A NEW FINITE DIFFERENCE METHOD FOR THE HELMHOLTZ EQUATION USING SYMBOLIC COMPUTATION

LARRY A. LAMBE, RICHARD LUCZAK, AND JOHN W. NEHRBASS

Abstract. A new finite difference method for the Helmholtz equation is presented. The method involves replacing the standard "weights" in the central difference quotients (Sects. 2.1, 2.2, and 2.3) by weights that are *optimal* in a sense that will be explained in the Sects. just mentioned. The calculation of the optimal weights involves some complicated and error prone manipulations of integral formulas that is best done using computer aided symbolic computation (SC). In addition, we discuss the important problem of *interpolation* involving meshes that have been refined in certain subregions. Analytic formulae are derived using SC for these interpolation schemes. Our results are discussed in Sect. 5. Some hints about the computer methods we used to accomplish these results are given in the Appendix. More information is available and access to that information is referenced.

While we do not want to make SC the focus of this work, we also do not want to underestimate its value. Armed with robust and efficient SC libraries, a researcher can *comfortably* and *conveniently* experiment with ideas that he or she might not examine otherwise.

1. Introduction

A standard computational tool for approximating solutions to systems of partial differential equations with boundary conditions is the *finite difference method* [1] [2] [3]. In the case of the Helmholtz equation

$$\nabla^2 u = -\kappa^2 u, \tag{1}$$

it was shown in Chapt. four of [4] that numerical errors that can occur in the central difference quotients could be corrected *without* increasing the order by optimally adjusting "weights" for these quotients. This material will be reviewed in Sect. 2. The problem of accurately interpolating when a mesh is refined in some subregion of interest for a given initial mesh for the Helmholtz equation was also addressed in Chapt. five of [4] and this will be reviewed in Sect. 4. The calculations necessary for this involved working out some rather complicated integral formulas and manipulating complex algebraic expressions, all of which can be tedious and error prone. In this paper, it will be shown how SC can be used to relieve the tedium and eliminate the inevitable typographical errors involved in hand calculations.

2. Optimizing Weights in a Finite Difference Method for the Helmholtz Equation

2.1. **Dimension one.** Consider a function u of one variable. Classical finite difference schemes are derived under the assumption that u can be expanded in terms

of a Taylor series. One then has

$$\begin{aligned} u(x+h) &= u(x) + u'(x)\frac{h}{1!} + u''(x)\frac{h^2}{2!} + \dots, \\ u(x-h) &= u(x) - u'(x)\frac{h}{1!} + u''(x)\frac{h^2}{2!} - \dots \end{aligned} \tag{2}$$

from which the approximations

$$\begin{aligned} u'(x) &\approx \frac{u(x+h) - u(x-h)}{2h}, \\ u''(x) &\approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \end{aligned} \tag{3}$$

are easily derived. The idea in [4] is to replace the coefficient two in the approximation above by a new coefficient ω which minimizes

$$\left| u'' - \frac{u(x+h) - \omega u(x) + u(x-h)}{h^2} \right|. \tag{4}$$

To derive this new weight, consider $u(x+h) + u(x-h)$ in general. We have

$$u(x+h) + u(x-h) = 2\left(u(x) + u^{(2)}(x)\frac{h^2}{2!} + u^{(4)}(x)\frac{h^4}{4!} + \dots \right) \tag{5}$$

from equations (2) above. By iterating the relation $u'' = -\kappa^2 u$, it follows that

$$u^{(2n)} = (-1)^n \kappa^{2n} u. \tag{6}$$

Substituting these values into Eq. (5) gives

$$u(x+h) + u(x-h) = 2\cos(\kappa h)u. \tag{7}$$

Thus, the equation

$$-\kappa^2 u = u''(x) = \frac{u(x+h) - \omega u(x) + u(x-h)}{h^2} \tag{8}$$

has an *exact* solution in this case, viz.

$$\omega = 2\cos(\kappa h) + (\kappa h)^2. \tag{9}$$

and we have an "adjusted weight" for a new finite difference scheme.

Remark 2.1. *In this case, it is actually well known that the exact solution to the one dimensional Helmholtz equation is given by* $\alpha e^{kxj} + \beta e^{-kxj}$ *(*$j^2 = -1$*) where the constants* α *and* β *are determined by the boundary values and the adjusted weight above can be directly calculated from this as is done in* [4].

2.2. **Dimension two.** Eq. (1) in dimension two is

$$u_{xx} + u_{yy} = -\kappa^2 u(x, y). \tag{10}$$

The classic central difference scheme gives the approximation

$$u_{xx} + u_{yy} \approx \frac{u(x+h, y) + u(x-h, y) - 4u(x, y) + u(x, y+h) + u(x, y-h)}{h^2} \tag{11}$$

and we want to replace the coefficient four above by an optimal (in a sense to be made precise) weight ω. Unfortunately, the method of the last section does not directly generalize due to the existence of cross derivative terms. However, an argument was given in [4] which gives a satisfactory answer that is optimal in the sense that we recall here.

Consider Eq. (10) in the absence of boundaries. It is known that the plane waves

$$f_\theta(x,y) = e^{j\kappa(x\cos(\theta)+y\sin(\theta))} \tag{12}$$

are solutions ($j^2=-1$). A straightforward calculation gives

$$\begin{aligned} f_\theta(x+h,y)+f_\theta(x-h,y)+f_\theta(x,y+h)+f_\theta(y,x-h) = \\ 2(\cos(\kappa\ h\cos(\theta))+\cos(\kappa\ h\sin(\theta)))f_\theta(x,y). \end{aligned} \tag{13}$$

Thus, in this case, we would like to find ω such that

$$\frac{2(\cos(\kappa h\cos(\theta))+\cos(\kappa h\sin(\theta)))f_\theta-\omega\ f_\theta}{h^2} \tag{14}$$

is as close to $-\kappa^2 f_\theta$ as possible. Equivalently, we want to find an optimal ω such that

$$2(\cos(\kappa h\cos(\theta))+\cos(\kappa h\sin(\theta)))-(\omega-\kappa^2h^2)\approx 0. \tag{15}$$

Eq. (15) does not have a solution that is independent of the angle θ, however, it is reasonable to try to minimize the average over all angles and hope that there is a unique solution. In other words, we seek a solution ω to the equation

$$\int_0^{2\pi}(2(\cos(\kappa h\ \cos(\theta))+\cos(\kappa h\sin(\theta)))-(\omega-\kappa^2h^2))d\theta = 0. \tag{16}$$

Equivalently,

$$2\pi\omega = 2\int_0^{2\pi}(\cos(\kappa h\cos(\theta))+\cos(\kappa h\sin(\theta)))d\theta+2\pi\kappa^2h^2, \tag{17}$$

and there is a unique solution

$$\omega = \frac{1}{\pi}\int_0^{2\pi}(\cos(\kappa h\cos(\theta))+\cos(\kappa h\sin(\theta)))d\theta+\kappa^2h^2. \tag{18}$$

In fact, there is an analytic expression for this integral in terms of the Bessel function of the first kind. We have

$$\int_0^{2\pi}(\cos(\kappa h\cos(\theta))+\cos(\kappa h\sin(\theta)))d\theta = 4\pi J_0(\kappa h). \tag{19}$$

This follows from the classic formula [5] (where $z=x+yj$)

$$J_0(z) = \frac{1}{\pi}\int_0^{\pi}\cos(z\sin(\theta))d\theta = \frac{1}{\pi}\int_0^{\pi}\cos(z\cos(\theta))d\theta \tag{20}$$

and the easily proven formulas

$$\begin{aligned} \int_0^{\pi}\cos(z\sin(\theta))d\theta &= \int_{\pi}^{2\pi}\cos(z\sin(\theta))d\theta, \\ \int_0^{\pi}\cos(z\cos(\theta))d\theta &= \int_{\pi}^{2\pi}\cos(z\cos(\theta))d\theta. \end{aligned} \tag{21}$$

Thus, in dimension two, we have the adjusted weight

$$\omega = 4J_0(\kappa h)+(\kappa h)^2. \tag{22}$$

Note that, in this case, the integral involved is easy enough that there is little trouble in getting the required expressions by hand. We will nonetheless note that

the problem can be phrased in terms of "rewrite rule" transformations suitable for machine calculation, viz.

$$\begin{aligned}\int_0^{2\pi} \cos(z\sin(x))dx &\longmapsto \int_0^{2\pi} \cos(z\cos(x))dx,\\ \int_0^{2\pi} \cos(z\cos(x))dx &\longmapsto 2\pi J_0(x).\end{aligned} \tag{23}$$

It is then a simple matter of applying the first rule to get

$$\int_0^{2\pi} (\cos(\kappa h\cos(x)) + \cos(\kappa h\sin(x))dx \longmapsto 2\int_0^{2\pi} \cos(\kappa h\cos(x))dx \tag{24}$$

and then applying the second rule,

$$2\int_0^{2\pi} \cos(\kappa h\cos(x))dx \longmapsto 4\pi J_0(x). \tag{25}$$

Most symbolic computational systems allow the user to define such transformations and it is easy to see how a program might be written to automate the process.

2.3. **Dimension three.** In dimension three, Eq. (1) becomes

$$u_{xx} + u_{yy} + u_{zz} = -\kappa^2 u(x,y,z) \tag{26}$$

and the standard central difference approximation for the left hand side of (26) is

$$\begin{aligned}&\frac{u(x-h,y,z)+u(x+h,y,z)+u(x,y-h,z)+u(x,y+h,z)}{h^2}\\ &\qquad+\frac{u(x,y,z-h)+u(x,y,z+h)-6u(x,y,z)}{h^2}\end{aligned} \tag{27}$$

and we want to replace the coefficient six in u by an optimal ω. The method used in dimension two generalizes in a straightforward way using the plane waves

$$f_{\theta,\phi}(x,y,z) = e^{j\kappa(\sin(\theta)\cos(\phi)x+\sin(\theta)\sin(\phi)y+\cos(\theta)z)}. \tag{28}$$

Substituting $u = f_{\theta,\phi}$ into (27), replacing the six by ω, and letting

$$\varphi(\theta,\phi) = \cos(\kappa h\sin(\theta)\cos(\phi)) + \cos(\kappa h\sin(\theta)\sin(\phi)) + \cos(\kappa h\cos(\theta)), \tag{29}$$

we get

$$\frac{2\varphi(\theta,\phi)f_{\theta,\phi} - \omega f_{\theta,\phi}}{h^2} \tag{30}$$

which we want to be as close as possible to $-\kappa^2 f_{\theta,\phi}$, i.e. we want

$$2\varphi(\theta,\phi) + h^2\kappa^2 - \omega \tag{31}$$

as close to zero as possible. Again, we cannot solve this uniformly for all θ and φ, but we can try to minimize the average by integrating (31) over an arbitrary sphere of non-zero radius (equivalently, in this case, over a sphere of radius one) and setting that equal to zero to get

$$\int_0^{2\pi}\left(\int_0^{\pi} \omega\sin(\theta)d(\theta)\right)d(\phi) = \int_0^{2\pi}\left(\int_0^{\pi} (2\varphi(\theta,\phi) + (\kappa h)^2)\sin(\theta)d(\theta)\right)d(\phi), \tag{32}$$

i.e.

$$\omega = \frac{1}{2\pi}\int_0^{2\pi}\left(\int_0^{\pi}(\varphi(\theta,\phi)+\frac{(\kappa h)^2}{2})\sin(\theta)d(\theta)\right)d(\phi), \tag{33}$$

Using (29), we have $\omega = p_1 + p_2 + p_3 + p_4$ where

$$\begin{aligned}
p_1 &= \frac{1}{2\pi}\int_0^{2\pi}\left(\int_0^{\pi}\cos(\kappa h\sin(\theta)\cos(\varphi))\sin(\theta)d(\theta)\right)d(\phi),\\
p_2 &= \frac{1}{2\pi}\int_0^{2\pi}\left(\int_0^{\pi}\cos(\kappa h\sin(\theta)\sin(\varphi))\sin(\theta)d(\theta)\right)d(\phi),\\
p_3 &= \frac{1}{2\pi}\int_0^{2\pi}\left(\int_0^{\pi}\cos(\kappa h\cos(\theta))\sin(\theta)d(\theta)\right)d(\phi),\\
p_4 &= \frac{1}{2\pi}\int_0^{2\pi}\left(\int_0^{\pi}\frac{(\kappa h)^2}{2}\sin(\theta)d(\theta)\right)d(\phi).
\end{aligned} \tag{34}$$

In this case, hand derivation of analytical expressions for these integrals is extremely tedious and "typo prone" so we want to, at the very least, check any hand calculations using symbolic computation on a computer.

3. Symbolic Computation

While most computer algebra systems such as Macsyma [6], REDUCE [7], Maple [8], Mathematica [9], and AXIOM [10], etc., include symbolic integration routines, they are not set up to handle the kinds of integrations above in general.

As already mentioned in the paragraph after Eq. (22), one possible way to proceed is to implement a customized set of *rewrite rules* [11] [12] [13] based on tables such as those given in [5] [14] [15]. There can be some problems with such an approach. Applying rewrite rule transformations in a meaningful and productive way can be challenging or even impossible (see the references just cited). Often, any attempts to try to automate such transformations can be doomed to endlessly cycle into each other without yielding any mathematical insight. For that reason, we need a system that gives very tight control and possible user interaction with rewrite rule transformations. In fact, it is certainly desirable to have total control of what one's code is doing when executing a rewrite rule transformation and to have complete knowledge of what that transformation actually accomplishes. The methods we employ allow us to accomplish these goals.

3.1. **Needed functionality.**

The kinds of symbolic computational facilities needed for this work are:

- Symbolic differentiation,
- the ability to create user defined operators,
- the ability to define the behavior of user defined operators, e.g.
  - the specification of the partial derivatives of a given operator in the system,
  - the ability to specify (bi)linearity of an operator with real or complex coefficients, etc.,
- the ability to transform subexpressions of a given expression in a very controlled, well-defined manner.

3.2. **ExprLib.** While it is possible to implement your own rewrite rule systems in any of the systems mentioned, we have chosen another system, ExprLib [16] [17], for several reasons:

- The library gives very precise user control over the *parse tree* of a given expression,
- it gives very precise user control of the way various subexpressions may be transformed in a given expression,
- it is extremely efficient as compared to other available systems and this becomes very important when we calculate precise interpolation formulas in Sect. 4.
- it is a portable ANSI C library which easily interfaces with other code as any C library does.

The ExprLib library has many facilities often found in symbolic computation systems (often called "computer algebra" systems) such as the ones mentioned above. There are some important differences. As an ANSI C library, it is compiled rather than interpreted. While most symbolic computation systems stress "exact" computations (arbitrary integer or rational constants) and arbitrary precision floating point constants, ExprLib's default is the C type `double`. We have found this quite sufficient for all of the work we describe.

We wrote an ExprLib program to compute the adjusted weights for the new finite difference method just described. Some details are in the appendix of this paper and more details can be found at [18]. Here we will just summarize the results for the Helmholtz equation in Sect. 2, we have

$$\begin{aligned} \omega &= 2\cos(\kappa h) + (\kappa h)^2 \quad \text{in dimension one,} \\ \omega &= 4J_0(\kappa h) + (\kappa h)^2 \quad \text{in dimension two,} \\ \omega &= 6j_0(\kappa h) + (\kappa h)^2 \quad \text{in dimension three.} \end{aligned} \tag{35}$$

4. Interpolation

In applying such a modified finite difference method as above, one establishes an error criteria. Then a mesh is chosen with fine enough resolution to achieve the desired accuracy. But there are several reasons why it may be desirable for the mesh to be reduced even farther than required for a given accuracy. For example, when dense materials are examined, the wave length inside the materials decreases, and so a finer mesh is required in order to achieve the desired accuracy. Another reason may be due to the physical geometry itself. The geometry in question may include a smooth surface in which the radius of curvature may be too small to be sufficiently resolved with the original mesh. Yet another justification occurs when small objects must be resolved. Objects that are smaller than the size of a single cell are poorly approximated.

The problem with using a finer mesh, is that it requires larger memory allocations and longer computation times. It would be preferable if a coarse mesh could be used over the majority of the computational domain, while a finer mesh is used only around the areas required. Traditionally, this has been accomplished by determining interpolate information between meshes as follows. If a field value was required in a location that was not sampled, an average value of the surrounding fields would be used. Unfortunately this technique causes large artificial reflections, and thus severely contaminates the solution. The reason that these interpolating schemes fail

is that there is no physical justification for the interpolating scheme used. Averages or centroids are often used in order to interpolate the field information yielding results that are often poor at best.

A systematic method will be presented in order to optimally choose the weights required in an interpolation scheme. For the purpose of this explanation, the following meshing scheme will be adopted. First, a coarse mesh is established and placed throughout the computational domain. Next, regions where the resolutions are to be increased are identified. In these sensitive regions the resolutions are doubled. Therefore, every coarse cell in this region is now divided into four finer cells. When adjacent cells do not have the same spatial resolution, the region will be referred to as a transition region. If still finer resolutions are required, the process is again repeated. By adopting this meshing scheme, all transition regions will contain adjacent cells with a 2:1 ratio of resolutions. As illustrated in Fig. 1 below:

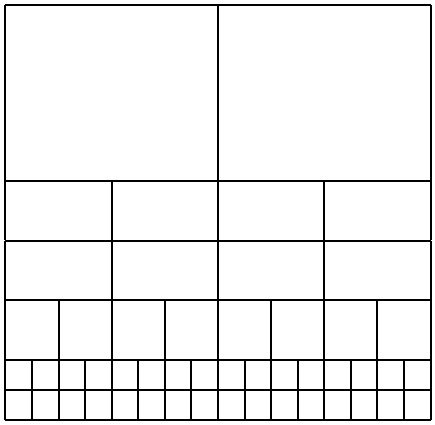

FIGURE 1. mesh refinement

In adopting this convention, interpolation takes place for three different stencil patterns. These stencils are referred to as the middle stencil (See Fig. 2), the cross stencil (See Fig. 3), and the corner stencil (See Fig. 4).

4.1. **Middle interpolation.** In Fig. 2, the shorter distances between the nodes shown is h and the two longer distances are $2h$. The coordinates of node n_0 are assumed to be $(0,0)$. The labeling of nodes starts at $(0,0)$ and proceeds clockwise. So, e.g. $n_7 = (h, 2h)$. In deriving this interpolation scheme, it is assumed that the region surrounding the stencil is homogeneous. First, a plane wave which propagates at an arbitrary direction, exists in the homogeneous region and is defined as $f = e^{\kappa j(x\cos(\phi)+y\sin(\phi))}$.

When the distance between the nodes are defined as in Fig. 2, the value of the plane wave sampled at the nodal locations shown is given by

$$\begin{aligned} f_0 &= f(n_0) = 1,\ f_1 = f(n_1) = e^{j\kappa h\cos(\phi)},\ f_2 = f(n_2) = e^{j\kappa h(\cos(\phi)-\sin(\phi))}, \\ f_3 &= f(n_3) = e^{-j\kappa h\sin(\phi)},\ f_4 = f(n_4) = e^{-j\kappa h(\cos(\phi)+\sin(\phi))}, \\ f_5 &= f(n_5) = e^{-j\kappa h\cos(\phi)},\ f_6 = f(n_6) = e^{j\kappa h(-\cos(\phi)+2\sin(\phi))}, \\ f_7 &= f(n_7) = e^{j\kappa h(\cos(\phi)+2\sin(\phi))}. \end{aligned} \tag{36}$$

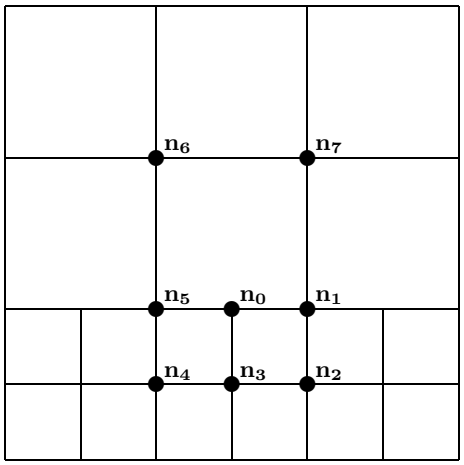

FIGURE 2. the middle stencil

It is now desired to approximate the field in the center of the stencil f_0 as a function of the neighboring nodes

$$\frac{w_0 f_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 + w_5 f_5 + w_6 f_6 + w_7 f_7}{h^2} = Res(\phi) \approx 0. \quad (37)$$

This equation could be solved for a given propagation angle. However, for arbitrary scatters, there is no way of knowing which angle we would need to solve for. Assuming that all angles are equally probable, we solve the approximation in an average sense

$$\int_0^{2\pi} h^2 Res(\phi) d\phi = 0. \quad (38)$$

Performing the integration and solving for w_0 yields

$$w_0 = -\Big[(w_1 + w_3 + w_5)J_0(\kappa h) + (w_2 + w_4)J_0(\sqrt{2}\kappa h) + (w_6 + w_7)J_0(\sqrt{5}\kappa h)\Big]. \quad (39)$$

An optimal value for w_0 is known as a function of the other weights. Before solving for the other weights we reduce the number of unknowns further by exploiting the symmetry of the problem. The relations ($w_1 = w_5$, $w_2 = w_4$, $w_6 = w_7$) must hold true in order that the wave does not have a preferred propagation direction. At this point we are allowed to pick one of the weights so that the others will be uniquely determined. We choose $w_1 = w_5 = 1$ so that our approximating equation becomes

$$w_0 + f_1 + (f_2 + f_4)w_2 + f_3 w_3 + f_5 + (f_6 + f_7)w_7 = 0. \quad (40)$$

Now substituting the value of w_0 and f_i, $i = 1, \ldots, 7$ into (40), we obtain

$$R = F_1 W_1 + F_2 W_2 + F_3 W_3 \quad (41)$$

where

$$\begin{aligned} W_1 &= w_2,\ W_2 = w_3,\ W_3 = w_6, \\ R &= 2J_0(\kappa h) - e^{-\kappa hj\cos(\phi)} - e^{-\kappa hj\cos(\phi)}, \\ F_1 &= e^{\kappa hj(\cos(\phi)-\sin(\phi))} + e^{-\kappa hj(\cos(\phi)+\sin(\phi))} - 2J_0(\sqrt{2}\kappa h), \\ F_2 &= e^{-\kappa hj\sin(\phi)} - J_0(\kappa h), \\ F_3 &= e^{\kappa hj(-\cos(\phi)+2\sin(\phi))} + e^{\kappa hj(\cos(\phi)+2\sin(\phi))} + 2J_0(\sqrt{5}\kappa h). \end{aligned} \quad (42)$$

We have thus reduced the problem to optimally approximating Eq. (41) for the unknown quantities W_1, W_2, and W_3, given R, F_1, F_2, and F_3. Before solving this problem, we will show that the other two cases also lead to this sort of problem as well.

4.2. **Cross interpolation.** In Fig. 3, the shorter distances between the nodes shown is h and the two longer distances are $2h$. The coordinates of node n_0 are assumed to be $(0,0)$. The labeling of nodes starts at $(0,0)$ and proceeds clockwise. So, e.g. $n_6 = (0, 2h)$.

When the distance between the nodes are defined as in Fig. 3, the values of the plane wave sampled at the nodal locations shown are given by

$$\begin{aligned} f_0 &= f(n_0) = 1,\ f_1 = f(n_1) = e^{j\kappa h\cos(\phi)},\ f_2 = f(n_2) = e^{j\kappa h(\cos(\phi)-\sin(\phi))} \\ f_3 &= f(n_3) = e^{-j\kappa h\sin(\phi)},\ f_4 = f(n_4) = e^{-j\kappa h(\cos(\phi)+\sin(\phi))}, \\ f_5 &= f(n_5) = e^{-j\kappa h\cos(\phi)}\ f_6 = f(n_6) = e^{j2\kappa h\sin(\phi)}. \end{aligned} \quad (43)$$

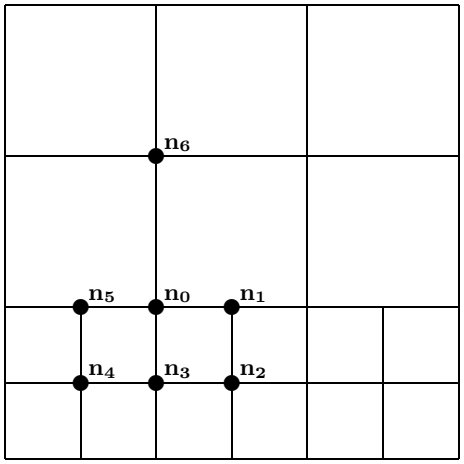

FIGURE 3. the cross stencil

Once again, we want to approximate the field in the center of the stencil f_0 as a function of the neighboring nodes

$$\frac{w_0f_0 + w_1f_1 + w_2f_2 + w_3f_3 + w_4f_4 + w_5f_5 + w_6f_6 + w_7f_7}{h^2} = Res(\phi) \approx 0 \quad (44)$$

Proceeding as in the last Sect., we integrate over a 2π interval and solve for w_0 as

$$w_0 = -\Big[2(1+w_3)J_0(2\kappa h) + 2w_2J_0(\kappa h) + w_4J_0(\sqrt{2}\kappa h)\Big] \quad (45)$$

where we have made use of the symmetry so that $w_4 = w_2$ and $w_5 = w_1$. We let $w_1 = 1$, so that our approximating equation becomes

$$w_0 + f_1 + (f_2 + f_4)w_2 + f_3 w_3 + f_5 + f_6 w_6 = 0. \tag{46}$$

By substituting for the known values, we again obtain the equation

$$R = F_1 W_1 + F_2 W_2 + F_2 W_2 \tag{47}$$

where

$$\begin{aligned} W_1 &= w_2,\ W_2 = w_3,\ W_3 = w_6 \\ R &= 3J_0(\kappa h) - e^{\kappa h j \cos(\phi)} - e^{-\kappa h j \cos(\phi)} \\ F_1 &= 2J_0(\sqrt{2}\kappa h) + e^{\kappa h j(\cos(\phi)-\sin(\phi))} + e^{-\kappa h j(\cos(\phi)+\sin(\phi))} \\ F_2 &= e^{-\kappa h j \sin(\phi)} - J_0(\kappa h) \\ F_3 &= e^{2\kappa h j \sin(\phi)} - J_0(2\kappa h). \end{aligned} \tag{48}$$

4.3. **Corner interpolation.** In Fig. 4, the shorter distances between the nodes shown is h and the two longer distances are $2h$. The coordinates of node n_0 are assumed to be $(0, 0)$. The labeling of nodes starts at $(0, 0)$ and proceeds clockwise (as indicated). So, e.g. $n_7 = (0, 2h)$.

When the distance between the nodes are defined as in Fig. 4, the values of the plane wave sampled at the nodal locations shown are given by

$$\begin{aligned} f_0 &= 1,\ f_1 = e^{j2\kappa h \cos(\phi)},\ f_2 = e^{-j\kappa h \sin(\phi)}, \\ f_3 &= e^{-j2\kappa h \sin(\phi)},\ f_4 = e^{-j\kappa h(\cos(\phi)+\sin(\phi))},\ f_5 = e^{-j\kappa h \cos(\phi)}, \\ f_6 &= e^{-j2\kappa h \cos(\phi)},\ f_7 = e^{j2\kappa h \sin(\phi)}. \end{aligned} \tag{49}$$

Once again, it is now desired to approximate the field in the center of the stencil f_0 as a function of the neighboring nodes

$$\frac{w_0 f_0 + w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 + w_5 f_5 + w_6 f_6 + w_7 f_7}{h^2} = Res(\phi) \approx 0. \tag{50}$$

Again we integrate over a 2π interval and solve for w_0 to obtain

$$w_0 = -\left[2(1 + w_3)J_0(2\kappa h) + 2w_2 J_0(\kappa h) + w_4 J_0(\sqrt{2}\kappa h)\right] \tag{51}$$

where we have made use of the symmetry and let $w_1 = w_7 = 1$, $w_2 = w_5$, and $w_3 = w_6$. Again, the simplified equation becomes

$$W_1 F_1 + W_2 F_2 + W_3 F_3 = R \tag{52}$$

where

$$\begin{aligned} W_1 &= w_2,\ W_2 = w_3,\ W_3 = w_4 \\ R &= 2J_0(2\kappa h) - e^{2\kappa h j \cos(\phi)} - e^{2\kappa h j \sin(\phi)} \\ F_1 &= -2J_0(\kappa h) + e^{-\kappa h j \sin(\phi)} + e^{-\kappa h j \cos(\phi)} \\ F_2 &= -2J_0(2\kappa h) + e^{-2\kappa h j \sin(\phi)} + e^{2\kappa h j \cos(\phi)} \\ F_3 &= -J_0(\sqrt{2}\kappa h) + e^{-\kappa h j(\cos(\phi)+\sin(\phi)}. \end{aligned} \tag{53}$$

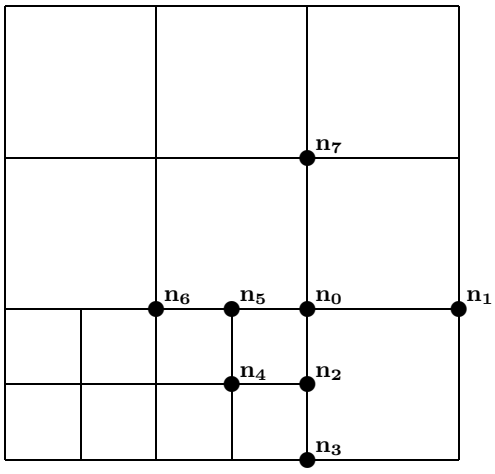

FIGURE 4. the corner stencil

4.4. **Summary.** In each case above, we have reduced the problem of finding an optimal interpolation scheme to the following mathematical problem.

Problem 4.1. *Given functions F_1, F_2, F_3, and R, find the coefficients W_1, W_2, and W_3 which gives the best approximation*

$$W_1F_1 + W_1F_2 + W_1F_3 \approx R. \tag{54}$$

In our case, the given functions are complex valued and integrable (in fact, analytic) over the interval $[0, 2\pi]$. Furthermore, they are linearly independent in the complex vector space of such functions taking $[0, 2\pi]$ to $\mathbb{C}$. The complex bilinear form given by

$$< f, g >= \int_0^{2\pi} f(\theta)\overline{g(\theta)}d\theta \tag{55}$$

allows us to apply Gram-Schmidt orthogonalization [19] to solve this problem. The idea is to apply the process to the linearly independent set $\{F_1, F_2, F_3\}$ to obtain an orthonormal set $\{P_1, P_2, P_3\}$ for which we know that

$$\sum_{i=1}^{3} \frac{< R, P_i >}{< P_i, P_i >} \tag{56}$$

is a best approximation (Theorem 4, p. 284 of [19]) to R. Since the Gram-Schmidt process is defined by the recursive sequence

$$\begin{aligned} P_1 &= F_1, \\ P_{k+1} &= F_{k+1} - \sum_{i=1}^{k} \frac{< F_{k+1}, F_i >}{< F_i, F_i >} P_i, \end{aligned} \tag{57}$$

it is clear that the P_i are linear combinations of the F_i and thus there are unique W_i such that

$$R \approx \sum_{i=1}^{3} \frac{< R, P_i >}{< P_i, P_i >} = \sum_{i=1}^{3} W_i F_i. \tag{58}$$

In general, this solution is complicated in two ways. The algebra needed to express the P_i as linear combinations of the F_j is complex and error prone if attempted "by hand". Secondly, in general, there may not be a "closed form" for the integrals involved in calculating the inner products. In our case, the integrals involved actually have analytic expressions and one case is calculated in the next Sect.

4.5. **Calculating inner products.** Consider the norm of $f = F_1$ in the corner stencil case (53). We have that

$$f(\phi) = e^{-\kappa h \sin(\phi) j} + e^{-\kappa h \cos(\phi) j} - 2J_0(\kappa h) \tag{59}$$

and so

$$\overline{f(\phi)} = e^{\kappa h \sin(\phi) j} + e^{\kappa h \cos(\phi) j} - 2J_0(\kappa h). \tag{60}$$

Substituting $f(\phi)$ and $\overline{f(\phi)}$ into the inner product formula yields

$$\begin{aligned} <f, f> \;=\; & -2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \sin(\phi) j} d\phi + \int_0^{2\pi} e^{\kappa h (\sin(\phi) - \cos(\phi)) j} d\phi - \\ & 2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \cos(\phi) j} d\phi + \int_0^{2\pi} e^{\kappa h (\cos(\phi) - \sin(\phi)) j} d\phi - \\ & 2J_0(\kappa h) \int_0^{2\pi} e^{-\kappa h \cos(\phi) j} d\phi - 2J_0(\kappa h) \int_0^{2\pi} e^{-\kappa h \sin(\phi) j} d\phi + \\ & (4J_0^2(\kappa h) + 2) \int_0^{2\pi} d\phi \end{aligned} \tag{61}$$

Now applying the formula

$$\begin{aligned} A\cos(\phi) + B\sin(\phi) \;=\; & \sqrt{A^2 + B^2}\cos(\phi - \alpha) \\ & \alpha = \arctan(B/A) \end{aligned} \tag{62}$$

to each integrand on the right hand side of the inner product yields

$$\begin{aligned} <f, f> \;=\; & -2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \cos(\phi - \alpha_1) j} d\phi + \int_0^{2\pi} e^{\sqrt{2}\kappa h \cos(\phi - \alpha_2) j} d\phi - \\ & 2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \cos(\phi) j} d\phi + \int_0^{2\pi} e^{\sqrt{2}\kappa h \cos(\phi - \alpha_3) j} d\phi - \\ & 2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \cos(\phi - \alpha_4) j} d\phi - 2J_0(\kappa h) \int_0^{2\pi} e^{\kappa h \cos(\phi - \alpha_5) j} d\phi + \\ & (4J_0^2(\kappa h) + 2) \int_0^{2\pi} d\phi \end{aligned} \tag{63}$$

where $\alpha_1 = \frac{\pi}{2}, \alpha_2 = \frac{3\pi}{4}, \alpha_3 = -\frac{\pi}{4}, \alpha_4 = -\pi, \alpha_5 = -\frac{\pi}{2}$. Substituting

$$\int_0^{2\pi} e^{x \cos(\phi + \alpha) j} d\phi = 2\pi J_0(x) \tag{64}$$

into the right hand side of the inner product yields

$$\begin{aligned} <f, f> \;=\; & -2J_0(\kappa h)(2\pi J_0(\kappa h)) + (2\pi J_0(\sqrt{2}\kappa h)) - 2J_0(\kappa h)(2\pi J_0(\kappa h)) + \\ & (2\pi J_0(\sqrt{2}\kappa h)) - 2J_0(\kappa h)(2\pi J_0(\kappa h)) - 2J_0(\kappa h)(2\pi J_0(\kappa h)) + \\ & (4J_0^2(\kappa h) + 2)(2\pi) \\ =\; & 4\pi(J_0(\sqrt{2}\kappa h) - 2J_0^2(\kappa h) + 1) \end{aligned} \tag{65}$$

Obviously, the solution of this specific example involves quite complex transformations due to operations with exponential functions and complexity of the inner product. Although the inner product was derived for relatively simple functions in this case, the hand derived calculations are error prone and quite tedious.

5. Results

5.1. **Dimension one.** The selection of the new weights in place of the classical weights has a tremendous effect on the corresponding scheme as we will now show in dimension one.

Consider the boundary value problem where $u(0) = 1$ and $u(1) = \cos(\kappa)$ over the unit interval $[0, 1]$. The exact solution is $u(x) = \cos(\kappa x)$. Using the partition $0 = x_0, \ldots, x_i = \frac{i}{h}, \ldots, x_n = 1$ for some $0 < h < 1$, the discretization is $u_i \approx u(x_i)$ where

$$\frac{u_{i+1} - wu_i + u_{i-1}}{h^2} + \kappa^2 u_i = 0. \tag{66}$$

Thus one obtains a tridiagonal system where the first equation is $cu_1 + u_2 = -1$, the j^{th} equation for $2 \leq j \leq n-2$ is $u_{j-1} + cu_j + u_{j+1} = 0$, and the $(n-1)^{th}$ equation is $u_{n-2} + cu_{n-1} = -\cos(\kappa)$ where $c = -2\cos(\kappa h)$. Here is a comparison for the case $\kappa = 10$ and $h = \frac{1}{6}$. A precision of 50 digits was used. The approximation using the classic weight 2 is given by $\overline{u_i}$, the approximation using the adjusted weight $2\cos(\kappa h) + \kappa^2 h^2$ is given by $\widetilde{u_i}$ and the exact value is given by u_i below for $i = 1, \ldots, 5$.

$$\begin{aligned}
\overline{u_1} &= 1.7500767378597993568997067667310199837437259830851 5\\
\widetilde{u_1} &= -0.0957235480143755841156138368653112310075927500671 0\\
u_1 &= -0.0957235480143755841156138368653112310075927500670 9\\
\overline{u_2} &= -2.3611707961131772775886608185685710984673424312884 5\\
\widetilde{u_2} &= -0.9816740047110790643351106905121066656544109963985 9\\
u_2 &= -0.9816740047110790643351106905121066656544109963985 9\\
\overline{u_3} &= 0.0863894368948940812248072032667575372864292412503 0\\
\widetilde{u_3} &= 0.2836621854632262644666391715135573083344225922522 1\\
u_3 &= 0.2836621854632262644666391715135573083344225922522 1\\
\overline{u_4} &= 2.2939790118615929921915885493610930139112307992048 8\\
\widetilde{u_4} &= 0.9273677030509753619969537088242912205997879961027 3\\
u_4 &= 0.9273677030509753619969537088242912205997879961027 3\\
\overline{u_5} &= -1.8705953350094664084849316305476076592173865295207 7\\
\widetilde{u_5} &= -0.4612040391631887423311305017804203760785369737289 8\\
u_5 &= -0.4612040391631887423311305017804203760785369737289 7
\end{aligned}$$

5.2. **Dimension two.** A simple two dimensional experiment is performed in order to demonstrate how the modified central difference approximation reduces the numerical dispersion errors and thus contributes to reducing the total errors for a wave propagation problem. In this experiment, a plane wave is modeled as it propagates over a region of free space. As the plane wave solution is known in closed form, the exact boundary conditions can be placed on the outside edges of our selected domain. The Helmholtz's equation is then used to model the transverse electric fields

at each grid location interior to the domain. Thus all grid points have a unknown associated with them. The same stencil as described earlier in this paper is used.

For each interior grid point a linear equation is formed and thus the system can be solved as a set of linear equations with boundary conditions. Finally, the error can be calculated at each grid location as the exact analytical solution is know for all space. This experiment looks at the error as a function of grid spacing, and wave propagation angle for both the traditional central difference approximation (referred to as the old method) and the modified approximation (refereed to as the new method). Note that the only difference between these two methods is the value of the weight given to the center grid in the stencil. Therefore, computation times are exactly the same.

The errors are calculated in this paper as follows: for all grid points the complex difference between the calculated value and the analytical value is computed. Each difference is multiplied by its complex conjugate and then all resultant values are summed. The sum is then divided by the total number of grid points used in the sum and thus the average root mean squared values is descriptive of the global error. The maximum errors were also calculated for each experiment, however since the trends were the same, we chose to present only the above errors.

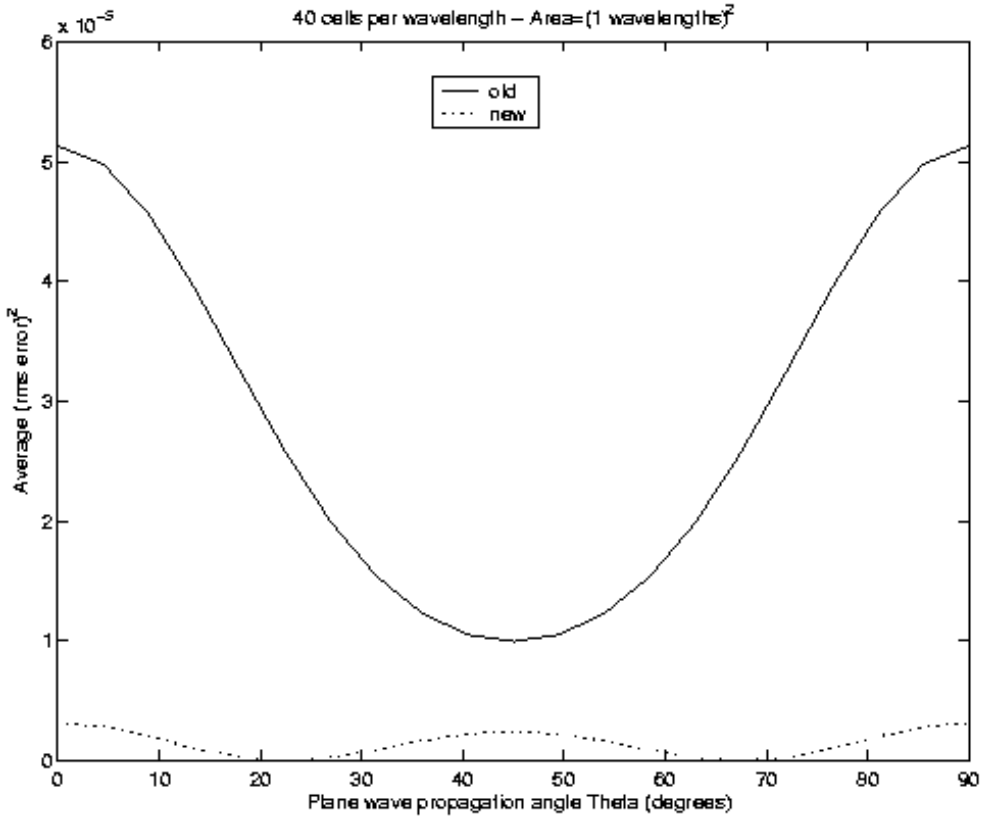

FIGURE 5

In Fig. 5, the classical variation in error due to propagation angle with respect to the grid is illustrated. Here a grid spacing of 40 cells per wavelength is chosen, and

the size of the computational domain is one square wavelength. The solid lines are used for the old or classical central difference approximation while dotted lines are used for the new method. This is consistent for Fig. 5, Fig. 6, and Fig. 7. Notice that for the old method, the errors are maximum at zero degrees (aligned with the grid axis) minimum at 45 degrees and then symmetrically increase to a maximum error at 90 degrees. This pattern periodically repeats every 90 degrees. In contrast, the new method is maximum at zero, minimum at 22.5 degrees and increase to a maximum at 45 degrees. This pattern repeats every 45 degrees. For all angles, the error calculated by the new weights is smaller than the old. This will be true for other grid resolutions.

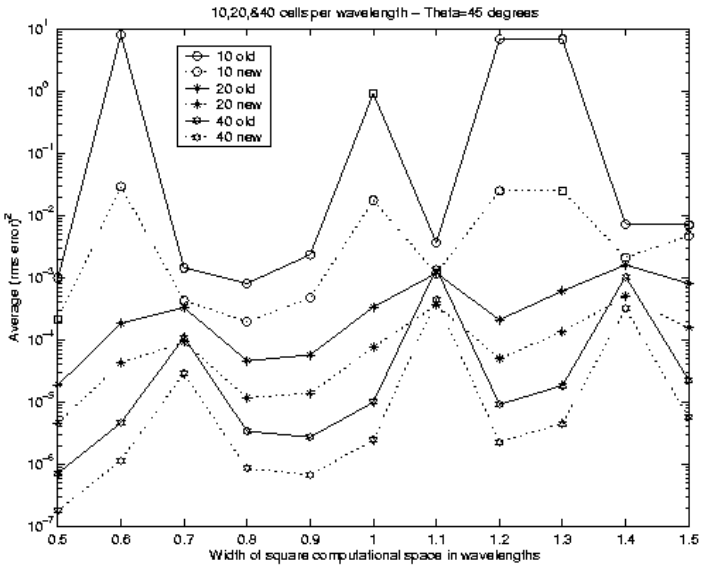

FIGURE 6

In the next experiment, the propagation angle and grid resolution is fixed while the computational size is varied from a size of $(0.5 \text{ wavelength})^2$ to $(1.5 \text{ wavelength})^2$. Three grid resolution are chosen as 10, 20, and 40 cells per wavelength. The two extreme cases of 45 and 22.5 degrees are chosen for the propagation angles. From Fig. 5, it is expected that the 45 degree case will produce the best results for the old weights and worst case for the the new weights. In contrast, the 22.5 case is expected to produce the smallest errors for the new weights.

Fig. 6 illustrates the experiment with a propagation of 45 degrees. Circles are used for the 10 cells per wavelength case, asterisks for the 20 cells per wavelength case, and stars for the 40 cells per wave length case. Notice that for this angle, the trends for both methods appear to follow the same pattern. Again for all cases, the new method produces lower errors.

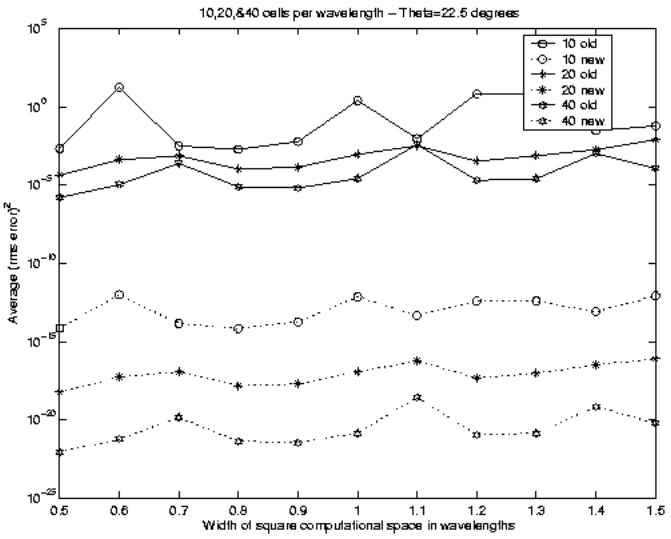

FIGURE 7

In Fig. 7, the same legends are used, but the propagation angle is changed to 22.5 degrees. At this angle the differences are extreme. For real scattering problems, a composite spectrum of propagation angles will be present. Because the new method produces smaller errors at all angles and all resolutions and all computational sizes, it will also produce smaller errors for scattering problems.

5.3. **Further calculations: interpolation formulas.** Using the method of Sects. 4.4 and 4.5, an ANSI C program was written, compiled and linked to ExprLib. The results for all three coefficients W_1, W_2, and W_3 were obtained in 0.38 sec. on an Intel 400 MHz i686 with 196M of real memory running Linux using gcc.

For example, we found that

```
W3 = ((- 2 Js2 + 2 Js10) Js5^2 + ((2 J0 J4 + (2 J2 - 2)
J3 + 6 J0 J2 - 4 J0) Js2 + (- 8 J0 J2 + 4 J0) Js10 - 2
J0 J4 + 2 J0) Js5 + ((- 2 J2 + 2) J4 + 2 J2s2 - 4 J2^2 +
2 J2) Js2^2 + ((- 2 J2s2 + 4 J2^2 - 2) Js10 + (- 2 J2 -
2 J0^2 + 2) J4 + (- 2 J0 J2s2 + 6 J0 J2 - 4 J0) J3 +
(- 2 J0^2 + 2) J2s2 - 4 J2^2 + (2 J0^2 + 2) J2) Js2 + ((4
J0^2 - 2) J2s2 + 4 J2^2 - 8 J0^2 J2 + 6 J0^2 - 2) Js10 +
(4 J0^2 J2 - 2 J0^2) J4 + (2 J0 J2s2 - 4 J0 J2^2 +
2 J0) J3 - 2 J0^2 J2s2 + 4 J0^2 J2^2 - 4 J0^2 J2) /
((Js2^2 - 1) Js5^2 + (- 2 J0 Js2^2 + (- 4 J0 J2 + 4 J0)
Js2 + 4 J0 J2 - 2 J0) Js5 + (- J2s2 + 4 J2 - 3) Js2^3 +
(- J2s2 + 4 J2 + J0^2 - 3) Js2^2 + ((4 J0^2 + 1) J2s2 - 2
J2^2 - 12 J0^2 J2 + 8 J0^2 + 1) Js2 + (- 4 J0^2 + 1) J2s2 +
(4 J0^2 - 2) J2^2 + 4 J0^2 J2 - 5 J0^2 + 1)
```

where

```
J0 = J_0(k h), J2 = J_0(2 k h), J3 = J_0(3 k h), J4 = J_0(4 k h),
Js2 = J_0 (sqrt (2) k h), Js5 = J_0 (sqrt (5) k h),
Js10 = J_0 (sqrt (10) k h), J2s2 = J_0 (2 sqrt (2) k h).
```

The other two coefficients, `W1` and `W2` are much longer. This fact prevents us from presenting them here. All the coefficients and the complete ANSI C source code used to derive them is available from the second author. Further details are available at [18] as well.

5.4. **Final summary.** We have presented a method to derive new numerical schemes with optimized weights for Helmholtz equation in one, two, and three dimensions and have also derived an optimal interpolation scheme for refined meshes. While it is possible to carry out hand calculations for these schemes as seen in [4], the complexity of the algebra involved is great and it is almost certain that such hand calculations will suffer from numerous typographical errors. By using symbolic computation, errors and tedium in deriving the necessary formulas were eliminated. We found that with relatively few functions out of the 130+ functions available in ExprLib, we were able to achieve all of our goals.

Certain extensions of this work to other equations are possible and symbolic computation is playing a role both in the theory and practical applications. The results of this new work will be reported in future papers.

l.lambe@bangor.ac.uk
Richard.Luczak@wpafb.af.mil
nehrbass@ee.eng.ohio-state.edu

Acknowledgements

This publication made possible through support provided by DoD High Performance Computing Modernization Program (HPCMP) Programming Environment and Training (PET) activities through Mississippi State University under the terms of Contract No. N62306-01-D-7110.
The authors would also like to thank the referees for their useful suggestions about organizing the presentation of this material.

References

[1] Andrew Ronald Mitchell and D. F. Griffiths. *The finite difference method in partial differential equations*. John Wiley & Sons Ltd., Chichester, 1980. A Wiley-Interscience Publication.
[2] Stanley J. Farlow. *Partial differential equations for scientists and engineers*. Dover Publications Inc., New York, 1993. Revised reprint of the 1982 original.
[3] J. W. Thomas. *Numerical partial differential equations: finite difference methods*. Springer-Verlag, New York, 1995.
[4] John Nehrbass. *Advances in Finite Difference Methods for Electromagnetic Modeling*. PhD thesis, Ohio State Univ., Dec 1996.
[5] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. John Wiley & Sons Inc., New York, 1984. Reprint of the 1972 edition, Selected Government Publications.
[6] Richard Pavelle. Macsyma: capabilities and applications to problems in engineering and the sciences. In *Applications of computer algebra (Philadelphia, Pa., 1984)*, pages 1–61. Kluwer·Nijhoff, Boston, MA, 1985.
[7] A. C. Hearn, J. D. Marti, et al. REDUCE user's manual, version 3.0 (standard LISP report). i Rand publication cp78 (4/83) (uucs-78-101) (University of Utah, Technical Report TR-6, 1978), Rand Corporation, Santa Monica, CA, USA, 1983.
[8] André Heck. *Introduction to Maple*. Springer-Verlag, New York, 1993.
[9] Stephen Wolfram. *The Mathematica book*. Wolfram Media, Inc., Champaign, IL, fourth edition, 1999.
[10] Richard D. Jenks and Robert S. Sutor. *Axiom. The scientific computation system*. Springer-Verlag, Berlin, Heidelberg, New York, 1992.
[11] J. W. Klop. Term rewriting systems. In *Handbook of logic in computer science, Vol. 2*, pages 1–116. Oxford Univ. Press, New York, 1992.
[12] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 263–297. Pergamon, Oxford, 1970.
[13] M. R. Sleep, M. J. Plasmeijer, and M. C. J. D. van Eekelen, editors. *Term graph rewriting*. John Wiley & Sons Ltd., Chichester, 1993. Theory and practice.
[14] Constantine A. Balanis. *Advanced Engineering Electromagnetics*. John Wiley & Sons, New York, 1989.
[15] I. S. Gradshteyn, I. M. Ryzhik, and A. Jeffrey. *Table of Integrals, Series, and Products*. Academic Press, New York, 1996.
[16] info@mssrc.com, MSSRC P.O. Box 6667, Bloomingdale, IL 60108, USA.
[17] Larry A. Lambe, Richard Luczak, and John W. Nehrbass. Symbolic computation in electromagnetic modeling. The 2001 Electromagnetic Code Consortium (EMCC) Annual Meeting (May), and the DOD UGC 2001 Meeting (June).
[18] Larry A. Lambe. Notes on Symbolic Computation. http://www.bangor.ac.uk/~mas019/symb/nsc.html (Sect. on "Symbolic Computation and the Finite Difference Method").
[19] Kenneth Hoffman and Ray Kunze. *Linear Algebra*. Prentice-Hall, Inc., New Jersey, 1971. Second Edition.

Appendix A. The Use of Symbolic Computation

A.1. **The optimal weights.** We wrote a short ExprLib program to calculate the integrals (34) above. Here are the results for p_1:

```
tor:~/tex/lambe/lln -> time p1

p1 = 2 j0

0.010u 0.000s 0:00.00 0.0%      0+0k 0+0io 141pf+0w
```

All four integrals were calculated in a similar way. Here is the result of running that program on the same machine as used above.

```
tor:~/tex/lambe/lln -> time omega3d

omega3D = kh^2 + 6 j0

0.000u 0.010s 0:00.00 0.0%      0+0k 0+0io 140pf+0w
```

Note that, to save time and space, the product `k*h` was processed as one variable, so the reported result is actually `k^2 h^2 + 6 j0`. Only one set of rewrite rules was needed to compute all four integrals. The details are available at [18].

A.2. **Deriving interpolation formulas using symbolic computation.** We want to show how to solve Problem 4.1. The first thing needed is a method for calculating the Gram-Schmidt formula. For this, a complex bilinear form is required. Next, the proper collection of rewrite rules will have to be set up to deal with the actual brackets (integrals)

$$< f, g >= \int_0^{2\pi} f(\phi)\overline{g(\phi)}d\phi \tag{67}$$

for the particular set of functions involved. In order to do this, we need to be able to define the bilinear operator to represent the bracket.

While the usual operators are built into ExprLib, users may also define their own operators. An operator represents a function of zero or more variables. The number of arguments of an operator is called its *arity*. An operator with zero arguments is a symbolic constant. It is easy to create new operators. One needs to specify a string (the print name) and an unsigned integer (the arity). The user can seamlessly use the operator by registering the operator in the parser table as the following code segment indicates.

```
myOp = opCreate ("myOp", 3);
registerOp (myOp);
```

So given Problem 4.1, the strategy is to

- define a binary operator "`brack`" and register it with the parser
- "teach" the library that `brack` is complex bilinear:

$$\begin{aligned} \texttt{brack}(\texttt{ax},\texttt{y}) &= \texttt{a}\,\texttt{brack}(\texttt{x},\texttt{y}), \\ \texttt{brack}(\texttt{y},\texttt{x}) &= \overline{\texttt{brack}(\texttt{x},\texttt{y})}, \\ \texttt{brack}(\texttt{x}+\texttt{y},\texttt{z}) &= \texttt{brack}(\texttt{x},\texttt{z}) + \texttt{brack}(\texttt{y},\texttt{z}), \end{aligned}$$

  etc., and finally,
- teach the library how to integrate the relevant class of functions.

In slightly more detail, we will use symbolic names f_i for the F_i and r for R and calculate the Gram-Schmidt formula for the f_i symbolically. By bilinearity of the bracket, the result will automatically be expanded in terms of a linear expression in the f_i with coefficients involving the brackets $< f_i, f_j >$. Similarly, the approximating expression for r will be linear in the f_i with coefficients involving the brackets $< r, f_i >$. The next step is to use routines for the symbolic integrations to calculate the exact values of $< F_i, F_j >$ and $< R, F_i >$ and substitute these actual values for the symbolic brackets above. This will yield the desired coefficients W_i. We need one more ExprLib function, viz.

```
Expr exprDiff (Expr f, String x);
```

As might be clear from the name, this function returns the derivative of the expression `f` with respect to the variable `x`.
The main program can be summarized by the following.

- Program the G–S formula and the formula `ans` given by

$$\texttt{ans} = \frac{< r, p_1 >}{< p_1, p_1 >} p_1 + \frac{< r, p_2 >}{< p_2, p_2 >} p_2 + \frac{< r, p_3 >}{< p_3, p_3 >} p_3$$

  symbolically in terms of symbolic expressions f_i representing the F_i (by bilinearity, the formula will automatically be expressed as a function of $< f_i, f_j > and < r, f_j >$ and will be linear in the f_i)
- pick out the coefficients, e.g. the coefficient of f_1 is just $\texttt{exprDiff}(\texttt{ans}, \texttt{f}_1)$
- calculate the brackets for the actual F_1, F_2, F_3, R
- use substitutions to replace the symbolic brackets in the coefficients by their actual values computed in the last step.

For a detailed explaination of the ExprLib notation used in this Sect., see [18]. Here is the code for the bilinear bracket. It is assumed an the operator `brack` of arity two has been defined and was registered with the parser. The name `obrack` ("outer bracket") is used to distinguish the routine from the operator (the "internal bracket"). It is assumed that the arguments `f` and `g` are linear in the variables that are specified in the `String` (`String` is the type `char *`) array `s` of length `n`. To get the coefficient of `s[i]` in `f`, for example, one simply uses `exprDiff (f, s[i])`. Use is also made of the built in operation `opApp2 (op, f, g)` which applies an operator `op` of arity two to two expressions `f` and `g` and returns the result which is of type `Expr`. The `obrack` routine calls an external routine called `obar` which acts as follows. If `arg` is not a bracket, `obar(arg) = arg`, otherwise, `obar(brack (a, b)) = brack (b, a)`.

```
Expr
obrack (Expr f, Expr g, String *s, UInt n)
{
  Expr ans = exprZero ();
  Expr coi, coj, base;
  int i, j;

  if (exprIsZero (f) || exprIsZero (g))
    return exprZero ();
  else
    {
      for (i = 1; i < n; i++)
```

```
      for (j = 1; j < n; j++)
        { coi = exprDiff (f, s[i]);
          coj = exprDiff (g, s[j]);
          coi = exprTimes (coi, obar (coj));
          base =
            opApp2 (brack, parseStrToExpr (s[i]),
                                parseStrToExpr (s[j]));
          ans = exprPlus (ans, exprTimes (coi, base));
        }
    }
  return ans;
}
```

Again, full details are available at [18].

Department of Informatics, Division of Mathematics, University of Wales, Bangor, Dean St., Bangor, Gwynedd LL57 1UT, United Kingdom

Science Applications International Corporation, 4031 Colonel Glenn Highway, Beavercreek, Ohio 45431-1600, United States

Aeronautical Systems Center, Major Shared Resource Center, Building 676, 2435 Fifth Street, Dayton, Ohio 45433 – 7802, United States