Discretization of the Poisson equation using the interpolating scaling function with applications

Jahrul M Alam\textsuperscript{a,}*, Raymond P Walsh\textsuperscript{a}, M Alamgir Hossain\textsuperscript{a}, Andrew M Rose\textsuperscript{a}

\textsuperscript{a}Department of Mathematics and Statistics, Memorial University, Canada, A1C 5S7

Abstract

Dyadic translations of the interpolating scaling function generate a basis that can be used to approximate functions and develop a multiresolution methodology for constructing smooth surfaces or curves. Many wavelet methods for solving partial differential equations are also derived from the interpolating scaling function. However, little is done for developing a higher order numerical discretization methodology using the scaling function. In this article, we have employed an iterative interpolation scheme for the construction of scaling functions in a two-dimensional mesh that is a finite collection of rectangles. We have studied the development of a weighted residual collocation method for approximating partial derivatives. We show that the discretization error is controlled by the order of the scaling function. The potential of this novel technique has been verified with some representative examples of the Poisson equation. We have extended the technique for solving nonlinear advection-diffusion equations, and simulated a shear driven flow in a square cavity at CFL = 2.5 (Courant Friedrichs Lewy) and Re = 1 000 (Reynolds number). Agreement with the reference solution at a large CFL = 2.5 explores the potential of this development for advection dominated problems.

\*Corresponding author

Email address: a\textsuperscript{a}lamj@mun.ca (Jahrul M Alam)

Preprint submitted to Computers and Mathematics with Applications May 11, 2014
1. Introduction

Obtaining an efficient and accurate solution of the Poisson equation is a long-standing challenge in many scientific and engineering applications. In chemical engineering, the Poisson equation models the electrostatic potential of an electric field with continuously distributed charges \[1-5\]. A solution is often obtained by evaluating the integral of the charged distribution using the fast multiple method. In Fluid Dynamics, a Poisson equation is solved for computing the divergence free velocity – in pressure-based approaches, or the stream function – in vorticity based approaches \([6,7]\). Although iterative techniques – such as the multigrid method – provides a rapid computing algorithm, in heat and mass transfer analysis (e.g. \([8]\)), the scalar Poisson equation takes more computational overhead than the accompanying vector advection-diffusion equations of the system (see \([9]\) for a comprehensive review). Developing novel algorithms for solving the Poisson equation is thus an active research area.

This research aims to employ a smooth surface for approximating the solution of the Poisson equation. Smooth curves and surfaces can be constructed by iterative interpolation of a given discrete sample. This iterative interpolation is also known as a subdivision scheme. It is built on the fundamental function of Deslauriers and Dubuc \([10]\). This process also constructs second generation wavelets on a complex geometry and adaptive multiresolution approximation schemes for functions with localized features. The beauty of this powerful mechanism lies in deriving the interpolating scaling function. The subdivision scheme can be
extended to discretize a partial derivative, as well as to obtain the best $N$ term approximation of a continuous or a discontinuous function that is sampled on an irregular or an adaptive mesh [11]. Such a function may be defined on a domain that is a collection of rectangles or rectangular prisms of non-uniform sizes, or a manifold, and does not have to be the entire space. In contrast, Fourier based techniques seek solutions in the entire space or require a periodic boundary condition.

A robust technique for solving the Poisson equation must use an accurate discretization method and a rapid algorithm for solving the discrete system. This article focuses mainly on the discretization, where the discrete system is solved with a Krylov method. A brief literature review indicates that the discretization technique presented in this article has not been studied for solving a Poisson equation, albeit similar approaches are available. The scaling function developed with a dyadic subdivision scheme is continuously differentiable, and may be at most twice differentiable, depending on the specific subdivision scheme. This limitation is a principal challenge for extending the subdivision scheme to solve partial differential equations (PDE) over a basis of scaling function. In the past decade, the second generation adaptive wavelet collocation method (AWCM) was verified as an efficient Computational Fluid Dynamics technique [12], where a wavelet basis is constructed from the interpolating scaling function (see [13–16]), and derivatives are approximated with a differential quadrature [17–19]. One also distinguishes the second generation wavelet method from the first generation one, where a finite difference method is employed for discretization and wavelets are used for the grid generation [20]. The discretization technique is one ingredient, which has restricted the AWCM to be applied only on simple geometries such as a rectangle or a rectangular prism.
A weighted residual collocation method for discretization in a more general domain can be developed. A brief literature review indicates that the development of collocation methods for PDEs, which is based on the interpolating scaling function, remains open. For example, the charge distribution was approximated with the scaling function, and the Poisson equation was solved for the electrostatic potential by evaluating an integral of the Green’s function \cite{21}. Vasilyev and Kevlahan \cite{14} applied a differential quadrature on a multi-level grid for solving two- and three-dimensional Poisson equations in rectangles and rectangular prisms, using a wavelet basis, which is generated from the scaling function. Mehra and Kevlahan \cite{22} extended this multi-level differential quadrature AWCM for solving PDEs on a sphere, where the scaling function is extended to build spherical wavelets.

In this article, we aim to study the development of a weighted residual collocation method for approximating derivatives of functions on a domain that is a finite collection of rectangles or rectangular prisms. Such a domain has boundaries parallel to coordinate axis, and may contain holes. First, we focus on accurate approximation of the Laplacian

$$\nabla^2 P = \rho \quad \text{in} \quad \Omega \subseteq \mathbb{R}^d, \ (d = 1, 2, 3),$$

(1)

using a weighted residual collocation method based on the interpolating scaling function, where the potential \( P(x) \), \( x \in \Omega \), is approximated by a smooth surface. Second, we study the solution of (1) and some of its applications in Computational Fluid Dynamics, where \( \rho(x) \) is given, and \( P(x) \) is desired. Third, we extend the methodology to approximate nonlinear PDEs and the Navier-Stokes equation.

In section 2 the weighted residual collocation method and the interpolating scaling function is studied briefly. Section 3 extends the developed methodology to solve PDEs. The present research has been summarized in section 4.
2. The collocation method and the interpolating scaling function

This section presents a methodology for constructing a set \( \{ \phi_k(x) \} \) of uniformly continuous functions that are at most twice differentiable. Each multi-dimensional scaling function, \( \phi_k(x) \), is an extension of the one-dimensional fundamental function, \( \varphi(x) \), that was originally outlined by Deslauriers and Dubuc [10]. A numerical solution of (1) is obtained by considering a trial solution that is spanned by the set \( \{ \phi_k(x) \} \). The methodology is now briefly outlined.

2.1. The collocation method

Consider two sets \( \{ \phi_k(x) \} \) and \( \{ \tilde{\phi}_k(x) \} \) of continuous functions. If we consider \( \phi_k(x) \) has the interpolating property over a set of nodes \( \{ x_k \} \); i.e., \( \phi_k(x_j) = \delta_{kj} \), and choose \( \tilde{\phi}_k(x) = \delta(x - x_k) \), then we have the following inner product

\[
\langle \phi_k(x), \tilde{\phi}_k(x) \rangle = \int_{\Omega} \phi_k(x) \tilde{\phi}_k(x) dx = \delta_{kj}.
\]

The method of weighted residual for solving (1) forms the \( N \) term trial solution

\[
P_N(x) = \sum_{k=0}^{N-1} c_k \phi_k(x)
\]

over the basis \( \{ \phi_k(x) \} \), where the residual \( r(x) = P(x) - P_N^{-1}(x) \) satisfies the vanishing weighted inner product, \( \langle r(x), \tilde{\phi}_k(x) \rangle = 0 \). This implies that the error of approximating \( P(x) \) by the trial solution \( P_N^{-1}(x) \) is exactly zero on all nodes \( x_k \). Evaluating the inner product, and using the orthogonality of \( \varphi \) and \( \tilde{\varphi} \), we get \( P(x_k) = c_k \). In other words, \( \lim_{N \to \infty} P_N^{-1}(x) = P(x) \). Replacing \( P(x) \) with \( P_N^{-1}(x) \) in (1), and setting the weighted inner product of the corresponding residual to zero, we have

\[
\sum_k \nabla^2 \phi_k(x_j) c_k = \rho(x_j),
\]
where $k$ and $j$ are the indices of all nodes which are not on the boundary. If the polynomials $\varphi_k(x)$ are exactly known or their derivatives are given, then the system (3) can be inverted – along with suitable boundary conditions – to find $c_j$'s, and an approximate solution of (1) is obtained from (2). The accuracy of the numerical solution is affected by how the basis $\varphi_k(x)$ is constructed. More specifically, $P^N(x)$ belongs to a function space that is generated from the basis. The trial solution (2) can be taken from any of the nested approximation spaces

$$V^0 \subseteq \ldots \subseteq V^{s-1} \subseteq V^s \subseteq V^{s+1} \ldots \quad \text{and} \quad \bigcup_{s=0}^{\infty} V^s = L_2(\Omega),$$

where each $V^s$ contains functions which do not oscillate at a frequency larger than $2^{s-1}$. These function spaces are also equipped with a nested sequence of $b$-adic meshes,

$$G^0 \subseteq \ldots \subseteq G^{s-1} \subseteq G^s \subseteq G^{s+1} \ldots \quad \text{and} \quad \lim_{s \to \infty} G^s = \Omega.$$

Such a mesh, $G^s$, is a finite collection of elements which are either rectangles or rectangular prisms, and has a total of $N$ nodes $x_k$ for $k = 0, \ldots, N - 1$ such that $x_k \in \Omega \subseteq \mathbb{R}^d$. The mesh $G^{s+1}$ is obtained by refining elements of the mesh $G^s$ with a factor of $b$ in each direction. In a $b$-adic refinement, an element gets $b^d$ child elements, and the mesh refinement can be managed efficiently with a tree data structure. In this article, we keep $b = 2$, and work with two-dimensional ($d = 2$) meshes unless otherwise stated, where each node $x_k$ may also be denoted by $x_{ij}$ for $0 \leq i < n_x$ and $0 \leq j < n_y$. With respect to the index $k$ of the mesh $G^s$, one sees that $x_k \in G^s$ and $x_{2k} \in G^{s+1}$ represent the same node because the meshes are nested, i.e. $G^s \subseteq G^{s+1}$. In other words, each node $x_{2k} \in G^{s+1}$ is present in the coarser mesh, and has $2^d - 1$ neighbors $x_{2k+1} \in G^{s+1}$ those were not present in the coarser mesh. For example, neighbors of $x_{2k} = x_{2i,j}$ are $x_{2k+1} = \{x_{2i+1,2j}, x_{2i,2j+1}, x_{2i+1,2j+1}\}$ with some exceptions on the boundaries.
Naturally, if a mesh was refined, and a function was evaluated on the coarse mesh, one may use an interpolation of the coarse mesh data to the refined mesh. However, in the present development, the interpolation process is employed for constructing a set \( \{ \phi_k(x) \} \) of interpolating scaling functions as the basis of the approximation space \( \mathcal{V}^s \).

2.2. Iterative interpolation

The starting point for the iterative interpolation is a function evaluation \( \{ c_k \} \) at each node \( x_k \) of the mesh \( G^s \). This mesh \( G^s \) is refined, for example, dyadically to form a new mesh \( G^{s+1} \). Fig 1(a) uses • and \( \times \) to denote nodes in \( G^s \) and \( G^{s+1}\setminus G^s \), respectively. The given sample \( \{ c_k \} \) is interpolated to \( \times \) nodes, and a new sample is obtained in the mesh \( G^{s+1} \), which has been illustrated in Fig 1. The process can be repeated until \( s \to \infty \). The iterative interpolation extends \( \{ c_k \} \) to a uniformly continuous function \( \varphi(x) \) in the entire domain \( \Omega \), which may be twice differentiable [10]. More specifically, Deslauriers and Dubuc [10] showed that if one assigns \( c_k = 1 \) on a specific node \( x_k \) and \( c_k = 0 \) on all other nodes, the iterative interpolation of this data \( \{ c_k \} \) results into a uniformly continuous fundamental function. First, at each node \( x_k \in G^s \), \( \varphi(x) = \sum_k c_k P_k(x) \) is built by constructing a polynomial

\[
P_k(x_i) = \begin{cases} 
1 & \text{if } x_i = x_k \\
0 & \text{if } x_i \neq x_k 
\end{cases}
\]

using \( 2p \times 2p \) neighbors of the node \( x_k \) such that \( \varphi(x_k) = c_k \). Next, \( \varphi(x) \) is extended to the mesh \( G^{s+1} \) by assigning \( \varphi(x_{2k}) = \varphi(x_k) \) (on • nodes) and \( \varphi(x_{2k+1}) = P_k(x_{2k+1}) \) (on \( \times \) nodes). Fig 1(a) shows \( 2p \times 2p \) neighbors of the node \( x_k \) in the coarse mesh and its fine mesh neighbors \( x_{2k+1} \). As a result, we have \( \varphi(x_k) \) for all nodes \( x_k \in G^{s+1} \). The interpolation is iterated on \( G^{s+1} \) to extend \( \varphi(x) \) on the mesh.
$G^{s+2}$. Clearly, one obtains $\varphi(x)$ for every $x \in \Omega$ by employing the interpolation and the subdivision repeatedly.

Fig 1(b) shows a sample $\{c_k\}$ on a $5 \times 5$ initial mesh. This data can be interpolated with $p = 2$ to form a new $9 \times 9$ sample $\{c_k\}$ (e.g. Fig 1(c)). We now refine the mesh, and repeat the interpolation on each refined mesh with $p = 2$, until we get a $65 \times 65$ mesh. Fig 1(d) shows the constructed function on the $65 \times 65$ mesh.

A technical detail of this dyadic interpolation of order $p$ (DIP) for one-dimensional data and the properties of the resulting scaling function are given by Deslauriers and Dubuc [10].

2.3. Interpolating scaling function

This section aims to generate a polynomial basis for the space $\mathcal{V}^s$ so that we can construct the trial solution (2) on a mesh $G^s$ that is a collection of rectangles. At each node $x_k$ of $G^s$, we associate a scaling function $\varphi_k(x) = \varphi(x - x_k)$ applying the dyadic interpolation of order $p$, and as a result, the basis $\{\varphi_k(x)\}$ of $\mathcal{V}^s$ is obtained. Clearly, the basis $\{\varphi_k(x)\}$ is also a dyadic translation of the scaling function $\varphi(x)$.

We now demonstrate a few examples of constructed scaling functions $\varphi(x)$ in the square $[-1, 1] \times [-1, 1]$. They are presented in Fig 2 for $p = 2, 4, 6, \text{ and } 8$. For each $p$, $\varphi(x, 0)$ is also shown. Each of these two-dimensional scaling functions, $\varphi(x)$, is symmetric with respect to $x = 0$, $y = 0$, and $y = \pm x$. Note that the exact mathematical form of $\varphi(x)$ may not be known. We only need to know its initial function evaluation $\{c_k\}$, and the interpolation process.

In order to have an equivalent resolution, we assigned $\varphi(x, y) = 1$ for $(x, y) = (0, 0)$, and $\varphi(x, y) = 0$ for all other nodes in a $33 \times 33$ mesh of the square $[-1, 1] \times [-1, 1]$. As can be seen from Fig 2, the support of $\varphi(x, y)$ increases with $p$. For
Figure 1: Iterative interpolation and subdivision process. (a) The node $x_k \in G^s$ and its $2p \times 2p$ neighbors in $G^s$ for $p = 2$ are marked with •. Nodes in $G^{s+1}$ those are not present in $G^s$ are marked with ×, and among them, $x_{2k+1}$ are marked with ⊗. $P_k(x)$ takes a value 1 on $x_k$ and 0 on all other nodes •, and is used to interpolate new values on three ⊗ nodes. We start with $\varphi(x) = c_k$ on all • nodes, and evaluate $\varphi(x) = \sum_k c_k P_k(x)$ on all × nodes, thereby resulting into $\varphi(x)$ on $G^{s+1}$. (b) Iterative interpolation is applied with the initial $5 \times 5$ sample $\{c_k\}$. (c) A function is constructed on $9 \times 9$ and (d) $65 \times 65$ meshes.
Figure 2: Scaling function \( \varphi(x, y) \) in the domain \([-1, 1] \times [-1, 1]\). In the left column, only the portion \([-0.5, 0.5] \times [-0.5, 0.5]\) of the domain is shown for clarity. Vertically downward, rows correspond to \( p = 2, 4, 6, \text{ and } 8 \), respectively. In the right column, \( \varphi(x, 0) \) is shown for each \( p \). Each curve in the right column has exactly \( 2p \) zeros; however, at large \( p \), fluctuation of \( \varphi(x, 0) \) away from the center is not visible with naked eye.


\( p = 8 \), \( \varphi(x, y) \) vanishes for all \((x, y) \notin (-15\Delta x, 15\Delta x) \times (-15\Delta y, 15\Delta y)\), where \( \Delta x = 2/32 \) and \( \Delta y = 2/32 \).

If the scaling function \( \varphi(x) \) is built on an one-dimensional mesh, then it is exactly the fundamental function \( \varphi(x) \) of Deslauriers and Dubuc [10], which has the following properties.

- \( \varphi(x) \) is an interpolating polynomial, which vanishes outside the interval \([x_{-2p+1}, x_{2p-1}]\), where \( p \) is an integer. Moreover, \( \varphi(x_0) = 1 \) and \( \varphi(x) \) has exactly \( 2p \) zeros in the interval \([x_{-2p+1}, x_{2p-1}]\).

- \( \varphi(x) \) is symmetric about \( x = x_0 \); i.e., \( \varphi(x) \) is an even function.

- \( \varphi(x) \) is uniformly continuous for all \( p \) on any finite interval, and is differentiable for \( p > 1 \). Moreover, \( \varphi(x) \) has at least two continuous derivatives for \( p = 3 \) (see, [10]).

- \( \varphi(x) \) is translated to construct a basis \( \{\varphi_k(x)\} \) that satisfies \( \varphi_k(x_j) = \delta_{kj} \) and reproduces polynomials up to degree \( 2p - 1 \).

- For the even function

\[
  f(x) = \begin{cases} 
  1 & \text{for } x = x_k \\
  0 & \text{for } x > x_{k+2p-1} \text{ and } x < x_{k-2p+1}
  \end{cases}
\]

having \( 2p \) zeros within its support \([x_{k-2p+1}, x_{k+2p-1}]\), we have

\[
  f(x) = \sum_{k=-4p+2}^{4p-2} c_k \varphi_k(x).
\]

This property is also known as the two scale relation because \( f(x) \) is defined on \( G^s \) and \( \varphi_k(x) \) is defined on \( G^{s+1} \).
2.4. Differentiation

In this section, we study the weighted residual collocation method for the numerical differentiation of the trial solution (2) on the basis of the scaling function. In a collocation method, one aims to compute \( \frac{\partial}{\partial x} P^{N-1}(x) \) at all nodes \( x_k \) of a mesh \( G^s \). This is done by using some properties of \( \varphi_k(x) \), and considering a corresponding expansion (2) for \( \frac{\partial}{\partial x} P^{N-1}(x) \). Since \( \varphi_k(x) \) is an even function with respect to \( x = x_k \), and \( \varphi_k(x) \) has exactly \( 2p \) zeros within its support \( [x_k-2p+1, x_k+2p-1] \), the following statements are true. (i) The 1st derivative \( \varphi'_k(x) \) is an odd function, (ii) it vanishes at \( x_k \), i.e. \( \varphi'_k(x_k) = 0 \), (iii) \( \varphi'_k(x) \) takes nonzero values at \( 2p - 2 \) zeros of \( \varphi_k(x) \) in \( (x_k-2p+1, x_k+2p-1) \), and \( \varphi'_k(x) \) vanishes for all other \( x \not\in (x_k-2p+1, x_k+2p-1) \).

Using these properties, it is easy to see that the inner product \( \langle \frac{\partial}{\partial x} P^{N-1}(x), \tilde{\varphi}(x) \rangle \) results into

\[
\frac{\partial}{\partial x} P^{N-1}(x_k) = \sum_{j=k-2p+1}^{j=k+2p-1} c_j \varphi'_k(x_j). \tag{4}
\]

To evaluate the right side of (4), let us obtain the nodal values of \( \varphi_k(x) \) for \( x \in [x_k-2p+1, x_k+2p-1] \) from the interpolation process, without knowing the actual mathematical form of \( \varphi_k(x) \), using the the barycentric formula

\[
\varphi_k(x) = \frac{w_k(x)}{\sum_{l=k-2p+1}^{k+2p-1} w_l(x)}
\]

at \( x_j \) for \( j = k - 2p + 1, \ldots, k + 2p - 1 \) (see [10, 23, 24]). The weights \( w_k(x) \) are associated with \( 2p + 1 \) nodes, and are extended from the iterative interpolation process that derives \( \varphi(x) \). In order to employ the weighted residual collocation method, let us define two weights,

\[
\frac{1}{w_k(x)} = (x - x_k) \prod_{j \neq k} (x_k - x_j) \tag{5}
\]
and
\[ s(x) = \sum_l w_l(x)(x - x_l) \]
and assume the weighted inner product
\[ \langle [\varphi_k(x)s(x)]', \tilde{\varphi}_k(x) \rangle = 0. \]

A quick calculation leads to
\[ \varphi'_k(x_j) = \begin{cases} \frac{w_j(x)(x-x_k)}{w_l(x)(x-x_j)(x_j-x_k)} & \text{for } k \neq j \\ -\sum_{k \neq j} \varphi'_k(x_j) & \text{for } k = j. \end{cases} \]

Clearly, knowing the ingredients, \( w_k \)'s, of the iterative interpolation, we are able to compute derivatives of \( \varphi_k(x) \) exactly on all nodes. Using a equally spaced one-dimensional dyadic mesh with \( p = 2, 3 \), we find that the values \( \varphi'_k(x_j) \) obtained from the above formula agrees exactly with those presented by Deslauriers and Dubuc [10].

Using the above expression for \( \varphi'_k(x_j) \), (4) provides the first derivative of the trial solution (2) at all nodes. It is also clear from (4) that the first derivative of the trial solution is given by the products of its nodal values with \( \varphi'_k(x_j) \)'s. Clearly, the process has \( O(N) \) complexity, which does not require global operations on the \( N \times N \) differentiation matrix. In other words, one does not need to store the differentiation matrix explicitly, and the overall CPU time for discretization is asymptotically optimal if \( N \to \infty \).

The weighted residual collocation method for computing the second order derivative of the trial solution (2) is computed using the nodal values \( \varphi'_k(x_j) \). Let us denote \( c'_k = \frac{\partial}{\partial x} P^{N-1}(x_k) \), and rewrite (4) for the second derivative
\[ \frac{\partial^2}{\partial x^2} P^{N-1}(x_k) = \sum_{j=k-2p+1}^{j=k+2p-1} c'_j \varphi'_k(x_j). \]
Since $\varphi'_k(x_j)$’s have been computed, $c'_j$’s can be computed from (4), and hence, the right side of (6) can be evaluated. However, we can also rewrite the right side of (6) as

$$\sum_{j=k-2p+1}^{j=k+2p-1} c'_j \varphi'_k(x_j) = \sum_{j=k-2p+1}^{j=k+2p-1} c_j \varphi''_k(x_j),$$

where $\varphi''_k(x_j)$’s are some necessary weights for the second derivative. Using a similar approach, the weights $\varphi''_k(x_j)$ for the second order derivative of the trial solution (2) are given in terms of the first derivative of the scaling function by

$$\varphi''_k(x_j) = \begin{cases} 
-2\varphi'_k(x_j) \left[ \sum_{i \neq k} \varphi'_k(x_i) - \frac{1}{x_j-x_k} \right] & \text{for } k \neq j \\
- \sum_{k \neq j} \varphi''_k(x_j) & \text{for } k = j.
\end{cases}$$

For a dyadic interpolation with $p = 3$, $\varphi_k(x)$ is twice differentiable. We have checked that eq. (6) provides nodal values of the second derivatives of $\varphi_k(x)$, which agree exactly with those derived by Deslauriers and Dubuc [10].

2.4.1. Example

Consider the function

$$u(x, y) = \frac{1}{\pi \nu} \exp \left( -\frac{x^2 + y^2}{\nu} \right), \quad (x, y) \in [-1, 1] \times [-1, 1]$$

for which $\nabla^2 u$ is known exactly, and we have used this function to check the numerical error. $u(x, y)$ has a singularity and a localized structure, depending on the value of $\nu$, near the origin $(0, 0)$. For $\nu = 10^{-2}$, we have estimated the error $|\nabla^2 u(x, y) - \nabla^2 u^{N-1}(x, y)|_\infty$ on a $129 \times 129$ mesh with $p = 2, 3, 4, 5, 6$, and the results are listed in the Table [1]. As expected, the error is reduced with increased order ($p$) of interpolation. This behavior of the error is consistent with the error bounds given by Deslauriers and Dubuc [10].
Table 1: The errors $|\nabla^2 u(x, y) - \nabla^2 u^{N-1}(x, y)|_\infty$ for estimating the Laplacian of the function in example (2.4.1) with $p = 2, 3, 4, 5,$ and $6$ have been listed.

3. Application to partial differential equations

Numerical solution of PDEs requires a discretization technique and a methodology for solving the resulting system of algebraic equations. The focus of this article is on the discretization. A multi-level solution methodology may be developed to take full benefits of the present research. In this section, we examine the proposed discretization methodology with a few representative examples, where a Krylov method has been used to solve the discrete system. More specifically, we have employed the restarted GMRES (generalized minimal residual) algorithm [25].

3.1. A toy model for the charged distribution

To demonstrate the accuracy of the present method for solving (1), consider a charged distribution as shown in Fig 3(a). Applying the GMRES method to the discretization of (1), we compute the potential $P(x, y)$, which is shown in Fig 3(b). When the numerical solution $P(x, y)$ is compared with the exact solution in a color filled contour plot, no difference can be identified. For a more quantitative comparison, we have compared the numerical potential $P(x, 0.5)$ with its correspond-
Figure 3: (a) The prescribed charged distribution, \( \rho(x, y) \). (b) The numerical solution of (1) for \( P(x, y) \). (c) \( P(x, 5.0) \) has been compared with its exact value.

An excellent agreement has been observed, albeit negligible error appears near the center of the domain, where the potential field has a sharp gradient.

3.2. Helmholtz-Hodge decomposition of a vector field

A vector field can be decomposed as the sum of a divergence free vector field and a curl free or conservative vector field, \( \text{i.e. } \mathbf{u}^* = \mathbf{u} + \nabla P \), where \( \nabla \cdot \mathbf{u} = 0 \) and \( P(x, y) \) is a scalar potential. When the incompressible Navier-Stokes equation is solved with a fractional step time integration scheme, which was originally proposed by Chorin \[26\], the Helmholtz-Hodge decomposition is employed \[7\].
Let \( u^* \) be a given velocity field such that \( \nabla \cdot u^* \neq 0 \), we have

\[
\nabla^2 P = \nabla \cdot u^*, \quad \nabla P \cdot \hat{n} = 0.
\]

(7)

Eq. (7) is one application of the Poisson model (1) in Fluid Dynamics, where \( \nabla \cdot u^* \) appears as if a charge distribution. Eq. (7) is solved to compute the divergence-free component according to

\[
u = u^* - \nabla P.
\]

In order to verify the accuracy of the numerical solution, consider the following manufactured velocities

Taylor-Green velocity
\[
\begin{align*}
  u^* &= -\cos(2\pi x) \sin(2\pi y) + \pi \sin(4\pi x), \\
  v^* &= \sin(2\pi x) \cos(2\pi y) + \pi \sin(4\pi y),
\end{align*}
\]

which are constructed by adding noise terms into the Taylor-Green vortex solution of the incompressible Navier-Stokes equation – a commonly used CFD toy model (e.g. [27]). We can verify that

\[
P = -\frac{1}{4} \left[ \cos(4\pi x) + \cos(4\pi y) \right]
\]

is a solution of (7) in \([0, 1] \times [0, 1]\), and

\[
u = -\cos(2\pi x) \sin(2\pi y), \quad v = \sin(2\pi x) \cos(2\pi y).
\]

Eq (7) is discretized with the proposed method, and the resulting system of equations is solved with a GMRES method [25]. The computed potential \( P^{N-1}(x, y) \) with \( N = 65 \times 65 \) is presented in Fig 4(a), which is compared with the exact solution in Fig 4(b). The numerical solution \( u^{N-1}(0.5, y) \) is compared with the exact solution \( u(0.5, y) \) in Fig 4(c) as well as \( v^{N-1}(0.5, y) \) is compared with the exact
solution \( v(0.5, y) \) in Fig 4(d). From these plots, it is hard to see the difference between the exact and the numerical solution with a naked eye. These numerical experiments verify the performance of the present development for solving a Poisson equation with Neumann boundary conditions.

3.3. Poisson equation in complex geometry

The present collocation method can adopt a complex geometry. To verify this, we have solved (7) in the domain \([-2, 2] \times [-2, 2] \setminus [-0.5, 0.5] \times [-0.5, 0.5] \times [-2, 2] \times [-2, 2] \setminus [-0.5, 0.5], which has a hole. Since the exact solution is known, we can assess the accuracy. Without going to further details, let us present the numerical solution and the associated mesh. For a better visualization, we have presented only a portion of the mesh and the solution in the entire domain in Fig 5. This example verifies two benefits of the present collocation method. First, we can work with complex geometry which is a collection of rectangles. Second, we can employ a nonuniform mesh.

3.4. Nonlinear advection diffusion problem

In this section, we extend the proposed methodology to solve the following nonlinear advection-diffusion model:

\[
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \nu \nabla^2 \mathbf{u},
\]

where \( \mathbf{u} = [u_x, u_y]^T \). Using the same initial and boundary conditions from the reference solution (e.g. problem 1) presented by Zhu et al. [28], eq. (8) has been solved in the domain \([0, 1] \times [0, 1]\). A Poisson like nonlinear system of equations, \( L(\mathbf{u}) = f \), such that

\[
\frac{-\nu \nabla^2 \mathbf{u}^{n+1} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \frac{2}{\Delta t} \mathbf{u}^{n+1}}{L(\mathbf{u})} = \frac{\nu \nabla^2 \mathbf{u}^n - \mathbf{u}^n \cdot \nabla \mathbf{u}^n + \frac{2}{\Delta t} \mathbf{u}^n}{f}
\]
Figure 4: Numerical solution of (7) for $P(x, y)$, $u(x, y)$, and $v(x, y)$; (a) $P(x, y)$. Computed $P(x, 0.5)$, $u(0.5, y)$, and $v(x, 0.5)$ are compared with the corresponding exact values in (b), (c), and (d), respectively. An excellent agreement is seen.

Figure 5: Numerical solution of (7) in a domain with a hole using a nonuniform mesh. (a) Solution, (b) mesh; only a portion of the mesh is shown for clarity.
is obtained by discretizing (8) in time and space, where a second order Crank-Nicolson method is adopted, and a trial solution of the form (2) is assumed for each component of $u$. The nonlinear system is solved with a Newton method, where the Jacobian free Newton Krylov (JFNK) algorithm \cite{29} has been employed. In the present work, the discretization differs – both in space and time – from what was used in \cite{28}.

We have analyzed this example with a time step, $\Delta t$, between $10^{-1}$ and $10^{-4}$, as well as, a resolution between $33 \times 33$ and $129 \times 129$. With $\Delta t = 10^{-1}$ and $\nu = 1.25 \times 10^{-2}$ we have found the maximum absolute error $5.05 \times 10^{-4}$. Zhu et al. \cite{28} reported a maximum absolute error $7.5 \times 10^{-4}$ with $\Delta t = 10^{-4}$ and $\nu = 1.25 \times 10^{-2}$. Using a $\Delta t$ that is 100 times larger than what was used by Zhu et al. \cite{28}, we see that the error is slightly smaller than what was presented by Zhu et al. \cite{28}. This explains the performance of the present method for the nonlinear advection-diffusion problem.

In Fig. 6 we compare numerical solution with the exact solution, where $\Delta t = 10^{-2}$, $\nu = 10^{-3}$, and $N = 129 \times 129$. The plots include $u_x(x, 0.5)$, $u_x(0.5, y)$, $u_y(x, 0.5)$, and $u_y(0.5, y)$. The excellent agreement between the exact and the numerical solutions encourage the extension of the methodology to the field of Computational Fluid Dynamics.

3.5. Computational Fluid Dynamics

Solving of the incompressible Navier-Stokes equation is a challenging endeavor in the field of Computational Fluid Dynamics. E and Liu \cite{30} discussed the occurrence of artificial numerical boundary layer if a classical fractional step method is employed (see also, \cite{9}). In this article, we do not have enough room to address these unresolved challenges; however, we aim to demonstrate the po-
Figure 6: Numerical solutions of (8) (a) $u_x(x, 0.5)$, (b) $u_y(0.5, y)$, (c) $u_x(x, 0.5)$, and $u_y(0.5, y)$.

tential of the present development to the field of CFD, using a simulation of the classical shear driven flow. This is an incompressible flow in a square cavity $[0, 1] \times [0, 1]$ with boundary conditions $u = 0 = v$ on boundaries, $x = 0$, $x = 1$, and $y = 0$. To model the lid of the cavity with a constant speed, $u = 1$ and $v = 0$ are used on the boundary, $y = 1$. Since we can not derive the exact solution for the shear driven flow, a reference simulation may be used to verify our simulation. Ghia et al. [31] and Ghadimi et al. [32]) examined a similar shear driven flow, using the steady-state vorticity equation (see eqs(1-2) of [32]). Although the present numerical method and the set of equations are different than those used by Ghia et al. [31] and Ghadimi et al. [32], these reference results are useful feedback for assessing a simulation of shear driven flow using the proposed collocation method.

We have modelled the shear driven flow with the incompressible Navier-Stokes
which has been discretized using a Crank-Nicolson scheme for the viscous term, the advection term, and the pressure gradient term. In order to compute mass and momentum transfer simultaneously at each time step, we have adopted the method of artificial compressibility on a collocated mesh, which was proposed by Chorin [33]. 256 uniform rectangles in both the $x$ and $y$ directions have been used, where the total number of degrees of freedom is $N = 257 \times 257$. With this $N$, we have $\Delta x = \Delta y \sim 4 \times 10^{-3}$. Another reason for adopting the method of artificial compressibility is that we want to examine the proposed Poisson solver to a nonlinear system of Poisson like equations, without depending on pressure correction steps, which is what would be needed otherwise.

A nonlinear system of $3N$ coupled equations is solved at each time step with the Jacobian-free Newton-Krylov algorithm [16, 29]. A number of simulations with $N$ between $33 \times 33$ and $257 \times 257$ and Reynolds number, $Re$, between 100 and 1 000 have been considered. For $N = 257 \times 257$, we have tested time steps ($\Delta t$) between $10^{-5}$ and $10^{-2}$. Analysis of the simulated data shows that the average CPU time in the wall-clock unit, for CFL numbers 0.25 ($\Delta t = 10^{-3}$) and 2.5 ($\Delta t = 10^{-2}$), are about 12 days and 1 day, respectively, for the same dimensionless integration time. Most classical CFD code would use a CFL $< 1$ because the advection term is typically treated with an explicit scheme. In contrast, implicit treatment in the present work has saved about 90% of the CPU time for this particular test.

For $Re = 1000$, the simulated velocity fields are presented in Fig 7(a-b), showing an overall pattern of the circulation, which are in good agreement with
Figure 7: Velocities for the shear driven flow. (a) $u_x(x,y)$, (b) $u_y(x,y)$, and (c) $u_x(0.5,y)$ for $Re = 100, 400, \text{ and } 1000$. The overall circulation in (a-b) and the velocity profiled in c are in good agreement with the reference results. Previously reported results [8, 31, 32]. In Fig 7c, we have compared the simulated velocity, $u(0.5,y)$, for $Re = 100, 400, \text{ and } 1000$. The pattern of the velocity profile with increasing $Re$ is similar to what was presented by Ghia et al. [31] and Ghadimi et al. [32]. Table 2 confirm a quantitative assessment of the present simulation with respect to the reference model, where the minimum values of $u_x(x,y)$ and $u_y(x,y)$ have been reported for $Re = 100, 400, \text{ and } 1000$. 

23
Table 2: Comparison of the velocities $u_x$ and $u_y$ for the shear driven flow simulation. The agreement between two simulations is excellent, albeit different methods and equations have been used to model the shear driven flow.

4. Conclusion and future direction

This article outlines a methodology for discretizing partial derivatives using the interpolating scaling functions. This scaling function forms the foundation of multiresolution approximation and multiscale simulation methodologies. A weighted residual collocation method has been developed and the performance of the method has been verified in this article.

Results show that the numerical error can be controlled by increasing the order of dyadic interpolation, in which the global number of the degrees of freedom remains the same. A Poisson solver that is based on the interpolating scaling function has been verified, where the exact solution is known. Extension to a nonlinear advection-diffusion problem shows that the accuracy remains equivalent to a reference solution, although the present work has used a time step as large as 100 times of what was used in the reference model. Analysis with the shear driven flow exhibits the potential of the present development to the field of CFD.

These results provide potential feedback for constructing fast multiresolution algorithms for numerical solution of partial differential equation. We are in-
interested to extend the proposed discretization methodology, using the adaptive wavelet basis so that adaptively refined local meshes could be employed. Parallel implementation of this work is currently underway.

Acknowledgments

JMA acknowledges financial support from the National Science and Research Council I (NSERC), Canada.

References

[1] B. t Roux, T. Simonson, Implicit solvent models, Biophysical Chemistry 78 (1999) 1 – 20.

[2] J. D. Jackson, Classical Electrodynamics Third Edition, Wiley, third edition, 1998.

[3] F. Fogolari, A. Brigo, H. Molinari, The poisson boltzmann equation for biomolecular electrostatics: a tool for structural biology, J. Mol. Recognit. 15 (2002) 377–92.

[4] J. M. Alam, J. M. Penney, A lagrangian approach for modelling electro-kinetic mass transfer in microchannels, International Journal of Heat and Mass Transfer 55 (2012) 7847 –57.

[5] J. M. Alam, J. C. Bowman, Energy conserving simulation of incompressible electro-osmotic and pressure driven flow, Theoret. Comput. Fluid Dynamics. 16 (2002) 133–50.
[6] L. S. Tuckerman, Divergence-free velocity fields in nonperiodic geometries, Journal of Computational Physics 80 (1989) 403–41.

[7] J. C. Tannehill, D. A. Anderson, R. H. Pletcher, Computational Fluid Mechanics Heat Transfer, Taylor and Francis, 1997.

[8] J. M. Alam, N. K.-R. Kevlahan, O. V. Vasilyev, Z. Hossain, A multi-resolution model for the simulation of transient heat and mass transfer, Numerical Heat Transfer, Part B 61 (2012) 1–24.

[9] O. San, A. E. Staples, A coarse-grid projection method for accelerating incompressible flow computations, J. Comput. Phys. 233 (2013) 480–508.

[10] G. Deslauriers, S. Dubuc, Symmetric iterative interpolation process, Constructive Approximation 5 (1989) 49–68.

[11] R. A. DeVore, Nonlinear approximation, Acta Numerica (1998) 51–150.

[12] K. Schneider, O. V. Vasilyev, Wavelet methods in computational fluid dynamics, Annu. Rev. Fluid Mech. 42 (2010) 473–503.

[13] O. V. Vasilyev, C. Bowman, Second-generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–93.

[14] O. V. Vasilyev, N.-R. Kevlahan, An adaptive multilevel wavelet collocation method for elliptic problems, J. Comput. Phys. 206 (2005) 412–31.

[15] J. Alam, N. K.-R. Kevlahan, O. Vasilyev, Simultaneous space–time adaptive solution of nonlinear parabolic differential equations, Journal of Computational Physics 214 (2006) 829–57.
[16] J. Alam, Towards a multi-scale approach for computational atmospheric modelling, Monthly Weather Review 139 (2011).

[17] O. V. Vasilyev, S. Paolucci, M. Sen, A multilevel wavelet collocation method for solving partial differential equations in a finite domain, J. Comput. Phys. 120 (1995) 33–47.

[18] O. V. Vasilyev, S. Paolucci, A dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain, J. Comput. Phys. 125 (1996) 498–512.

[19] O. V. Vasilyev, S. Paolucci, A fast adaptive wavelet collocation algorithm for multidimensional PDEs, J. Comput. Phys. 138 (1997) 16–56.

[20] L. Jameson, A wavelet-optimized, very high order adaptive grid and order numerical method, SIAM J. Sci. Comput. 19 (1998) 1980–2013.

[21] L. Genovese, T. Deutsch, S. Goedecker, Efficient and accurate three-dimensional poisson solver for surface problems, The Journal of Chemical Physics 127 (2007).

[22] M. Mehra, N.-R. Kevlahan, An adaptive wavelet collocation method for the solution of partial differential equations on the sphere, J. Comput. Phys. 227 (2008) 5610–32.

[23] L. B. Winrich, Note on a comparison of evaluation schemes for the interpolating polynomial, The Computer Journal 12 (1969) 154–5.

[24] J.-P. Berrut, L. N. Trefethen, Barycentric lagrange interpolation, SIAM Rev 46 (2004) 501–17.
[25] P. Brown, Y. Saad, Convergence theory for nonlinear newton–krylov algorithms, SIAM J. Opt. 4 (1994) 297–330.

[26] A. Chorin, Numerical solution of navier-stokes equation, Math. Commp. 22 (1968) 745–62.

[27] M. Jobelin, C. Lapuerta, J.-C. Latch, P. Angot, B. Piar, A finite element penalty-projection method for incompressible flows, Journal of Computational Physics 217 (2006) 502 –18.

[28] H. Zhu, H. Shu, M. Ding, Numerical solutions of two-dimensional burgers’ equations by discrete adomian decomposition method, Comput. Math. Appl. 60 (2010) 840–8.

[29] D. A. Knoll, D. E. Keyes, Jacobian-free newton-krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–97.

[30] W. E, J.-G. Liu, Projection method I: Convergence and numerical boundary layers, SIAM Journal on Numerical Analysis 32 (1995) 1017–57.

[31] U. Ghia, K. Ghia, C. Shin, High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method, Journal of Computational Physics 48 (1982) 387–411.

[32] P. Ghadimi, M. Fard, A. Dashtimanesh, Application of an iterative high order difference scheme along with an explicit system solver for solution of stream function-vorticity form of navier stokes equations, J. Fluids Eng. 135 (2013) 041401–11.
[33] A. J. Chorin, A numerical method for solving incompressible viscous flow problems, Journal of Computational Physics 2 (1967) 12 – 26.