{j=0}^{4}(1 + x^{4-j})\frac{d^j}{dx^j}\sin(150x) \tag{62}$$

on the interval $[0, 2\pi]$, subject to the boundary conditions $\alpha_{l,0} = 0$, $\alpha_{r,0} = 0$, $\alpha_{l,1} = 150$ and $\alpha_{r,1} = 150$. The solution is apparently $\phi(x) = \sin(150x)$. However, since the frequency is much higher, this is a worse-conditioned problem than (61), and so cannot be solved to full machine precision independent of the choice of algorithm. However, our algorithm successfully recovers the solution to within the error permitted by the problem's condition number. The algorithm was run using $n = 15$ Gaussian nodes per subinterval. The results are displayed in Tables 4–6.

4.3 Beam with fixed ends

In this example, we consider the problem of determining the shape of a beam with non-uniform stiffness subjected to an external force. We parametrize the x-axis by the interval $[0, 1]$. We take the stiffness of the beam to be the function $c(x)$ given by:

$$c(x) = (x - 1/2)^2 + 1 \tag{63}$$

and the external force to be the function $f(x)$ given by:

$$f(x) = \sin(2\pi x) + 1. \tag{64}$$

If $\phi(x)$ is the shape of the beam, then ϕ satisfies the following differential equation [8]:

$$\frac{d^2}{dx^2}\left(c(x)\frac{d^2\phi}{dx^2}\right) = f(x) \tag{65}$$

or equivalently

$$((x - 1/2)^2 + 1)\frac{d^4\phi}{dx^4}(x) + 4(x - 1/2)\frac{d^3\phi}{dx^3}(x) + 2\frac{d^2\phi}{dx^2}(x) = \sin(2\pi x) + 1. \tag{66}$$

We impose fixed endpoints on the beam, meaning that $\alpha_{l,0} = 0$, $\alpha_{r,0} = 0$, $\alpha_{l,1} = 0$ and $\alpha_{r,1} = 0$. The algorithm was run using $n = 10$ Gaussian nodes per subinterval. The results are displayed in Tables 7–9. The solution is plotted on the left side of Figure 2.

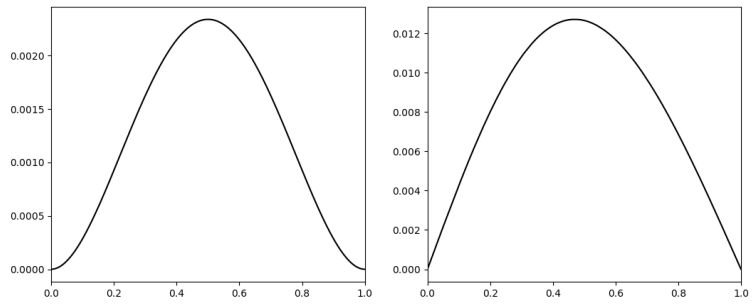

Figure 2: Left: the beam with fixed ends. Right: the beam with simply-supported ends.

4.4 Beam with simply-supported ends

In this example, we consider the same beam-bending problem (66) as in Section 4.3, except we require that the ends of the beam are simply-supported, or $\phi(0) = \phi(1) = 0$ and $\phi''(0) = \phi''(1) = 0$. Because the boundary vales involve the second derivatives, this problem does not immediately fit into the form of (1)–(5), and we include it as an example for that reason.

To introduce the correct boundary conditions, we let $\mathbf{L}$ denote the differential operator on the left side of (66), and we solve the equation $\mathbf{L}\phi = f$ with boundary conditions $\alpha_{l,0} = \alpha_{r,0} = \alpha_{l,1} = \alpha_{r,1} = 0$; we will call this function $\tilde{\phi}$. We also produce four linearly independent solutions to the equation $\mathbf{L}\phi = 0$ by solving it with linearly independent boundary conditions; we will call these solutions ϕ_1, ϕ_2, ϕ_3 and ϕ_4.

We will find coefficients b_i, $1 \leq i \leq 4$, so that the function

$$\phi(x) = \tilde{\phi}(x) + \sum_{i=1}^{4} b_i \phi_i(x) \tag{67}$$

satisfies the sought-after boundary conditions $\phi(0) = \phi(1) = 0$ and $\phi''(0) = \phi''(1) = 0$. The b_i satisfy the linear equations:

$$0 = \sum_{i=1}^{4} b_i \phi_i(0) \tag{68}$$

$$0 = \sum_{i=1}^{4} b_i \phi_i(1) \tag{69}$$

$$0 = \tilde{\phi}''(0) + \sum_{i=1}^{4} b_i \phi_i''(0) \tag{70}$$

$$0 = \tilde{\phi}''(1) + \sum_{i=1}^{4} b_i \phi_i''(1). \tag{71}$$

This system is non-singular because the ϕ_i are linearly independent. This permits us to solve for the b_i, and then define the solution ϕ by (67). The algorithm was run using $n = 10$ Gaussian nodes per subinterval. The results are shown in Tables 10–12. The solution is plotted on the right side of Figure 2.

4.5 The Bessel function $J_{10}(x)$

In our final example, we solve for the Bessel function $J_{10}(x)$ on the interval $[0, 100]$, which is plotted in Figure 3. J_{10} satisfies the second-order equation:

$$x^2 \frac{d^2}{dx^2}\phi(x) + x\frac{d}{dx}\phi(x) + (x^2 - 100)\phi(x) = 0. \tag{72}$$

We differentiate this equation twice to arrive at the fourth-order equation:

$$\left(x^2 \frac{d^4}{dx^4} + 5x\frac{d^3}{dx^3} + (x^2 - 96)\frac{d^2}{dx^2} + 4x\frac{d}{dx} + 2\right)\phi(x) = 0. \tag{73}$$

We solve this equation subject to the boundary values matching the known values of $J_{10}(x)$ and $J'_{10}(x)$ at 0 and 100. Because of the singularity at $x = 0$, we represent the 0 endpoint by the square root of machine zero. We ran the algorithm with $n = 20$ nodes per subinterval. We include this example to demonstrate the algorithm's performance when the leading coefficients has a root, as does the leading coefficient x^2 at the left endpoint $x = 0$. Because of this singularity, the solution cannot be obtained to full machine precision even on the mn Gaussian nodes themselves. Nevertheless, our algorithm obtains the solution up to this necessary loss of accuracy. The results are shown in Tables 13–15.

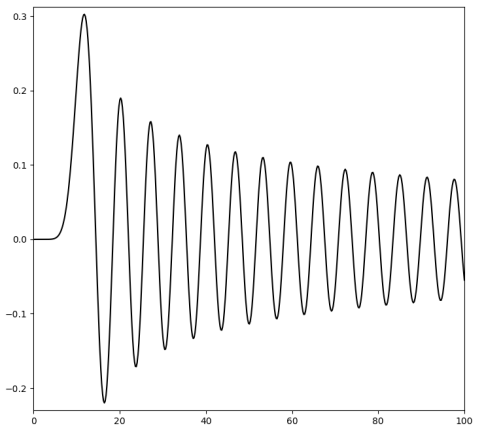

Figure 3: The Bessel function J_{10} on the interval $[0, 100]$.

Acknowledgements

William Leeb was supported by a postdoctoral fellowship from the Simons Collaboration on Algorithms and Geometry. Vladimir Rokhlin was supported in part by the Office of Naval Research under the grant N00014-16-1-2123 and the Air Force Office of Scientific Research under the grant FA9550-16-1-0175.

References

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover Publications, Inc., 1972.

[2] Mario Bertero and F. Alberto Grunbaum. Commuting differential operators for the finite Laplace transform. *Inverse Problems*, 1:181–192, 1985.

[3] Mario Bertero and F. Alberto Grunbaum. Spectral properties of a differential operator related to the inversion of the finite Laplace transform. *Inverse Problems*, 2:131–139, 1986.

[4] Frederick W. Byron and Robert W. Fuller. *Mathematics of Classical and Quantum Physics*. Courier Corporation, 2012.

[5] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume I. Interscience Publishers, 1937.

[6] G. Dahlquist and A. Bjorck. *Numerical Methods*. Prentice Hall, Inc., 1974.

[7] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.

[8] James M. Gere and Stephen P. Timoshenko. *Mechanics of Materials*. PWS Engineering, 2nd edition, 1984.

[9] A. Glaser, X. Liu, and V. Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM Journal of Scientific Computation*, 29(4):1420–1438, 2007.

[10] Andreas Glaser and Vladimir Rokhlin. A new class of highly accurate solvers for ordinary differential equations. *Journal of Scientific Computing*, 38(3):368–399, 2009.

[11] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, 8th edition, 2015.

[12] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[13] Leslie Greengard and Vladimir Rokhlin. On the numerical solution of two-point boundary value problems. *Communications on Pure and Applied Mathematics*, 44(4):419–452, 1991.

[14] Anders C. Hansen and John Strain. On the order of deferred correction. *Applied Numerical Mathematics*, 61(8):961–973, 2011.

[15] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, 2nd edition, 2002.

[16] Dan Kushnir and Vladimir Rokhlin. A highly accurate solver for stiff ordinary differential equations. *SIAM Journal on Scientific Computing*, 34(3):A1296–A1315, 2012.

[17] R. R. Lederman. *On the Analytical and Numerical Properties of the Truncated Laplace Transform*. PhD thesis, Yale University, May 2014.

[18] Roy R. Lederman and Vladimir Rokhlin. On the analytical and numerical properties of the truncated Laplace transform I. *SIAM Journal on Numerical Analysis*, 53(3):1214–1235, 2015.

[19] Roy R. Lederman and Vladimir Rokhlin. On the analytical and numerical properties of the truncated Laplace transform, part II. *SIAM Journal on Numerical Analysis*, 54(2):665–687, 2016.

[20] June-Yub Lee and Leslie Greengard. A fast adaptive numerical method for stiff two-point boundary value problems. *SIAM Journal on Scientific Computing*, 18(2):403–429, 1997.

[21] Frigyes Riesz and Béla Sz.-Nagy. *Functional Analysis*. Dover Publications, 1955.

[22] Page Starr and Vladimir Rokhlin. On the numerical solution of two-point boundary value problems II. *Communications on Pure and Applied Mathematics*, 47(8):1117–1159, 1994.

[23] Gilbert Strang and George Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, 2nd edition, 2008.

[24] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, 1963.

Table 1: Relative errors, for $\sin(5x)$

m	$R(\phi)$	$R(\phi')$	$R(\phi'')$	$R(\phi^{(3)})$	$R(\phi^{(4)})$
16	0.2722E-09	0.2723E-09	0.2722E-09	0.2723E-09	0.2734E-09
32	0.2697E-12	0.2697E-12	0.2697E-12	0.2697E-12	0.2700E-12
64	0.2640E-15	0.2646E-15	0.2640E-15	0.2646E-15	0.2641E-15
128	0.2581E-18	0.2586E-18	0.2581E-18	0.2586E-18	0.2582E-18
256	0.2521E-21	0.2526E-21	0.2521E-21	0.2526E-21	0.2521E-21
512	0.2462E-24	0.2467E-24	0.2462E-24	0.2467E-24	0.2462E-24
1024	0.2405E-27	0.2409E-27	0.2405E-27	0.2409E-27	0.2405E-27
2048	0.1607E-29	0.4084E-30	0.2694E-30	0.2604E-30	0.2575E-30

Table 2: CPU times, for $\sin(5x)$

m	Factorizing	Solving	Total
16	0.6800E-01	0.3200E-01	0.1000E+00
32	0.9600E-01	0.3600E-01	0.1320E+00
64	0.1840E+00	0.7200E-01	0.2560E+00
128	0.3560E+00	0.1400E+00	0.4960E+00
256	0.6920E+00	0.2200E+00	0.9120E+00
512	0.1356E+01	0.3480E+00	0.1704E+01
1024	0.2712E+01	0.6800E+00	0.3392E+01
2048	0.5348E+01	0.1332E+01	0.6680E+01

Table 3: Relative residual sizes, for $\sin(5x)$

	Iteration number						
m	1	2	3	4	5	6	7
16	0.9187E-09	0.1162E-10	0.4903E-19	0.1987E-20	0.1204E-28	0.4934E-30	0.6651E-33
32	0.9092E-12	0.1647E-13	0.7903E-25	0.3993E-26	0.4482E-33		
64	0.8907E-15	0.1742E-16	0.8514E-31	0.4428E-32	0.5743E-33		
128	0.8706E-18	0.1734E-19	0.4401E-33	0.4636E-33	0.8539E-33		
256	0.8503E-21	0.1701E-22	0.3492E-33	0.1134E-32			
512	0.8302E-24	0.1663E-25	0.1295E-32				
1024	0.1037E-26	0.1622E-28	0.1044E-32				
2048	0.6554E-25	0.1733E-29	0.1059E-32				

Table 4: Relative errors, for $\sin(150x)$

m	$R(\phi)$	$R(\phi')$	$R(\phi'')$	$R(\phi^{(3)})$	$R(\phi^{(4)})$
16	0.1000E+01	0.1000E+01	0.1000E+01	0.1005E+01	0.1315E+01
32	0.4264E+00	0.3559E+00	0.3603E+00	0.3704E+00	0.3360E+00
64	0.1348E-03	0.1322E-03	0.1351E-03	0.1351E-03	0.1347E-03
128	0.7434E-08	0.7380E-08	0.7435E-08	0.7394E-08	0.7434E-08
256	0.2604E-12	0.2624E-12	0.2604E-12	0.2628E-12	0.2604E-12
512	0.8366E-17	0.8177E-17	0.8366E-17	0.8183E-17	0.8366E-17
1024	0.2608E-21	0.2485E-21	0.2608E-21	0.2484E-21	0.2608E-21
2048	0.6758E-25	0.7789E-26	0.7803E-26	0.7780E-26	0.7803E-26
4096	0.3132E-25	0.2447E-27	0.1476E-29	0.2507E-30	0.7206E-30

Table 5: CPU times, for $\sin(150x)$

m	Factorizing	Solving	Total
16	0.1640E+00	0.4400E-01	0.2080E+00
32	0.2440E+00	0.7200E-01	0.3160E+00
64	0.4600E+00	0.1200E+00	0.5800E+00
128	0.9080E+00	0.2440E+00	0.1152E+01
256	0.1764E+01	0.4640E+00	0.2228E+01
512	0.3496E+01	0.9160E+00	0.4412E+01
1024	0.6920E+01	0.1800E+01	0.8720E+01
2048	0.1384E+02	0.2984E+01	0.1682E+02
4096	0.2752E+02	0.4752E+01	0.3228E+02

Table 6: Relative residual sizes, for $\sin(150x)$

	Iteration number						
m	1	2	3	4	5	6	7
16	0.9081E-02	0.1615E-05	0.1599E-20	0.1696E-22	0.2960E-33	0.5246E-33	0.3061E-33
32	0.2990E-01	0.7195E-06	0.2276E-24	0.3397E-27	0.2127E-33	0.2420E-33	
64	0.1261E-04	0.3152E-09	0.2517E-32	0.2444E-33	0.1341E-33		
128	0.2261E-08	0.9298E-14	0.1571E-33	0.1707E-33	0.2110E-33		
256	0.1336E-13	0.1058E-17	0.6158E-33	0.2061E-33	0.1354E-33		
512	0.1165E-17	0.2580E-22	0.4351E-33	0.1369E-33	0.1889E-33		
1024	0.9517E-22	0.4374E-27	0.2364E-33	0.1809E-33	0.1864E-33		
2048	0.6160E-26	0.6838E-32	0.2441E-33	0.1405E-33			
4096	0.4036E-30	0.1736E-33	0.1630E-33				

Table 7: Relative errors, for beam with fixed ends

m	$R(\phi)$	$R(\phi')$	$R(\phi'')$	$R(\phi^{(3)})$	$R(\phi^{(4)})$
2	0.2671E-07	0.7362E-07	0.6648E-07	0.1843E-06	0.1262E-06
4	0.3026E-10	0.5149E-10	0.1417E-09	0.5111E-10	0.6806E-09
8	0.2659E-13	0.7074E-13	0.1093E-12	0.1548E-12	0.4938E-12
16	0.2608E-16	0.6953E-16	0.1071E-15	0.1525E-15	0.4865E-15
32	0.2551E-19	0.6791E-19	0.1048E-18	0.1490E-18	0.4763E-18
64	0.2492E-22	0.6635E-22	0.1023E-21	0.1456E-21	0.4652E-21
128	0.2432E-25	0.6484E-25	0.9986E-25	0.1423E-24	0.4541E-24
256	0.2375E-28	0.6331E-28	0.9750E-28	0.1390E-27	0.4434E-27
512	0.6995E-31	0.1115E-30	0.1439E-30	0.1385E-30	0.4329E-30
1024	0.6634E-31	0.9183E-31	0.1071E-30	0.2835E-31	0.1263E-31

Table 8: CPU times, for beam with fixed ends

m	Factorizing	Solving	Total
2	0.1200E-01	0.4000E-02	0.1600E-01
4	0.2000E-01	0.8000E-02	0.2800E-01
8	0.3200E-01	0.8000E-02	0.4000E-01
16	0.5200E-01	0.2000E-01	0.7200E-01
32	0.9600E-01	0.2000E-01	0.1160E+00
64	0.1840E+00	0.4000E-01	0.2240E+00
128	0.3480E+00	0.8000E-01	0.4280E+00
256	0.6840E+00	0.1600E+00	0.8440E+00
512	0.1340E+01	0.3240E+00	0.1664E+01
1024	0.2680E+01	0.6120E+00	0.3292E+01

Table 9: Relative residual sizes, for beam with fixed ends

	Iteration number						
m	1	2	3	4	5	6	7
2	0.5076E-07	0.4138E-07	0.6825E-15	0.1913E-15	0.5263E-23	0.8843E-24	0.3413E-31
4	0.8287E-10	0.2094E-10	0.5314E-21	0.6000E-22	0.2523E-32		
8	0.1190E-12	0.5092E-14	0.2206E-27	0.5444E-29	0.1133E-33		
16	0.1153E-15	0.1512E-17	0.2875E-33	0.9051E-34	0.6140E-34		
32	0.1124E-18	0.4163E-20	0.1240E-33				
64	0.1097E-21	0.5265E-23	0.1185E-33				
128	0.1071E-24	0.5699E-26	0.1167E-33				
256	0.6699E-27	0.1489E-28	0.2177E-33				
512	0.8823E-26	0.3535E-28	0.2681E-33				
1024	0.1172E-25	0.5591E-28	0.2968E-33				

Table 10: Relative errors, for beam with simply-supported ends

m	$R(\phi)$	$R(\phi')$	$R(\phi'')$	$R(\phi^{(3)})$	$R(\phi^{(4)})$
2	0.2890E-07	0.3157E-07	0.4856E-07	0.1670E-06	0.1163E-06
4	0.2783E-10	0.4309E-10	0.1095E-09	0.1030E-09	0.6114E-09
8	0.2797E-13	0.4355E-13	0.1036E-12	0.1659E-12	0.4446E-12
16	0.2578E-16	0.3880E-16	0.9211E-16	0.1576E-15	0.4378E-15
32	0.2405E-19	0.3548E-19	0.8416E-19	0.1508E-18	0.4284E-18
64	0.2286E-22	0.3340E-22	0.7917E-22	0.1458E-21	0.4184E-21
128	0.2200E-25	0.3199E-25	0.7580E-25	0.1417E-24	0.4083E-24
256	0.2132E-28	0.3091E-28	0.7330E-28	0.1379E-27	0.3987E-27
512	0.1556E-30	0.1089E-30	0.1365E-30	0.1429E-30	0.3896E-30
1024	0.1551E-30	0.1098E-30	0.1018E-30	0.4602E-31	0.1762E-31

Table 11: CPU times, for beam with simply-supported ends

m	Factorizing	Solving	Total
2	0.1200E-01	0.2400E-01	0.3600E-01
4	0.2000E-01	0.2000E-01	0.4000E-01
8	0.2800E-01	0.4400E-01	0.7200E-01
16	0.4800E-01	0.8400E-01	0.1320E+00
32	0.9200E-01	0.9600E-01	0.1880E+00
64	0.1760E+00	0.1920E+00	0.3680E+00
128	0.3440E+00	0.3800E+00	0.7240E+00
256	0.6680E+00	0.7440E+00	0.1412E+01
512	0.1360E+01	0.1476E+01	0.2836E+01
1024	0.2648E+01	0.2884E+01	0.5532E+01

Table 12: Relative residual sizes, for beam with simply-supported ends

	Iteration number						
m	1	2	3	4	5	6	7
2	0.5076E-07	0.4138E-07	0.6825E-15	0.1913E-15	0.5263E-23	0.8843E-24	0.3413E-31
4	0.8287E-10	0.2094E-10	0.5314E-21	0.6000E-22	0.2523E-32		
8	0.1190E-12	0.5092E-14	0.2206E-27	0.5444E-29	0.1133E-33		
16	0.1153E-15	0.1512E-17	0.2875E-33	0.9051E-34	0.6140E-34		
32	0.1124E-18	0.4163E-20	0.1240E-33				
64	0.1097E-21	0.5265E-23	0.1185E-33				
128	0.1071E-24	0.5699E-26	0.1167E-33				
256	0.6699E-27	0.1489E-28	0.2177E-33				
512	0.8823E-26	0.3535E-28	0.2681E-33				
1024	0.1172E-25	0.5591E-28	0.2968E-33				

Table 13: Relative errors, for $J_{10}(x)$

m	$R(\phi)$	$R(\phi')$	$R(\phi'')$	$R(\phi^{(3)})$	$R(\phi^{(4)})$
16	0.2120E-14	0.1170E-14	0.2791E-14	0.1966E-13	0.8241E-12
32	0.1006E-20	0.2662E-20	0.1455E-20	0.2364E-18	0.1939E-16
64	0.1692E-26	0.8051E-25	0.5482E-24	0.4209E-21	0.6658E-19
128	0.2625E-27	0.1084E-27	0.1538E-26	0.2724E-23	0.8361E-21
256	0.6680E-27	0.3037E-28	0.2437E-28	0.9114E-25	0.5650E-22
512	0.5439E-27	0.2523E-28	0.2072E-28	0.5276E-26	0.6609E-23

Table 14: CPU times, for $J_{10}(x)$

m	Factoring	Solving	Total
16	0.3080E+00	0.1440E+00	0.4520E+00
32	0.5280E+00	0.1600E+00	0.6880E+00
64	0.1016E+01	0.2880E+00	0.1304E+01
128	0.1912E+01	0.5720E+00	0.2484E+01
256	0.3712E+01	0.1128E+01	0.4840E+01
512	0.7352E+01	0.2220E+01	0.9572E+01

Table 15: Relative residual sizes, for $J_{10}(x)$

	Iteration number						
m	1	2	3	4	5	6	7
16	0.3376E-10	0.1154E-11	0.2294E-12	0.4247E-13	0.7863E-14	0.1456E-14	0.2695E-15
32	0.2703E-16	0.7637E-18	0.6004E-19	0.6428E-20	0.6883E-21	0.7370E-22	0.7892E-23
64	0.1360E-19	0.1540E-20	0.1350E-21	0.1193E-22	0.1053E-23	0.9270E-25	0.7836E-26
128	0.2532E-22	0.3730E-23	0.3045E-24	0.2440E-25	0.2089E-26	0.6049E-27	
256	0.2122E-24	0.1838E-25	0.1325E-26	0.5645E-27	0.4061E-27		
512	0.4847E-25	0.2117E-27					

	Iteration number						
m	8	9	10	11	12	13	14
16	0.4991E-16	0.9240E-17	0.1711E-17	0.3168E-18	0.5865E-19	0.1086E-19	0.2010E-20
32	0.8431E-24	0.9132E-25	0.9707E-26	0.1525E-26	0.6545E-28		
64	0.3637E-27	0.5771E-28					

	Iteration number						
m	15	16	17	18	19	20	21
16	0.3722E-21	0.6893E-22	0.1275E-22	0.2372E-23	0.4270E-24	0.8963E-25	0.6186E-26