Advancements in Numerical Modeling: High-Order Methods for Fractional Initial Value Problems and Meshfree Solvers for Partial Differential Equations

by

Andrew P. Lawrence

B.S., United States Air Force Academy, 2016

M.S., Air Force Institute of Technology, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2024

Committee Members:

Bengt Fornberg, Chair

Natasha Flyer

Ian Grooms

Mark Hoefer

Cècile Piret

Lawrence, Andrew P. (Ph.D., Applied Mathematics)

Advancements in Numerical Modeling: High-Order Methods for Fractional Initial Value Problems and Meshfree Solvers for Partial Differential Equations

Thesis directed by Prof. Bengt Fornberg

This dissertation presents four topics which broadly fall into the category of computational mathematics. A method is described for performing quadrature with up to 10th order accuracy for functions when one or both end points of the integration interval do not coincide with any of the equispaced grid points. This method can be utilized, for example, when functions to be integrated feature discontinuities at locations that can be separately determined (at or in between the equispaced grid points). The same underlying quadrature routine is then used to develop a solver for initial value problems on fractional differential equations. Order 6 convergence is achieved for problems with minimal smoothness assumptions. An algorithm which efficiently subsamples quasi-uniform, variable density node sets is then presented. This subsampling algorithm is able to preserve the local densities of the original node set. Finally a meshfree PDE solver is presented which utilizes the subsampling algorithm to implement a multilevel solver with RBF-FD. Efficiencies in the algorithm are gained through domain decomposition techniques.

Dedication

To the Way and the Truth and the Life, John 14:6

Acknowledgements

I would like to thank so many people for their support during my PhD. I'd like to begin by thanking my advisor and mentor Dr. Bengt Fornberg without whom I would not be here: thank you for countless words of wisdom and mathematical insights, it has truly been an honor to work with you! Next, I would like to thank my collaborator Dr. Morten Nielsen: thank you for treating me like a peer and hosting me in Denmark for a very productive month. I would also like to thank Dr. Cècile Piret for patiently working with me through difficult questions over the limited medium of email. I would like to thank Dr. Nancy Rodriguez for her incredible support during my struggles through analysis during the first year of my degree. I would also like to thank Gabriella Kirkley for her positivity and support.

I would also like to extend thanks to all of my peers in APPM who helped me brainstorm when I needed help with math and who provided distractions when I needed help with stepping away from math. Thanks to Jacob for being my gym buddy and so much more, my original cohort for helping with my transition back into school, and for Kevin, Cait, Sara, and Killian for their friendship.

Finally I would like to thank my friends and family, specifically my mother and father, my sister, Renee, my community at Cornerstone, and my friends in Boulder and beyond. Thank you for your support during the highs and the lows. Thank you for the sacrifices you made to allow me the time and space I needed to finish this PhD.

Disclaimer

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or U.S. government.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

This work presents various advancements to the field of computational mathematics. The application areas for these works remain open as the methods themselves are designed so as to be widely applicable. The primary themes of these works are to enable, derive, or demonstrate methods which are able to achieve high orders of accuracy. Additionally, in most cases, computational efficiency is also prioritized.

Chapter 2 describes an end corrected quadrature scheme for piecewise-smooth discontinuous functions, given equispaced data. An enhanced trapezoidal rule based on Gregory's method of integration has been shown to achieve high orders of accuracy for functions on equispaced grids [35]. This chapter serves to extend that method to cases where at least one limit of integration does not align with the equispaced grid. This case is equivalent to that of quadrature on discontinuous functions. The work contained in this chapter has been published in [39].

Chapter 3 presents a method for solving initial value problems for fractional differential equations of the Caputo type. Fractional derivatives are most commonly defined in terms of integrals, thereby making them non-local operators, unlike their integer-order counterparts. In particular, the idea behind Gregory's method, that of an end corrected quadrature rule, is applied to the integral in the definition of the Caputo derivative [41] to discretize the fractional derivative with high orders of accuracy. This discretization is then incorporated into a linear multi-step method to advance in time. This method is referred to as the fractional backward quadrature formula in comparison to the fractional backward difference formula which can be found in the literature [22, 59]. Some

optimization and interpolation methods are implemented to kick-start the method. This work is demonstrated to achieve $\mathcal{O}(h^6)$ accuracy which is much better than almost anything else in the literature. Also, it is much computationally faster than the few other high order methods. This work is not yet published.

The algorithm developed in Chapter 4 serves to coarsen a variable density, quasi-uniform node set while preserving the qualities of the initial node set. In the context of executing a geometric multilevel method on a meshfree domain, it is important that the geometric properties of the initial fine node set are maintained at each coarse node level. That importance is further emphasized in the context of meshfree partial differential equation (PDE) solvers such as radial basis function-generated finite differences (RBF-FD) which require very robust node set quality. Therefore, a method was developed to properly subsample node sets for this purpose. It is compared with other methods according to heuristics, two novel measures of comparative local regularity, and computational time. It is demonstrated to equal or outperform the other methods in all three categories. Further, this method is demonstrated to enable a meshfree multilevel elliptic PDE solver to achieve the expected computational cost $\mathcal{O}(N)$ for a multilevel method while still achieving the expected convergence rates for the RBF-FD sovler. This work is published in [57].

Finally, Chapter 5 introduces a robust meshfree multilevel PDE solver for unfitted node sets. Four primary methods are cooperatively combined in this chapter. The meshfree RBF-FD solver and the multilevel method were previously combined in Chapter 4 with success. This chapter introduces Lagrange multiplier enforced boundary conditions, allowing for the interior node sets to remain unfitted to the boundary. This is particularly advantageous in systems where the boundary might be evolving in time, as the interior node set may then be defined as a subset of a much larger background node set. Points from the background node set may then be included or excluded according to the evolving boundary without having to fit the nodes at each time step. Similarly, in the context of moving boundaries, the multilevel operators must change along any boundaries. To circumvent this cost, a domain decomposition method is introduced to allow the multilevel method to operate on an unchanging core node set while the near-boundary domains are solved by some

direct method. The results of applying this method to time-independent PDEs demonstrate the $\mathcal{O}(N)$ cost of the multilevel method while maintaining the RBF-FD convergence rates. Although the case of moving boundaries is not tested in this chapter, the examples which are included serve to demonstrate the viability of this combined method for future applications to the time-dependent case. The work in this chapter is not yet published.

Chapter 2

Enhanced Trapezoidal Rule for Discontinuous Functions

2.1 Introduction

In computational mathematics, numerical integration plays a crucial role. Most high-order quadrature rules are reliant on the ability to evaluate the integrand at mathematically optimal locations, such as for Gaussian quadrature-type methods. However, in most non-trivial applications, quadrature is a secondary task, and the expectation that data is available with such specificity is unrealistic. More common is the case when function values are available at equispaced locations. The trapezoidal rule (TR) is well-known for providing highly accurate approximations for smooth periodic functions, but its accuracy drops to second-order for non-periodic cases. The Gregory approach for end corrections can elevate the accuracy order significantly as shown in [42, 35].

In a further effort to acknowledge practical real-world data and applications, the desired integration interval may not align with the interval on which the data is provided. This is most easily understood as the result of equispaced data collected for a function with a discontinuity present, but not aligned with the sampling of the data. This can then be treated as two integration intervals, each of which has an end point not aligned with the data.

This chapter serves as a summary of the work published in [39], in collaboration with Dr. Fornberg, which extends the trapezoidal rule to achieve high accuracy in cases where integration interval endpoints do not align with equispaced data points.

2.2 Background and Related Work

2.2.1 Composite Trapezoidal Rule

For smooth functions with integration intervals coinciding with grid points, the TR is given by:

$$\int_{x_0}^{x_N} f(x)\,dx \approx h\sum_{k=0}^{N} f(x_k) - \frac{h}{2}\left[f(x_0)+f(x_N)\right], \tag{2.1}$$

where the nodes x_k are spaced by a distance h. Around the year 1740, Euler and Maclaurin independently discovered an infinite sequence of further correction terms:

$$\begin{aligned}\int_{x_0}^{x_N} f(x)\;dx \approx h\sum_{k=0}^{N} f(x_k) - \frac{h}{2}[f(x_0)+f(x_N)] \\ + \frac{h^2}{12}\left[f^{(1)}(x_0)-f^{(1)}(x_N)\right] - \frac{h^4}{720}\left[f^{(3)}(x_0)-f^{(3)}(x_N)\right] \\ + \frac{h^6}{30240}\left[f^{(5)}(x_0)-f^{(5)}(x_N)\right] - \frac{h^8}{1209600}\left[f^{(7)}(x_0)-f^{(7)}(x_N)\right] + \cdots .\end{aligned}$$

These demonstrate that the dominant errors from a TR-like approximation to an integral arise due to end effects.

2.2.2 The Gregory Method

The Gregory method, developed by James Gregory in the 17th century [49] and summarized in Section 2.2.2, provides end corrections to enhance the accuracy order to $\mathcal{O}(h^9)$ before negative weights emerge. The idea behind Gregory's method is that, given the crude quadrature scheme

$$\int_{x_0}^{x_N} f(x)\;dx \approx h\sum_{k=0}^{N} f(x_k) \tag{2.2}$$

adjust $p-1$ weights at each end of the interval to obtain a scheme of accuracy order p. The TR represents the $p=2$ case.

Isolating the lower limit of integration, such a formulation would look like

$$\int_{0}^{\infty} f(x)\;dx \approx \left(\sum_{0}^{\infty} f(k)\right) + \left[b_0\Delta^0 + b_1\Delta^1 + b_2\Delta^2 + ...\right] f(0) \tag{2.3}$$

and to include only a few leading terms in the second sum. The operator Δ denotes, here, the forward differences, i.e., $\Delta f(k) = f(k+1) - f(k)$ such that

$$\begin{aligned}
\Delta^0 f(0) &= f(0) \\
\Delta^1 f(0) &= f(1) - f(0) \\
\Delta^2 f(0) &= f(2) - 2f(1) + f(0) \\
\Delta^3 f(0) &= f(3) - 3f(2) + 3f(1) - f(0) \\
&\vdots \quad \vdots
\end{aligned} \tag{2.4}$$

Substituting $f(x) = e^{-zx}, z \in \mathbb{C}$ into (2.3) gives

$$\frac{1}{z} = \frac{1}{1-e^{-z}} + \left[b_0 - b_1(1-e^{-z}) + b_2(1-e^{-z})^2 - b_3(1-e^{-z})^3 + ...\right].$$

The further substitution of $u = (1-e^{-z})$ or, equivalently, $z = -\log(1-u)$ gives

$$\frac{1}{\log(1-u)} + \frac{1}{u} = -b_0 + b_1 u - b_2 u^2 + b_3 u^3 - ... \tag{2.5}$$

The coefficients b_k can be calculated based on the Taylor expansion of $\log(1-u)^{-1}$, as shown in Table 2.1.

b_0	b_1	b_2	b_3	b_4	b_5	b_6
$-\frac{1}{2}$	$\frac{1}{12}$	$-\frac{1}{24}$	$\frac{19}{720}$	$-\frac{3}{160}$	$\frac{863}{60480}$	$-\frac{275}{24192}$
-0.5	0.0833	-0.0417	0.0246	-0.0187	0.0138	-0.0114

Table 2.1: The coefficients b_k from (2.3). Note the slow decay and alternating sign of the coefficients.

Equation, (2.3) can be manipulated to be in terms of correction weights w_k or corrections to weights of 1, d_k

$$\begin{aligned}
\int_0^\infty f(x)\, dx &\approx \left(\sum_0^\infty f(k)\right) + \left[b_0\Delta^0 + b_1\Delta^1 + b_2\Delta^2 + ...\right] f(0) \\
&\approx \sum_{k=0}^{\infty} w_k f(k) \\
&\approx \sum_{k=0}^{\infty} 1 \cdot f(k) + \sum_{k=0}^{n} d_k f(k)
\end{aligned}$$

where $w_k = 1 + d_k$ and $w_k = 1$ for $k > n$. For stability, it should hold that $w_k \geq 0$ or $d_k \geq -1$. Negative weights can lead to cancellation errors in the numerical evaluation, which may result in a lower practical accuracy of the quadrature.

Using the first $n+1$ coefficients $b_0, b_1, ..., b_n$, it follows from (2.3), (2.4) that the corrections $d_0, d_1, ..., d_n$ to the leading $n+1$ weights are obtained by

$$\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ \\ \\ d_n \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & \cdots \\ & 1 & -2 & 3 & \cdots \\ & & 1 & -3 & \cdots \\ & & & 1 & \cdots \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \\ \\ b_n \end{bmatrix}. \tag{2.6}$$

Explicit forms of the d_k weights can be found in Table ??. Since the b_k sequence is alternating in sign and is only slowly decreasing, the rapid growth in the Pascal triangle entries in the coefficient matrix of (2.6) causes the central entries in the d_k sequence to grow rapidly with n (and make the Gregory method impractical for higher accuracy orders). Note that the middle entry for $p = 10$ shows the first instance of a correction < -1, i.e., a negative weight $w_k = 1 + d_k$.

Table 2.2: Gregory corrections weights d_k up to $p = 10$.

Order p	Corrections d_k								
2	$-\frac{1}{2}$								
3	$-\frac{7}{12}$	$\frac{1}{12}$							
4	$-\frac{5}{8}$	$\frac{1}{6}$	$-\frac{1}{24}$						
5	$-\frac{469}{720}$	$\frac{59}{240}$	$-\frac{29}{240}$	$\frac{19}{720}$					
6	$-\frac{193}{288}$	$\frac{77}{240}$	$-\frac{7}{30}$	$\frac{73}{720}$	$-\frac{3}{160}$				
7	$-\frac{41393}{60480}$	$\frac{23719}{60480}$	$-\frac{11371}{30240}$	$\frac{7381}{30240}$	$-\frac{5449}{60480}$	$\frac{863}{60480}$			
8	$-\frac{12023}{17280}$	$\frac{6961}{15120}$	$-\frac{66109}{120960}$	$\frac{33}{70}$	$-\frac{31523}{120960}$	$\frac{1247}{15120}$	$-\frac{275}{24192}$		
9	$-\frac{2558783}{3628800}$	$\frac{1908311}{3628800}$	$-\frac{299587}{403200}$	$\frac{115963}{145152}$	$-\frac{426809}{725760}$	$\frac{112477}{403200}$	$-\frac{278921}{3628800}$	$\frac{33953}{3628800}$	
10	$-\frac{63887}{89600}$	$\frac{427487}{725760}$	$-\frac{3498217}{3628800}$	$\frac{500327}{403200}$	$-\frac{6467}{5670}$	$\frac{2616161}{3628800}$	$-\frac{24019}{80640}$	$\frac{263077}{3628800}$	$-\frac{8183}{1036800}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

2.2.3 Enhanced Gregory Method

The order of accuracy can be increased up to $\mathcal{O}(h^{20})$ when the solution is optimized to an under-determined form of the system as proven in [35]. That derivation is summarized here for background. The enhanced methodology incorporates free parameters to optimize the weight corrections, reducing their magnitudes and preventing ill-conditioning.

Inverting the coefficient matrix in (2.6) and extending the $\boldsymbol{d}$ vector (and the coefficient matrix correspondingly) gives

$$\left[\begin{array}{ccccc|ccc} 1 & 1 & 1 & 1 & \cdots & \cdots & \cdots & \cdots \\ & 1 & 2 & 3 & \cdots & \cdots & \cdots & \cdots \\ & & 1 & 3 & \cdots & \cdots & \cdots & \cdots \\ & & & 1 & \cdots & \cdots & \cdots & \cdots \\ & & & & \ddots & \cdots & \cdots & \cdots \end{array}\right] \left[\begin{array}{c} d_0 \\ d_1 \\ \vdots \\ d_n \\ \hline d_{n+1} \\ \vdots \\ d_N \end{array}\right] = \left[\begin{array}{c} b_0 \\ b_1 \\ \vdots \\ \\ b_n \end{array}\right] \tag{2.7}$$

which is an under-determined system. This system can then be optimized to reduce the magnitude of the d_k while keeping $d_k \geq -1$.

2.3 Extension for Non-aligned Endpoints

For intervals with endpoints not aligned with grid points, this section introduces a method based on that of [35] and Section 2.2.3 to achieve up to $O(h^{10})$ accuracy.

Consider

$$\int_{\xi}^{\infty} f(x)\, dx \approx \left(\sum_{0}^{\infty} f(k)\right) + \left[b_0(\xi)\Delta^0 + b_1(\xi)\Delta^1 + b_2(\xi)\Delta^2 + ...\right] f(0)$$

where $-1 < \xi \leq 0$. This is the case where the interval of integration is misaligned with the equispaced grid at one end point. The resulting expansion coefficients $b_k(\xi)$ now depend on the offset value ξ.

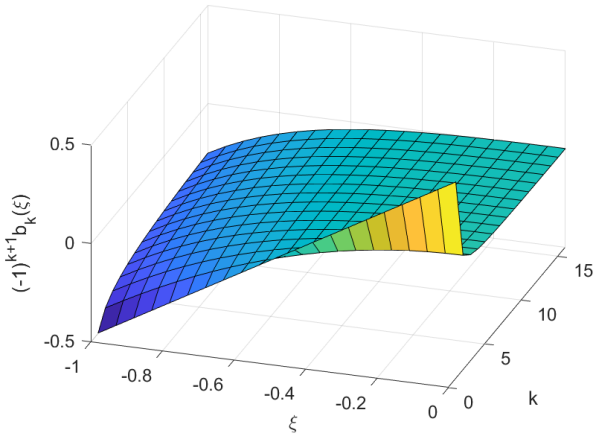


Figure 2.1: Figure 2.1: $(-1)^{k+1}b_k(\xi)$

The same steps which led to (2.5), now instead give

$$\frac{(1-w)^{\xi}}{\log(1-w)} + \frac{1}{w} = -b_0(\xi) + b_1(\xi)w - b_2(\xi)w^2 + b_3(\xi)w^3 - \cdots$$

and the $b_k(\xi)$ coefficients can be solved for, as seen in Table **??**. Figure 2.1 shows $(-1)^{k+1}b_k(\xi)$ as a function of $-1 \leq \xi \leq 0$ and $k = 0, 1, 2, ..., 15$.

From result in [83] it follows that

$$b_k(0) = -\int_0^1 \binom{s}{k+1}\, ds$$
$$b_k(\xi) = b_k(0) + \int_0^{-\xi} \binom{-s}{k}\, ds.$$

This further implies that $b_{k+1}(0) = b_k(-1) + b_{k+1}(-1)$ and that, for large k,

$$b_k(0) \sim \frac{(-1)^{k+1}}{k(\log k)^2}$$
$$b_k(-1) \sim \frac{(-1)^k}{\log k}.$$

2.4 Quadrature of Discontinuous Functions

2.4.1 Numerical calculation of weight sets

Using the same method as in [35] and Section 2.2.3 the weights $d_k(\xi)$ can be calculated for specific values of n, N, and ξ. It is only for certain combinations of n and N that the optimization

Table 2.3: Forward difference coefficients $b_k(\xi)$ for the non-aligned Gregory method.

$b_k(\xi)$	$\xi = 0$	$\xi \neq 0$
$b_0(\xi)$	$-\frac{1}{2}$	$b_0(0) + (-\xi)$
$b_1(\xi)$	$\frac{1}{12}$	$b_1(0) + \left(-\frac{1}{2}\xi^2\right)$
$b_2(\xi)$	$-\frac{1}{24}$	$b_2(0) + \left(\frac{1}{4}\xi^2 - \frac{1}{6}\xi^3\right)$
$b_3(\xi)$	$\frac{19}{720}$	$b_3(0) + \left(-\frac{1}{6}\xi^2 + \frac{1}{6}\xi^3 - \frac{1}{24}\xi^4\right)$
$b_4(\xi)$	$-\frac{3}{160}$	$b_4(0) + \left(\frac{1}{8}\xi^2 - \frac{11}{72}\xi^3 + \frac{1}{16}\xi^4 - \frac{1}{120}\xi^5\right)$

finds valid corrections for all ξ in the required range $-1 < \xi \leq 0$. Such combinations include $n = 4, N = 8$ and $n = 8, N = 20$. In the case of the former, the corrections $d_k(\xi)$ are visualized in Figure 2.2. The optimization is set to minimize $\sum_{k=0}^{N} d_k(\xi)^2 (k+1)^8$ while enforcing $d_k(\xi) \geq -1$[1]. The order of accuracy $p = 10$ appears to be as high as this procedure works across the full interval $-1 < \xi \leq 0$.

2.4.2 Numerical Tests

Consider the case of two functions meeting discontinuously at some non-grid point location within the interval instead of a single smooth function with integration endpoints not coinciding with grid points.

The rates of convergence as $h \to 0$ are jagged since the relative alignment of the discontinuity with its nearby grid points varies irregularly with h. The errors nevertheless in all cases follow their theoretically predicted rates.

In all cases, as h decreases, the accuracy reaches and then remains at the level of machine rounding errors, i.e., around 10^{-16}. Although optimization algorithms often do not reach to this level of accuracy, the accuracy conditions and the non-negativity constraints in the present method are implemented precisely, and optimization only enters for the secondary task of ensuring the decay of the weight corrections d_k.

[1]The power 8 in the sum is very arbitrary; low numbers do not force the correction weights to smoothly approach zero, while much higher numbers may cause difficulties for the optimization algorithm to converge

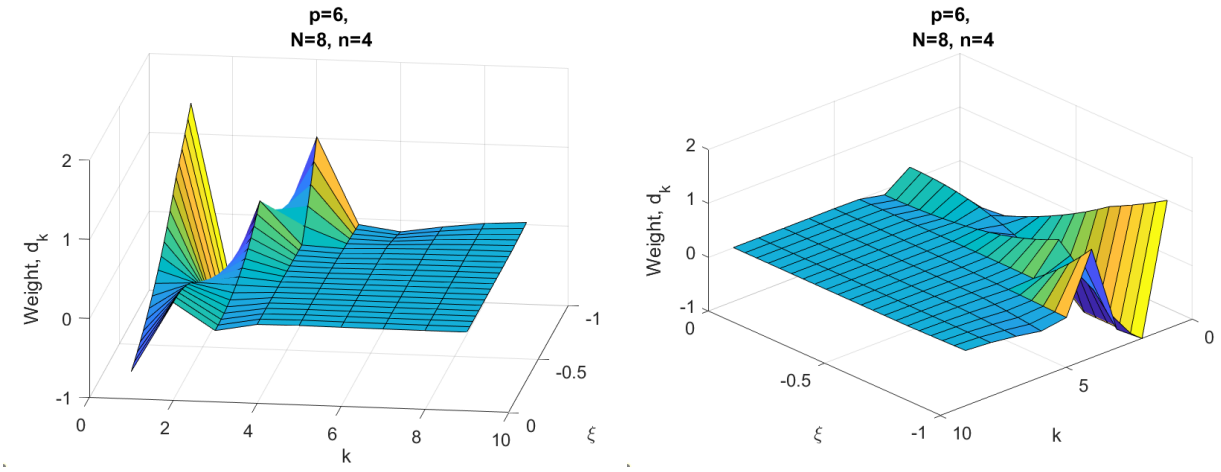


Figure 2.2: Figure 2.2: Corrections $d_k(\xi)$ in the case of $n = 4, N = 8$, seen from two perspectives in the k, ξ-plane

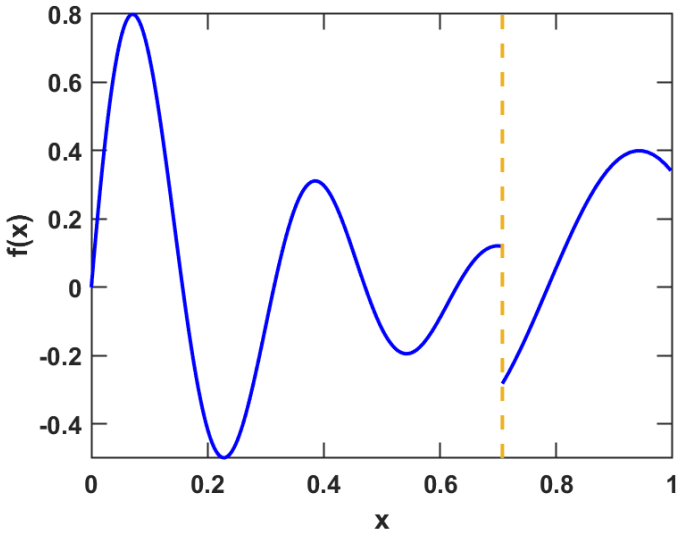


Figure 2.3: The test function (2.8), with a discontinuity at $x = 1/\sqrt{2}$

Consider the task of approximating

$$\int_0^1 f(x)\, dx \quad \text{with} \quad f(x) = \begin{cases} e^{-3x}\sin(20x), & 0 \le x < 1/\sqrt{2} \\ -\frac{2}{5}\cos(10x), & 1/\sqrt{2} \le x \le 1 \end{cases} \tag{2.8}$$

using equispaced nodes over $[0, 1]$. This test function is shown in Figure 2.3. Because the discontinuity location $1/\sqrt{2}$ is irrational, it will not coincide with a grid point at any level of refinement. Figure 2.4 shows the experimental rate of convergence using the classical trapezoidal rule, the present method with $n = 4, N = 8$, and the present method with $n = 8, N = 20$. Note that the trapezoidal rule is only able to achieve $\mathcal{O}(h)$ due to the discontinuity present in the function. As theoretically determined, the convergence rates of the two cases of the present method are given by $p = n + 2$, i.e., they are $\mathcal{O}(h^6)$ and $\mathcal{O}(h^{10})$, respectively.

2.5 Conclusion

This study extends the applicability of the enhanced trapezoidal rule for numerical integration from [35] to cases where integration endpoints do not align with equispaced grid points or discontinuous functions are to be integrated, making it suitable for practical applications involving

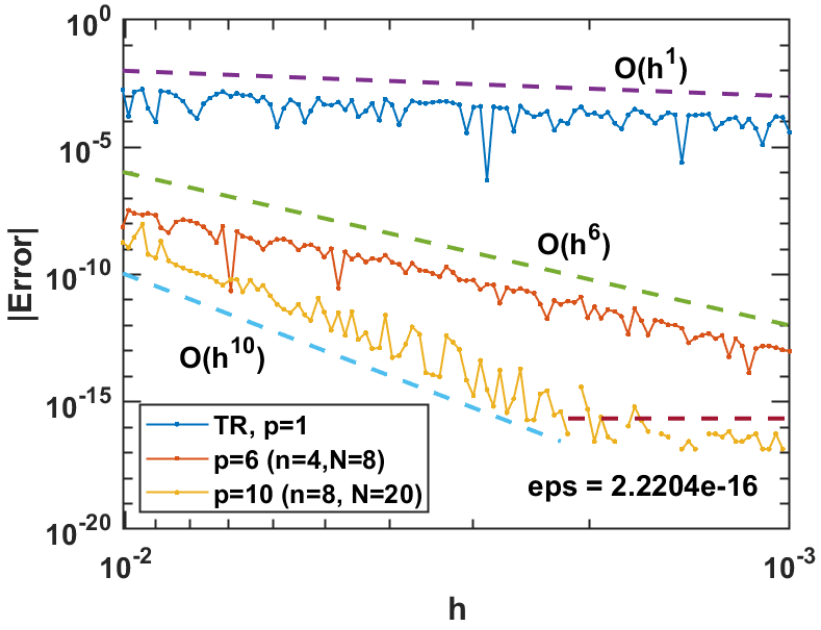


Figure 2.4: Error as a function of h for the test problem (2.8).

discontinuous functions. The proposed method readily achieves high accuracy (up to $\mathcal{O}(h^{10})$) and non-negative weights. The published version of this study [39] also includes comparisons against other recently published methods for the same task, showing superior convergence rates and numerical robustness.

Chapter 3

High-Order Numerical Solver for Fractional Order Initial Value Problems

The work in this chapter was perfomed in collaboration with Drs. Bengt Fornberg and Cècile Piret. A journal manuscript is being prepared for submission.

3.1 Introduction

The classical, integer-order derivative is an iconic and indispensable tool in the field of calculus. In contrast, derivatives of fractional-order are much less known. Given the lack of renown for fractional calculus, it may be surprising to some that it is nearly as old as its classical counterpart. Discussion of fractional derivatives dates back to 1695 with a correspondence between l'Hôpital and Leibniz. It wasn't until 1823, however, that a comprehensive framework for fractional calculus, to include a first physical application, was presented by Abel [2].

A brief inquiry into the theory of fractional calculus reveals fundamental differences which may explain why this branch of calculus has been slower to gain popularity. In particular, while integer-order derivatives have a unique definition, their fractional-order counterparts do not. Numerous definitions have been proposed, each with different nuances and areas of applicability [64, 13, 74, 103]. Often, the non-integer order derivatives are defined as inverses of certain fractional order integrals. As such, fractional derivatives are not, in general, defined as local operators[1]. Rather, they depend on an integral between a base point and an evaluation point. The non-local character of these fractional-order derivatives allows fractional differential equations to be a very

[1]Local fractional order derivatives (with somewhat different properties) have been considered [1, 51, 54].

natural tool for modeling memory-dependent phenomena. This chapter will discuss numerical approximations to fractional ordinary differential equations (FODEs, as opposed to fractional partial differential equations) with initial conditions.

Extensive literature is nowadays available on many aspects (mathematical, application oriented, etc.) of fractional derivatives, see for example [21, 48, 86, 89]. Good, critical reviews of the current state of the field of numerical methods for solving FODEs are given in [46, 22, 11]. FODEs have found a place in many areas of application thanks to their inherent non-locality which can describe memory, time delay, and dilation among other things. A short list of applications includes materials science, economics, groundwater flow, and quantum mechanics. A brief survey in [26] describes just some of the many applications of FODEs and includes references to many more surveys on the applications of FODEs.

Initial value problems (IVPs) defined in terms of the Caputo derivative [13, 52] require initial conditions in terms of integer order derivatives, which are those most easily measured and supplied in real world scenarios[2]. The number of initial conditions depends on the order α of the FODE and is equal to $\lceil \alpha \rceil$. For the sake of applications to real world problems, only FODEs defined by the Caputo derivative will be considered:

$$ {}_a^C D_t^\alpha y(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{\frac{d^n}{d\tau^n} y(\tau)}{(t-\tau)^{\alpha+1-n}} \, d\tau, \quad n-1 < \alpha < n. \tag{3.1} $$

This chapter will also restrict its scope to the class of time-dependent FODEs which begin at $t = 0$ and for which $0 < \alpha < 1$. Therefore, the notation for and definition of the fractional derivative can be simplified as follows

$$ D^\alpha y(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{y'(\tau)}{(t-\tau)^\alpha} \, d\tau, \quad 0 < \alpha < 1. \tag{3.2} $$

Even more often than for integer order ordinary differential equations (ODEs), fractional order FODEs lack closed form solutions, necessitating numerical approaches. For the remainder of

[2]The other most common definition of the fractional derivative is the Riemann-Liouville definition [65], which requires fractional order initial conditions.

the chapter, only FODEs of the following type will be considered:

$$D^{\alpha}y(t) = f(t, y(t)), \tag{3.3}$$

$$y(0) = y_0 \tag{3.4}$$

with $0 < \alpha < 1$. One of the primary difficulties faced when numerically solving FODEs is that of a lack of smoothness of the true solution, specifically near time $t = 0$ [46, 97]. As illustrated in (3.28), the solution to some FODEs will have an expansion as $t \to 0$ of the form

$$y(t) = ct^{\nu} + o(t^{\nu}) \tag{3.5}$$

with $c \neq 0$ and $0 < \nu < 1$ non-integer. Such singularities in the solution may be present even when the source term ($f(t, y(t))$ in (3.3)) is in $C^{\infty}([0, 1])$ [97]. Therefore the integral in (3.2) is assumed to have a singularity of this unknown order $1 - \nu$ at its lower end and of known order α at its upper end.

As noted above, the integral in (3.2) is singular at both ends. As the singularity at the evaluation point (upper end) is of known type, it can be handled with a real-valued version of the method introduced in [41]. The unknown type at the base point (lower end) needs to be handled differently, however. The method presented here suppresses the base point singularity by introducing two changes of variables of the form $t = \eta(r)$ which are discussed further in Section 3.4. The key task that is addressed in this chapter is to handle both of these singularities accurately, in combination with numerically stepping forward in the t-direction.

Indirect methods for solving FODEs involve the application of the fractional integral operator to (3.3). These methods isolate the solution and return an equivalent Volterra integral equation [24]. The fractional integral of the source term can then be discretized to derive various numerical methods. The fractional linear multi-step method (FLMM) introduced and developed primarily by Lubich [71, 72, 70, 73] approximates solutions to fractional problems by means of convolution quadrature. FLMMs are theoretically able to achieve error convergence on the same order as the corresponding classical linear multi-step methods upon which they are based, given certain

properties are satisfied [60]. However, convolution weights for FLMMs are not known explicitly and must be computed. Such computations may not be trivial [69] and may require large, ill-conditioned systems to be solved [25].

Instead of convolution quadrature, a fractional quadrature scheme such as the left and right rectangular methods and the Newton-Cotes methods can be employed to discretize the fractional integral of the source term. For an interpolant of order $r \in \mathbb{N}$, the fractional Newton-Cotes formula requires $f(t, y(t)) \in C^{r+1}([0, t])$ and achieves an error estimate of $\mathcal{O}(h^{r+1})$ which is one order lower than the classical Newton-Cotes methods [60].

The popular fractional Adams method is a predictor-corrector scheme based on an approximation by the fractional trapezoidal rule [23, 24]. A simple and efficient modification of the standard fractional Adams method results in a convergence of $\mathcal{O}(h^2)$ [20]. Runge-Kutta methods, which are very popular for solving classical ODEs, have been generalized to the fractional problem [67]. Use of Runge-Kutta methods for FODEs, however, is uncommon due to the complicated conditions necessary to guarantee a desired convergence order. A two-step spline collocation method has been recently derived which achieves $\mathcal{O}(h^{2m})$ accuracy for $m \in \mathbb{N}$ [14]. Convergence is demonstrated to be robust and computational time as a function of digits of accuracy is shown to be competitive with a 2nd order FLMM, especially for high degrees of accuracy [15]. However, the computational complexity may require a significant amount of computational effort compared to simpler numerical methods. Additionally, in order to achieve the optimal convergence, analysis of the source term must be carefully completed.

Alternatively, the fractional derivative may be discretized directly as in the so-called Grünwald-Letnikov methods ($\mathcal{O}(h)$) [82, 21, 115, 90] or the fractional backwards difference formulas (FBDFs) [22, 59]. The basic FBDFs can be modified to overcome the common singular nature of the solutions to FODEs and very nearly achieve the convergence and stability of the corresponding classical formulas [69, 68, 70]. Unfortunately, such modified FBDFs suffer from severe ill-conditioning for certain fractional orders which can nullify any improvements to convergence and stability [25].

Another family of methods is based on replacing the integrand function ($y(\tau)$ in (3.2)) with

different interpolating polynomials. The popular L1 method [82, 60] is derived by approximating the integrand function with a linear interpolation. Though it is similar in spirit to the classical trapezoidal rule, the integrand function must be differentiated following the interpolation according to (3.2). The L1 method is of order $2-\alpha$ and can be extended to higher order interpolants. As the order of the method increases, however, so does the smoothness requirement placed on the solution, e.g. the method of order $3-\alpha$ requires that $y \in C^3[0,t]$ [19, 44, 62, 3, 76, 75, 12]. Spectral methods have been derived in which generalized polynomials (such as Jacobi polynomials) approximate the integrand function [61, 93, 50, 113, 114]. Radial basis functions have been used to solve fractional problems as well [85].

While the present method does not have a direct analog in classical numerical ODE theory, comparisons can be drawn between this method and the backward difference formula. However, instead of a finite difference scheme replacing the differential operator, the Caputo derivative is discretized by a quadrature scheme, as a result of it being defined in terms of an integral (3.2). Modified Gregory quadrature is used to discretize the integral in the Caputo derivative once integration by parts has manipulated the integrand to be in terms of the undifferentiated solution, $y(\tau)$. With step size h, the present method, which shall be hereafter referred to as the fractional backward quadrature formula (FBQF) is demonstrated to achieve a high convergence rate of $\mathcal{O}(h^6)$ for low smoothness constraints, $f \in C[0,t]$.

The outline of this chapter is as follows: the numerical evaluation of fractional derivatives is discussed in Section 3.2, a scheme for numerically stepping forward in time the solution to a FODE is given in Section 3.3, the handling of changes of variables to suppress singularities is given in Section 3.4, the full algorithm is outlined in Section 3.5, specific examples with numerical results are given in Section 3.6, and a discussion of the performance of the method is given in Section 3.7.

3.2 Numerical Evaluation of Fractional Derivatives

Consider the computational evaluation of $D^\alpha y(t)$ as defined in (3.2) and, as before, $0 < \alpha < 1$. A quadrature scheme for evaluating the integral in (3.2) in the complex plane using Gregory's method [35] is described in [41]. That scheme achieves high orders of accuracy for equispaced nodes which is necessary in the context of FODEs. The same numercial approximation for the Caputo derivative will be derived here on the real line.

Assuming that $y(t)$ is without singularities on the interval $[0, t]$, Gregory quadrature cannot be applied due to two things: a singularity at $\tau = t$ and the integrand being in terms of $y'(\tau)$ instead of $y(\tau)$. However, the concept of an end-corrected trapezoidal rule approximation, which is at the core of the Gregory method, can still be applied here.

Consider the task of approximating

$$I = \int_0^t \frac{y'(\tau)}{(t-\tau)^\alpha} d\tau \tag{3.6}$$

for $0 < \alpha < 1$ and $t > 0$. Next in order to isolate the singularity in the denominator of the integrand, the domain of integration $[0, t]$ is split into $[0, t-h]$ and $[t-h, t]$. Then applying integration by parts to the integral over $[0, t-h]$ results in

$$I = \underbrace{-\frac{y(0)}{t^\alpha}}_{\text{Base Point}} \underbrace{-\alpha \int_0^{t-h} \frac{y(\tau)}{(t-\tau)^{\alpha+1}} d\tau}_{\text{Interior}} + \underbrace{\frac{y(t-h)}{h^\alpha} + \int_{t-h}^t \frac{y'(\tau)}{(t-\tau)^\alpha} d\tau}_{\text{Evaluation Point}}. \tag{3.7}$$

A summation similar to the trapezoidal rule will be used to approximate the Interior integral:

$$-\alpha \int_0^{t-h} \frac{y(\tau)}{(t-\tau)^{\alpha+1}} d\tau \approx -\alpha h \sum_{k=1}^{\frac{t}{h}-1} \frac{y(kh)}{(t-kh)^{\alpha+1}} \tag{3.8}$$

and correction stencils will be applied to the function values at the upper and lower limit of integration to increase the accuracy of this approximation.

3.2.1 Lower End / Base Point

The weights in the base point correction stencil are obtained by applying the method of exponential test functions to the error in the approximation from (3.8). Consider the substitution $e^{\tau\xi} = \frac{y(\tau)}{(t-\tau)^{\alpha+1}}$ for Re $\xi < 0$ and let $t \to \infty$ so as to isolate the base point correction. Then

$$\begin{aligned} -\alpha \int_0^\infty e^{\tau\xi}\, d\tau + \alpha h \sum_{k=1}^{\infty} e^{kh\xi} &= -\alpha \frac{h}{2}\left(1 + \coth\frac{h\xi}{2}\right) + \alpha \frac{1}{\xi} \\ &= \alpha h \left(\frac{\zeta(0)}{0!} + \frac{\zeta(-1)}{1!}(h\xi)^1 + \frac{\zeta(-2)}{2!}(h\xi)^2 + \frac{\zeta(-3)}{3!}(h\xi)^3 + \dots\right), \end{aligned}$$

with ζ denoting the Riemann zeta function.

Given the powers of $h\xi$, and setting aside the coefficient αh, it is natural to seek weights which would apply to $e^{z_n\xi}$ in order to represent the above 'error':

$$\begin{aligned} \frac{\zeta(0)}{0!} + \frac{\zeta(-1)}{1!}(h\xi)^1 + \frac{\zeta(-2)}{2!}(h\xi)^2 + \frac{\zeta(-3)}{3!}(h\xi)^3 + \dots &= \sum_{i=0}^{N} w_i^{(\mathrm{bp})} e^{z_i\xi} \\ &= \sum_{i=0}^{N} w_i^{(\mathrm{bp})} \sum_{\ell=0}^{\infty} \frac{(z_i\xi)^\ell}{\ell!} \end{aligned}$$

where the coefficient $-h$ has been dropped for convenience. The equation above is equivalent to the following linear system

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ z_1 & z_2 & z_3 & \dots & z_N \\ z_1^2 & z_2^2 & z_3^2 & \dots & z_N^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{N-1} & z_2^{N-1} & z_3^{N-1} & \dots & z_N^{N-1} \end{bmatrix} \begin{bmatrix} w_1^{(\mathrm{bp})} \\ w_2^{(\mathrm{bp})} \\ w_3^{(\mathrm{bp})} \\ \vdots \\ w_N^{(\mathrm{bp})} \end{bmatrix} = \begin{bmatrix} \zeta(0) \\ h\zeta(-1) \\ h^2\zeta(-2) \\ \vdots \\ h^{N-1}\zeta(-N+1) \end{bmatrix} \tag{3.9}$$

where the weights are then applied to values of $\frac{y(\tau)}{(t-\tau)^{\alpha+1}}$ and the coefficient of αh must be reintroduced.

3.2.2 Upper End / Evaluation Point

The end point correction is derived by applying a change of variables to the Interior and Evaluation Point terms in (3.7) and calculating the approximation error. Consider the substitutions $\sigma = t - \tau$ and $\gamma(\sigma) = y(t - \sigma)$ such that (3.8) becomes

$$-\alpha \int_0^{t-h} \frac{\gamma(\sigma)}{\sigma^{\alpha+1}} d\sigma \approx -\alpha h \sum_{k=1}^{\frac{t}{h}-1} \frac{\gamma(kh)}{(kh)^{\alpha+1}}. \tag{3.10}$$

The weights for the evaluation point correction stencil can be derived by evaluating the approximation error from (3.10) with the Evaluation Point terms from (3.7) and using the method of exponential test functions, as before. This time, however, $\gamma(\sigma)$ is replaced with with $e^{\sigma\xi}$ for Re $\xi < 0$ and as $\sigma \to \infty$ the evaluation point is isolated. Then the approximation error in (3.10) becomes

$$\begin{aligned} -\alpha \int_h^\infty \frac{e^{\sigma\xi}}{\sigma^{\alpha+1}} d\sigma + \alpha h \sum_{k=1}^{\infty} \frac{e^{\xi kh}}{(kh)^{\alpha+1}} &= -\alpha \Bigg((-\xi)^\alpha \Gamma(-\alpha) + h^{-\alpha} \sum_{k=0}^{\infty} \frac{1}{k!(\alpha-k)} (h\xi)^k \\ &\qquad - (-\xi)^\alpha \Gamma(-\alpha) - h^{-\alpha} \sum_{k=0}^{\infty} \frac{\zeta(1+\alpha-k)}{k!} (h\xi)^k \Bigg) \\ &= -\alpha h^{-\alpha} \sum_{k=0}^{\infty} \frac{1}{k!(\alpha-k)} (h\xi)^k + \alpha h^{-\alpha} \sum_{k=0}^{\infty} \frac{\zeta(1+\alpha-k)}{k!} (h\xi)^k \end{aligned} \tag{3.11}$$

and the Evaluation Point terms from (3.7) become

$$\frac{e^{h\xi}}{h^\alpha} - \int_0^h \frac{\xi e^{\xi\sigma}}{\sigma^\alpha} \, d\sigma = \alpha h^{-\alpha} \sum_{k=1}^{\infty} \frac{1}{k!(\alpha-k)} (h\xi)^k. \tag{3.12}$$

Note that the sum (3.12) exactly cancels the first sum in (3.11).

As before, setting aside the coefficient $\alpha h^{-\alpha}$, the powers of $h\xi$ are naturally represented by an expansion of $e^{z_j \xi}$. The above error can then be expressed as a weighted combination of $e^{z_j \xi}$:

$$\begin{aligned} \zeta(1+\alpha) + \frac{h\xi}{1!}\zeta(\alpha) + \frac{(h\xi)^2}{2!}\zeta(\alpha-1) + \frac{(h\xi)^3}{3!}\zeta(\alpha-2) + \ldots &= \sum_{j=0}^{M} w_j^{(\mathrm{ep})} e^{z_j \xi} \\ &= \sum_{j=0}^{M} w_j^{(\mathrm{ep})} \sum_{\ell=0}^{\infty} \frac{(-z_j\xi)^\ell}{\ell!} \end{aligned}$$

which is equivalent to the following linear system

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ z_1 & z_2 & z_3 & \dots & z_M \\ z_1^2 & z_2^2 & z_3^2 & \dots & z_M^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_1^{M-1} & z_2^{M-1} & z_3^{M-1} & \dots & z_M^{M-1} \end{bmatrix} \begin{bmatrix} w_1^{(\mathrm{ep})} \\ w_2^{(\mathrm{ep})} \\ w_3^{(\mathrm{ep})} \\ \vdots \\ w_M^{(\mathrm{ep})} \end{bmatrix} = \begin{bmatrix} \zeta(\alpha+1) \\ h\zeta(\alpha+0) \\ h^2\zeta(\alpha-1) \\ \vdots \\ h^{M-1}\zeta(\alpha-M+2) \end{bmatrix} \tag{3.13}$$

where the weights are then applied to values of $y(\tau)$ and the coefficient of $\alpha h^{-\alpha}$ is reintroduced.

3.2.3 Assembled Stencils

Combining the trapezoidal rule-like approximation (3.8) with the weighted correction stencils from Sections 3.2.1 and 3.2.2 for the base and evaluation points respectively, the numerical scheme to calculate the integral in (3.2) is as follows:

$$\int_0^t \frac{y'(\tau)}{(t-\tau)^\alpha}\, d\tau \approx -\frac{y(0)}{t^\alpha} - \alpha \left(h \sum_{k=1}^{t/h-1} \frac{y(kh)}{(t-kh)^{\alpha+1}} - h \sum_{i=0}^{N} w_i^{(\mathrm{bp})} \frac{y(s_i)}{(t-s_i)^{\alpha+1}} - h^{-\alpha} \sum_{j=0}^{M} w_j^{(\mathrm{ep})} y(\sigma_j) \right) \tag{3.14}$$

where $w_i^{(\mathrm{bp})}$ and $w_j^{(\mathrm{ep})}$ are the weights at the base point and the evaluation point, respectively, and $N, M > 0$ are the number of nodes in the base and evaluation stencils, respectively. The locations of the nodes in the base and evaluation stencils are not restricted to the interior of the interval. The values of s_i, σ_j may extend left of $\tau = 0$ and right of $\tau = t$. However, in order to avoid the singularity in the base point correction, it must hold that $s_i < t$ for $i = 0, ..., N$.

In order to achieve high accuracy with this method, the interior nodes for the end point stencils are chosen to be equispaced nodes. This choice not only provides high accuracy but it also aligns conveniently with the trapezoidal rule-like terms and a natural stepping scheme in time.

3.3 Fractional Backwards Quadrature Formula

Equipped with a method for numerically evaluating Caputo fractional derivatives, the problem of solving fractional differential equations can be addressed. A fractional linear multistep method similar to the FBDF is used to generate a nonlinear equation in terms of y_{n+1}. However, in contrast to the FBDF method, the FBQF method discretizes the integral within the Caputo derivative rather than the derivative itself.

Assume each end correction stencil is one-sided toward the interior with $M+1$ points such that $i, j = 0, ..., M$. Further assume that the values $y(ih)$ for $i = 0, 1, ..., M$ are given. These values will be initialized by an optimization routine as described in Section 3.5.1.1 or by interpolation as described in Section 3.5.2.1.

3.3.1 Implicit Method

Consider the equation

$$D^{\alpha} y(t_{n+1}) = f(t_{n+1}, y(t_{n+1})) \tag{3.15}$$

where $t_{n+1} = (n+1)h, h > 0$. Then, using the numerical scheme in (3.14) to replace the differentiation term gives

$$\int_0^{t_n+h} \frac{y'(\tau)}{(t_n+h-\tau)^{\alpha}}\, d\tau \approx -\frac{y(0)}{(t_n+h)^{\alpha}} - \alpha\Bigg(h \sum_{k=1}^{(t_n+h)/h-1} \frac{y(kh)}{(t_n+h-kh)^{\alpha+1}} - h\sum_{i=0}^{M} w_i^{(\mathrm{bp})} \frac{y(ih)}{(t_n+h-ih)^{\alpha+1}} - h^{-\alpha}\sum_{j=0}^{M} w_j^{(\mathrm{ep})} y(t_n+h-jh)\Bigg) \tag{3.16}$$

which simplifies to

$$\int_0^{t_n+h} \frac{y'(\tau)}{(t_n+h-\tau)^{\alpha}}\, d\tau \approx S^{\alpha}_{t_n+h} + \alpha h^{-\alpha} w_0^{(\mathrm{ep})} y(t_n+h)$$

where

$$S^{\alpha}_{t_n+h} = -\frac{y(0)}{(t_n+h)^{\alpha}} - \alpha\Bigg(\quad h \sum_{k=1}^{(t_n+h)/h-1} \frac{y(kh)}{(t_n+h-kh)^{\alpha+1}} - h\sum_{i=0}^{M} w_i^{(\mathrm{bp})} \frac{y(ih)}{(t_n+h-ih)^{\alpha+1}} - h^{-\alpha}\sum_{j=1}^{M} w_j^{(\mathrm{ep})} y(t_n+h-jh)\Bigg)$$

is made up entirely of known values of y such that (3.15) can now be written as

$$\frac{1}{\Gamma(1-\alpha)}\left(S^{\alpha}_{t_n+h} + \alpha h^{-\alpha} w_0^{(\mathrm{ep})} y(t_n+h)\right) = f(t_n+h, y(t_n+h)) \tag{3.17}$$

which can be solved with a standard nonlinear solver, such as Newton's method.

3.3.2 Newton's Method

Newton's method is used to find the root of

$$F(y_{n+1}) = \frac{1}{\Gamma(1-\alpha)}\left(S^{\alpha}_{t_n+h} + \alpha h^{-\alpha} w_0^{(\mathrm{ep})} y_{n+1}\right) - f(t_{n+1}, y_{n+1}) = 0.$$

where $y_{n+1} = y(t_n+h)$ and has iterates

$$y_{n+1}^{[0]} = y_n$$

$$y_{n+1}^{[i+1]} = y_{n+1}^{[i]} - \frac{F\left(y_{n+1}^{[i]}\right)}{F'\left(y_{n+1}^{[i]}\right)}$$

for which we need F':

$$F'(y_{n+1}) = \frac{\alpha h^{-\alpha} w_0^{(\mathrm{ep})}}{\Gamma(1-\alpha)} - \frac{d}{dy} f(t,y)\Big|_{y=y_{n+1}, t=t_{n+1}}$$

which requires the derivative of f with respect to y be numerically available.

Convergence criteria for Newton's Method in this scheme must satisfy both

$$|y_{n+1} - y_n| \geq |y_n - y_{n-1}|$$

and

$$|y_{n+1} - y_n| < tol$$

where $tol = 10^{-13}$ might be used in practice.

3.4 Changes of Variables

The quadrature scheme in (3.14) used to numerically evaluate the fractional derivatives and construct the fractional solver depends on the function being smooth, even at $\tau = 0$. As discussed in the introduction, two changes of variables are introduced to overcome the singularity that is typically present at the base point $t = 0$, inducing a smooth function on which the fractional solver can operate. The first change of variables will be of a monomial type, $\eta(\rho) = \rho^k$. The second change of variables will be of an exponential type, $\phi(\theta) = \theta e^{-\beta/\theta}$, $\beta > 0$.

For an arbitrary change of variables $\tau = \eta(\rho)$ on the domain $\tau \in [0, t]$ let $\rho \in [0, r]$ where $t = \eta(r)$. Let the changed solution be $u(\rho) = y(\eta(\rho))$. The error calculation for evaluating the fractional derivative with an arbitrary change of variables follows the derivation in Section 3.2 and results in the following

$$\begin{aligned}\int_0^t \frac{y'(\tau)}{(t-\tau)^\alpha}\, d\tau &= \int_0^r \frac{u'(\rho)}{(t-\eta(\rho))^\alpha}\, d\rho \\ &\approx -\frac{u(0)}{t^\alpha} - \alpha\left(h\sum_{n=1}^{r/h-1} \frac{u(nh)\kappa(nh)}{(r-nh)^{\alpha+1}} - h\sum_{i=0}^{M} w_i^{(\mathrm{bp})} \frac{u(\rho_i)\kappa(\rho_i)}{(r-\rho_i)^{\alpha+1}} - h^{-\alpha}\sum_{j=0}^{M} w_j^{(\mathrm{ep})} u(\sigma_j)\kappa(\sigma_j)\right)\end{aligned} \tag{3.18}$$

which is similar to (3.14). The difference is that the (now transformed) solution $u(\rho)$ is multiplied by a helper variable which is necessary due to the integration by parts:

$$\kappa(\rho) = \frac{\eta'(\rho)(r-\rho)^{(\alpha+1)}}{(t-\eta(\rho))^{(\alpha+1)}}. \tag{3.19}$$

Note, the change of variables may render the magnitude of the base point correction negligible, especially for $t \gg 0$.

3.5 Algorithm

The full algorithm is primarily composed of the FBQF method described in Section 3.3 and can be found on the authors GitHub [55]. The additional methods presented in this section are designed to suppress any singularities at the base point while also enabling the Mth order FBQF method to start given only $y(0) = y_0$ when $M + 1$ known function values are required (Section 3.5.1) and to step forward in time with equispaced steps as $t \to \infty$ (Section 3.5.2).

3.5.1 Monomial Change of Variables

For the FBQF scheme to be of order M, the base and end point correction stencils must include $M+1$ terms, i.e. their upper limits of summation must be M. Since $s_i < t$ for $i = 0, ..., M$ in order to avoid the singularity in the base point correction, the starting of the FBQF scheme must be handled carefully. Given the first M values $y(kh)$ for $k = 0, M - 1$ are known and the FBQF method is used to solve for $y(Mh)$, the base point correction stencil must be restricted to only M terms i.e. the upper limit of summation is $M - 1$. The term in the base point correction stencil for which $i = M$ would be singular since $t = Mh$. If, instead, the first $M+1$ function values are known and the FBQF method is used to solve for $y((M + 1)h)$ then a base point correction stencil including $M + 1$ terms does not encounter a singularity. The latter approach is assumed in the presentation of this algorithm. An optimization routine is used to determine these first $M + 1$ points.

When the initial condition is only provided at $t = 0$, a monomial change of variables and a nonlinear optimization are used to accurately determine the remaining M function values. A monomial change of variables of the form $\eta(\rho) = \rho^k$, for $k > 0$ sufficiently large, serves to suppress any singularity at the origin as well as permit a closed form solution to the fractional derivative. An optimization routine uses the closed form solution to solve for function values a short distance from the origin. The resultant function values allow for the FBQF to begin stepping forward in

time. The steps in monomial-time $\{\rho_i\}_{i=1}^{M-1}$ are equally spaced such that $\rho_i = ih_\rho$.

The only assumption made on the smoothness of the solution in this section is that $u'(\rho)$ deviates smoothly from being zero, where $u(\rho) = y(\rho^k)$, and $y(\tau)$ is the solution to (3.3).

3.5.1.1 Optimization

Given the monomial change of variables

$$\tau = \rho^k \tag{3.20}$$

for some $k \in \mathbb{N}$, the integral from (3.2) is transformed such that

$$\int_0^t \frac{y'(\tau)}{(t-\tau)^\alpha}\, d\tau = \int_0^r \frac{u'(\rho)}{(r^k-\rho^k)^\alpha}\, d\rho \tag{3.21}$$

where $u(\rho) = y\left(\rho^k\right)$ and $r = \sqrt[k]{t}$.

In order to evaluate the integral in (3.21), let

$$u'(\rho) \approx \sum_{n=0}^{N} a_n \rho^n. \tag{3.22}$$

Approximation by a polynomial is considered appropriate when $u'(\rho)$ deviates smoothly from the initial condition. Such behavior at the origin is dependent on the monomial order k and the order ν of the smallest term in the expansion of $y(t)$. Given (3.5), $u'(\rho) = \rho^{k\nu-1} + o(\rho^{k\nu-1})$ and for (3.22) to be appropriate, $k > 1/\nu$. Given that ν is generally unknown, k will be set as large as numerically possible (see 3.7).

Assuming $u'(\rho)$ is well approximated by a polynomial with coefficients $\vec{a}$, then $u(\rho)$ is also

$$u(\rho) \approx \sum_{n=0}^{N} a_n \frac{\rho^{n+1}}{n+1} + u(0)$$

and the values $\vec{a}$ can be solved for as follows:

$$\begin{aligned}
\frac{1}{\Gamma(1-\alpha)}\int_0^r \frac{u'(\rho)}{(r^k-\rho^k)^\alpha}\,d\rho &= \frac{1}{\Gamma(1-\alpha)}\int_0^r \frac{\sum_{n=0}^{\infty} a_n\rho^n}{(r^k-\rho^k)^\alpha}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r \frac{a_n\rho^n}{(r^k-\rho^k)^\alpha}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r \frac{a_n\rho^n}{r^{k\alpha}\left(1-\left(\frac{\rho}{r}\right)^k\right)^\alpha}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r a_n\frac{r^n\left(\frac{\rho}{r}\right)^n}{r^{k\alpha}\left(1-\left(\frac{\rho}{r}\right)^k\right)^\alpha}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r a_n\frac{r^n\left(\frac{\rho}{r}\right)^{k\frac{n}{k}}}{r^{k\alpha}\left(1-\left(\frac{\rho}{r}\right)^k\right)^\alpha}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r a_n\frac{r^{n+1}\left(\frac{\rho}{r}\right)^{k\frac{n}{k}}\left(\frac{\rho}{r}\right)^{1-k}}{kr^{k\alpha}\left(1-\left(\frac{\rho}{r}\right)^k\right)^\alpha}\frac{k}{r}\left(\frac{\rho}{r}\right)^{k-1}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r a_n\frac{r^{n+1}\left(\frac{\rho}{r}\right)^{k\frac{1+n-k}{k}}}{kr^{k\alpha}\left(1-\left(\frac{\rho}{r}\right)^k\right)^\alpha}\frac{k}{r}\left(\frac{\rho}{r}\right)^{k-1}\,d\rho \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty}\int_0^r a_n\frac{z^{\frac{1+n-k}{k}}}{k\,(1-z)^\alpha}r^{1+n-k\alpha}\,dz \\
&= \frac{1}{\Gamma(1-\alpha)}\sum_{n=0}^{\infty} a_n\frac{\Gamma(1-\alpha)\Gamma\left(\frac{1+n}{k}\right)}{k\Gamma\left(1-\alpha+\frac{1+n}{k}\right)}r^{1+n-k\alpha} \\
&\approx \sum_{n=0}^{N} a_n\frac{\Gamma\left(\frac{1+n}{k}\right)}{k\Gamma\left(1-\alpha+\frac{1+n}{k}\right)}r^{1+n-k\alpha} \\
&= \sum_{n=0}^{N} a_n\Lambda_k^\alpha(r,n) \qquad (3.23)
\end{aligned}$$

where the factor of $\frac{1}{\Gamma(1-\alpha)}$ has been introduced to fully represent the fractional derivative, Λ has been introduced to simplify notation, and the definition of the Beta function was used in conjunction with it's relation to the Gamma function as seen in Equations 6.5 and 6.6. in [40].

Considering the FODE in (3.3), (3.23) gives an approximation for the left hand side at each equally spaced monomial time step ρ_m and evaluating the right hand side for each time step results in the following system

$$\begin{bmatrix} \Lambda_k^\alpha(\rho_1,0) & \Lambda_k^\alpha(\rho_1,1) & \cdots & \Lambda_k^\alpha(\rho_1,N) \\ \Lambda_k^\alpha(\rho_2,0) & \ddots & & \vdots \\ \vdots & & & \\ \Lambda_k^\alpha(\rho_M,0) & \cdots & & \Lambda_k^\alpha(\rho_M,N) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = g(\vec{\rho}, u(\vec{\rho})) \tag{3.24}$$

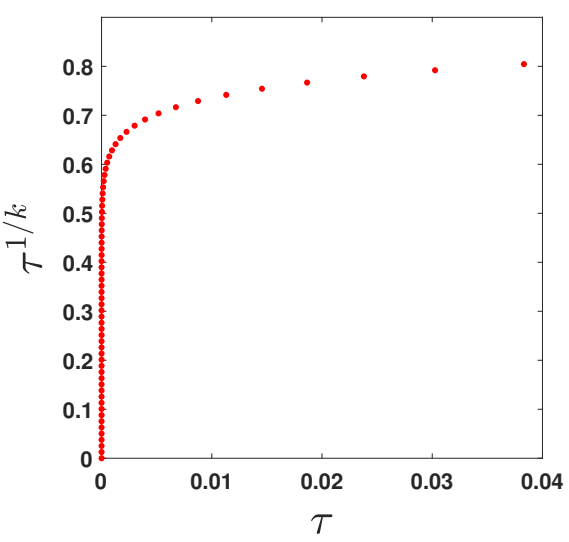
where $g(\rho, u(\rho)) = f(\tau, y(\tau))$, $\vec{\rho} = \rho_1, \rho_2, ..., \rho_M$, and

$$u(\vec{\rho}) = \begin{bmatrix} \rho_1^1/1 & \rho_1^2/2 & \cdots & \rho_1^{N+1}/(N+1) \\ \rho_2^1/1 & \ddots & & \vdots \\ \vdots & & & \\ \rho_M^1/1 & \cdots & & \rho_M^{N+1}/(N+1) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} + \begin{bmatrix} u(0) \\ u(0) \\ \vdots \\ u(0) \end{bmatrix}.$$

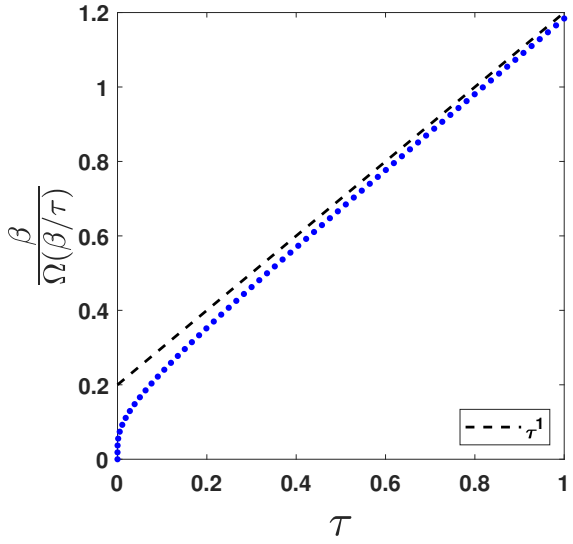
The system in (3.24) is not necessarily square nor is the right hand side guaranteed to be linear. As such, the vector of coefficients $\vec{a}$ must be solved for using an nonlinear optimization routine. A nonlinear least squares optimization routine is used in the present examples.

3.5.2 Exponential Change of Variables

The use of a monomial change of variables in conjunction with an optimization scheme allows the FBQF from Section 3.3 to step forward in time given only an initial condition. However, equally spaced steps in the monomial variable, ρ, correspond to steps with increasing separation in the untransformed variable, t, see Figure 3.1(a). Such growth becomes unreasonable as t becomes large. While a change of variables must be present in order to suppress the singularity at the base point, a new change of variables is introduced which grows linearly with t for t large, see Figure 3.1(b). Specifically, an exponential change of variables of the form $\phi(\theta) = \theta e^{-\beta/\theta}$ where $\beta > 0$ is introduced. The exponential-time steps are equally spaced such that $\theta_j = jh_\theta$ where $h_\theta \approx h$ for $t \gg 0$.



(a) Equally spaced points under a monomial change of variables $\eta(\rho) = \rho^k$, $k = 15$ compared to their value in the untransformed time, τ.



(b) Equally spaced points under a exponential change of variables $\phi(\theta) = \theta e^{-\beta/\theta}$, $\beta = 0.2$ compared to their value in the untransformed time, τ. The function $\Omega(z) = w$ is the Lambert W function such that $we^w = z$.

3.5.2.1 Interpolation

The function must be known at $\{\theta_j\}_{j=0}^{M}$ in order to use the FBQF to step forward in exponential-time. An interpolation scheme is used to find the function values in exponential-time from those located in monomial-time. To keep this an interpolation problem and not an extrapolation problem, the domain for the new data points, $[0, \phi(\theta_M)]$, must be a subset of the domain of the known data points $[0, \eta(\rho_{M+\mu})]$ where $\mu \in \mathbb{Z}$. That is, in addition to the minimum M function values which must be found to use the FBQF in monomial-time, the solution must be found for $\rho_i,\ i = M+1, ..., M+\mu$ where μ is the minimal value such that $\eta(\rho_{M+\mu}) \geq \phi(\theta_M)$. In practice, $\mu > 0$.

While the optimization routine alone can solve for the solution very far into the domain, it is less robust further from the base point. It is better to use the optimization routine to acquire the minimum M points necessary to start the FBQF and then step forward μ times. Then, interpolation determines the function values $y(\phi(\theta_j)),\ j = 0, ..., M$.

The ratio between the number of steps in monomial-time N_m and the number of steps in exponential-time N_e, affects the quality of the interpolation. While $\rho_{i+1} - \rho_i = h_\rho$ is true for all $i \in \mathbb{N}$, $\eta(\rho_{i+1}) - \eta(\rho_i)$ is not a constant value and will grow as i grows. As such, the largest distance between time values, when converting from monomial-time into exponential-time, will be that of $\eta(\rho_{M+\mu}) - \eta(\rho_{M+\mu-1})$. A parameter d is introduced to control the size of that spacing with regards to the spacing of nodes in exponential-time, h. Given $h = t/N_e$, then $h_\rho = \frac{T_m}{N_m}$ and $d = N_m/N_e$, where $T_m = \eta^{-1}(\phi(\theta_M)) = \rho_{M+\mu}$. The value of d should at least be that for which $\eta(\rho_{M+\mu}) - \eta(\rho_{M+\mu-1}) \leq \phi(\theta_M) - \phi(\theta_{M-1})$. Therefore we consider

$$d = \left\lceil \frac{\rho_{M+\mu}}{N_e \left(\eta^{-1}(\phi(\theta_M)) - \eta^{-1}\left(\phi\left(\theta_{M-1}\right)\right)\right)} \right\rceil. \tag{3.25}$$

3.5.3 Putting it all together

Given the multiple methods involved in making this algorithm work, we briefly recall the individual methods and how they fit together will be given. First, a monomial change of variables

is introduced. This change of variables serves to suppress the weak singularity in the integral of the Caputo derivative. The choice of a monomial change of variables also serves to enable a least-squares approach described in 3.5.1.1 for approximating the solution at M equally spaced monomial-time steps after the initial condition. Next, the solution is stepped forward in time using the FBQF described in 3.3. Once the solution has been found sufficiently far into the domain, the function values in ρ are interpolated onto locations $\{\theta_j\}_{j=0}^{M}$ based on an exponential change of variables. The second change of variables is introduced to maintain a linear relationship with untransformed time as time increases. For remaining time steps, the FBQF steps the solution forward in time.

3.6 Numerical Examples

For each of the examples below the parameters of the algorithm are the same. The monomial power is $k = 15$; the order of the polynomial approximation in (3.22) is $N = 1$; and a piecewise linear interpolation is used. The order of the fractional derivative will vary according to $\alpha = \{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99\}$. All calculations are done in double precision with about 16 significant digits.

All examples presented in this chapter have initial conditions $y(0) = 0$. The FBQF method, as it has been presented here, only converges robustly for zero initial conditions. The exact cause of this limitation is currently unknown and will be explored in future work.

Examples are classified based on their smoothness. In particular, example FODEs are included for which the solution is smooth (as in (3.26) and Figure 3.2) or not smooth (as in (3.27) and Figure 3.3 or (3.28) and Figures 3.4, 3.5, and 3.6). For solutions which are not smooth, a distinction is made between those solutions which produce a singularity proportional to α (as in (3.27)) and those which produce a singularity which is independent of α (as in (3.28)). The FODE in Example (3.26) is also nonlinear.

An instance of (3.28) is solved in Figure 3.4 for ν very near 0, in Figure 3.5 for ν very near 1, and in Figure 3.6 for $\nu = 4\sqrt{2}/11 \times 10^{-1} \approx 0.0514$. The first two values of ν are chosen to highlight

the performance of this algorithm in regimes where it struggles to converge and where it converges robustly, respectively. The third value of ν is chosen to be on the border between the two regimes. In each of these examples ν is chosen to be an arbitrary irrational number in order to ensure the singularity is truly independent of α.

It is a well-known behaviour for fractional problems and classical parabolic problems that a numerical method may achieve a higher rate of convergence in a norm which is restricted to a subdomain excluding any singularities when compared to the convergence in a global norm [97]. For each example, both the L^∞ norm of the error over the interval $[0, 1]$ and the error at $t = 1$ are shown in the figures. For each figure demonstrating the convergence of the method, note that h_θ is the step size once initializations are done with and the method can progress forward indefinitely.

Finally, in order to demonstrate the capability of this method in the context of the larger field of numerical methods for FODEs, a comparison has been drawn between this method and two others. The convergence and computational time when solving (3.26) for $\alpha = 1/2$ are compared between the FBQF method, the FLMM method [45], and the Two-step Spline Collocation (TS) method [14] in Figures 3.7(a) and 3.7(b) respectively. Since a rigorous analysis of the computational cost of the FBQF method as it is presented here is not included in this chapter, a demonstration of the computational time as compared to two other methods is supplied.

$$D^\alpha y(t) = \frac{2\Gamma(4)}{\Gamma(4-\alpha)} t^{3-\alpha} - \frac{3\Gamma(5)}{\Gamma(5-\alpha)} t^{4-\alpha} + \frac{\Gamma(6)}{\Gamma(6-\alpha)} t^{5-\alpha} + \left(t^5 - 3t^4 + 2t^3\right)^2 - y^2 \tag{3.26a}$$

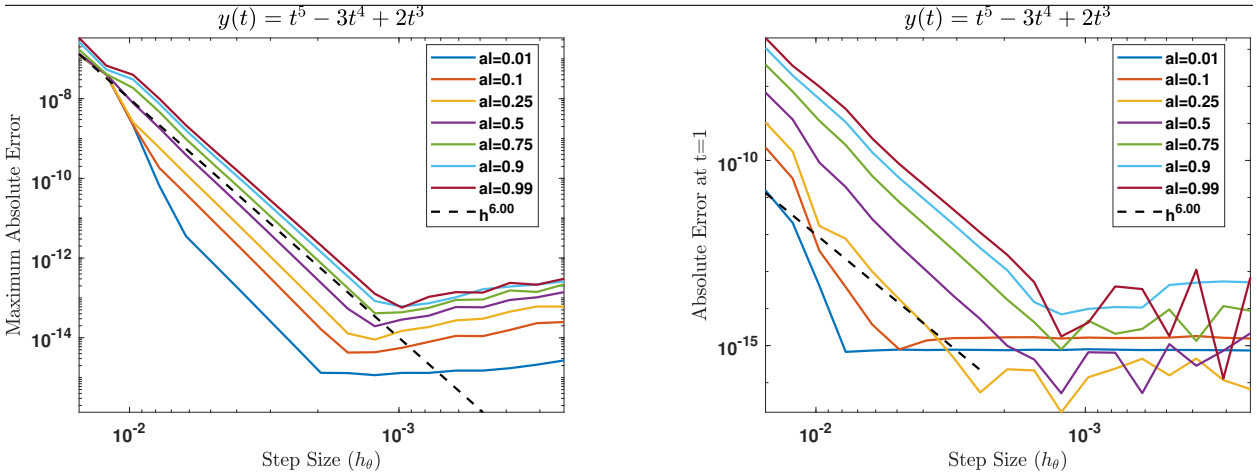
$$y(t) = t^5 - 3t^4 + 2t^3 \tag{3.26b}$$

$$y(0) = 0 \tag{3.26c}$$

$$D^\alpha y(t) = t - y(t) + 1 \tag{3.27a}$$

$$y(t) = 1 - E_\alpha\left(-t^\alpha\right) + t^{\alpha+1} E_{\alpha,\alpha+2}\left(-t^\alpha\right) \tag{3.27b}$$

$$y(0) = 0 \tag{3.27c}$$



(a) Maximum absolute error

(b) Absolute error at time $t = 1$

Figure 3.2: Errors in solving the FODE (3.26) for many values of α, various step sizes across the interval $[0, 1]$, and monomial order $k = 15$.

$$D^\alpha y(t) = t^{\nu-\alpha} \frac{\Gamma(1+\nu)}{\Gamma(\nu-\alpha+1)} \tag{3.28a}$$

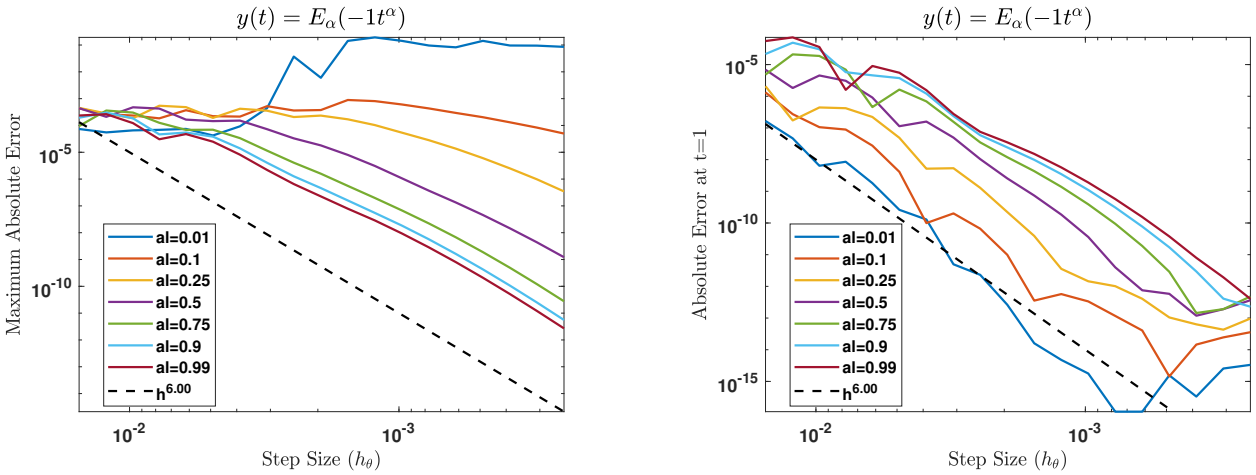
$$y(t) = t^\nu \tag{3.28b}$$

$$y(0) = 0 \tag{3.28c}$$

3.7 Discussion

The convergence plots in Figures 3.2-3.5 experimentally demonstrate that the method described in this chapter is accurate and robust. In the case of solutions with no singularity (Figure 3.2), the algorithm is able to achieve 6th order convergence for a wide range of fractional derivative orders α both at the end point $t = 1$ and when the maximum error is taken over the interval $[0, 1]$.

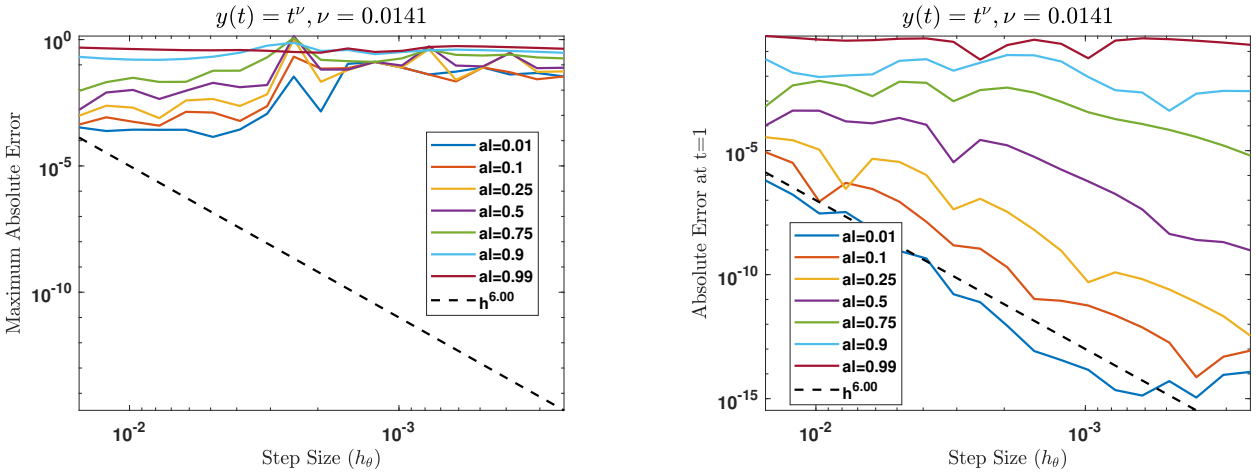
In the case of solutions which cause a base point singularity proportional to α as in (3.27), the method is able to achieve 6th order convergence for all values of α at the end point $t = 1$ as seen in Figure 3.3(b). When the error over the entire domain is considered, the method demonstrates $\mathcal{O}(h^6)$ convergence for $\alpha > 1/2$ and decreasing orders of convergence as $\alpha \to 0$. In these cases, the



(a) Maximum absolute error

(b) Absolute error at time $t = 1$

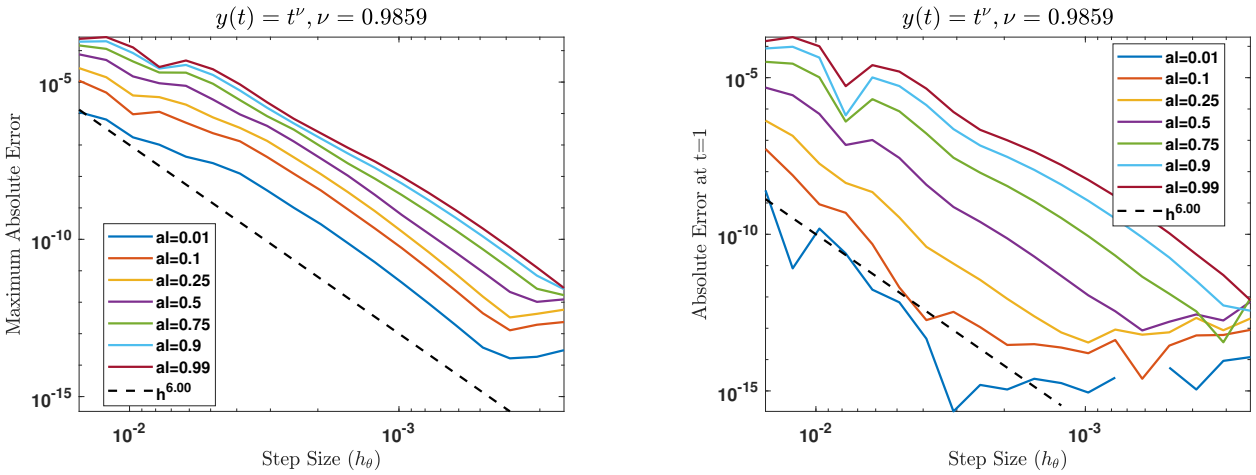
Figure 3.3: Errors in solving the FODE (3.27) for many values of α, various step sizes across the interval $[0, 1]$, and monomial order $k = 15$.



(a) Maximum absolute error

(b) Absolute error at time $t = 1$

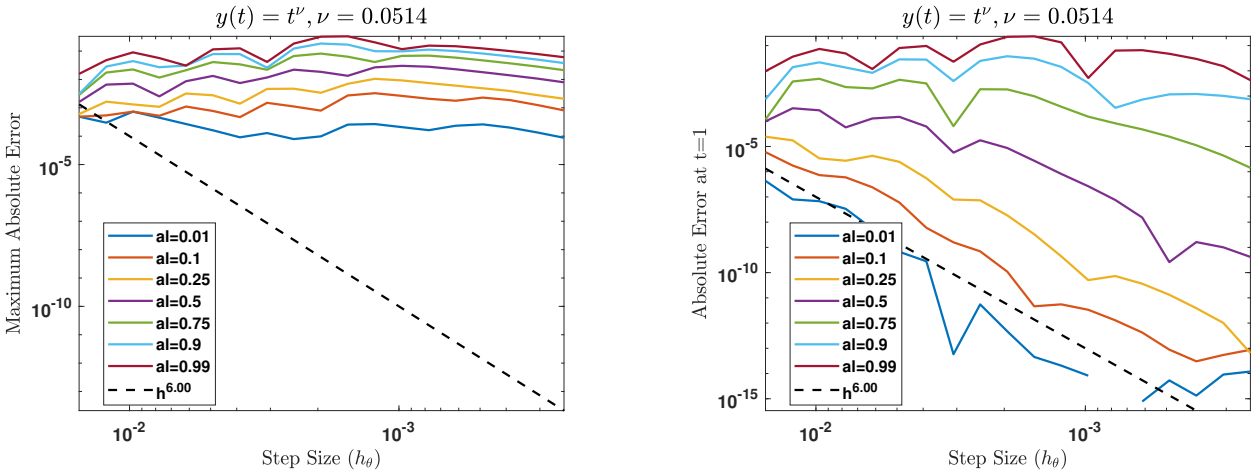
Figure 3.4: Errors in solving the FODE (3.28) where $\nu = \sqrt{2} \times 10^{-3} \approx 0.0014$ for many values of α, various step sizes across the interval $[0, 1]$, and monomial order $k = 15$.



(a) Maximum absolute error

(b) Absolute error at time $t = 1$

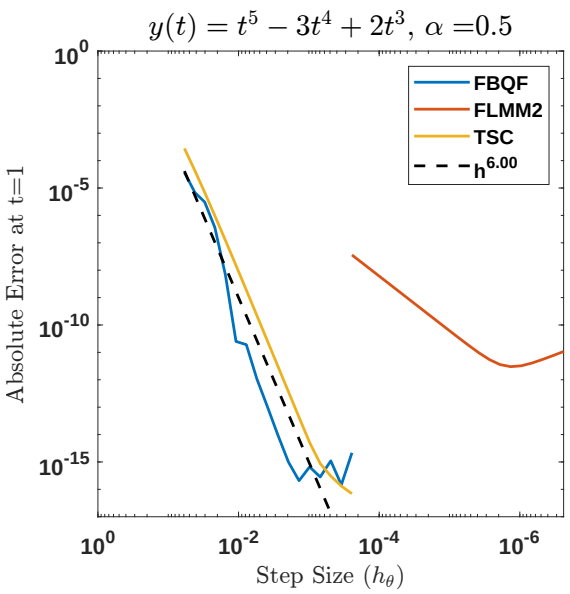
Figure 3.5: Errors in solving the FODE (3.28) where $\nu = 1 - \sqrt{2} \times 10^{-3} \approx 0.9991$ for many values of α, various step sizes across the interval $[0, 1]$, and monomial order $k = 15$.



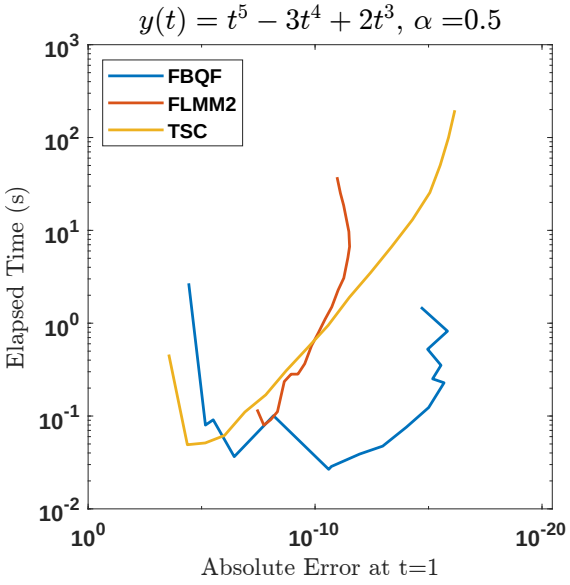
(a) Maximum absolute error

(b) Absolute error at time $t = 1$

Figure 3.6: Errors in solving the FODE (3.28) where $\nu = 4\sqrt{2}/11 \times 10^{-1} \approx 0.0514$ for many values of α, various step sizes across the interval $[0, 1]$, and monomial order $k = 40$.



(a) Convergence of the absolute error at time $t = 1$ compared between the FBQF, FLMM, and Two-step Spline Collocation (TSC) methods.



(b) Elapsed time as a function of absolute error at time $t = 1$ compared between the FBQF, FLMM, and Two-step Spline Collocation (TSC) methods.

Figure 3.7: Comparisons of error convergence and computation time between the FBQF, FLMM, and Two-step Spline Collocation (TSC) methods.

dominant errors result from the optimization step and thus are most prevalent very near $t = 0$[3].

Figures 3.4-3.6 demonstrate the performance of the method on solutions with singularities independent of α. In these cases, given $y(t) = t^\nu$, the singularity is of order $1 - \nu$. When ν is near 1, the method performs well across the entire interval $[0, 1]$ and at the point $t = 1$.

However, as the order of the singularity $(1 - \nu) \to 1$ (or $\nu \to 0$), as in Figure 3.4(a), the maximum absolute error becomes approximately $\mathcal{O}(1)$ for all values of α. The error is significantly more well behaved at the end point $t = 1$ as seen in 3.4(b) yet the maximum absolute error still fails to converge for α near one. This behavior is caused by numerical limitations in how large the monomial order can be. In practice, due to numerical overflow and underflow, the monomial order is limited to $m \approx 40$ when $0.01 \leq \alpha \leq 0.99$ and $2^6 \leq N_\theta \leq 2^{12}$. Therefore, it should not be expected that this method can robustly solve FODEs with singularities of order $(1 - \nu)$ for $\nu < 2/40$ when $\alpha > 1/2$. The case of $\nu = 4\sqrt{2}/11 \times 10^{-1} \approx 2/40$ is shown in Figure 3.6.

The method appears to work best for singularities of order $(1 - \nu)$ for $\nu > 1/2$, achieving $\mathcal{O}(h^4)$ to $\mathcal{O}(h^6)$ convergence for all tested values of α in these cases. When $\nu < 1/2$ and the monomial order is insufficient to overcome the singularity, relatively large errors are present near the base point. Such errors steadily decrease as the distance between the base and evaluation points increases. Convergence for $\alpha < 1/2$ is $\mathcal{O}(h^6)$ but decreases significantly as α decreases.

Optimal convergence of $\mathcal{O}(h^6)$ is achieved for all examples in which $f \in C[0, t]$. Note that the base point of $\tau = 0$ is included in the interval of continuity. For some cases in which f is not continuous at the base point, the method is still able to achieve $\mathcal{O}(h^6)$ (see (3.28) and Figure 3.4(b) for $\alpha = 0.01, 0.1, 0.25$). The convergence is suspected to be $\mathcal{O}(h^6)$ for all cases of $\nu > \alpha - 1/2$, however this has only been experimentally observed in limited cases. Rigorous error analysis should be done.

[3]For $\alpha = 0.01$, the initial increase in error does not represent a true divergence of the method as demonstrated by the stagnation as step size decreases. The low error for larger step sizes should be considered a serendipitous bonus.

3.8 Conclusion

This chapter presents a method for solving IVPs involving FODEs. A monomial change of variables is used to start the method via optimization and the FBQF method steps forward in monomial-time. Once a transition point has been reached, the function values in monomial-time are interpolated to exponential-time corresponding to a second change of variables. The FBQF method is used again to advance the model until the desired time.

For FODEs with smooth solutions, the method is demonstrated to achieve $\mathcal{O}(h^6)$ convergence for all orders of fractional derivative. For FODEs with solutions which cause singularities proportional to α (3.27) the convergence is $\mathcal{O}(h^6)$ for all tested values of α at the end point $t = 1$. For FODEs with solutions which cause singularities independent of α (3.28), optimal convergence of $\mathcal{O}(h^6)$ is achieved for all examples in which $\nu > \alpha - 1/2$. Convergence is limited to those FODEs with initial condition $y(0) = 0$.

Chapter 4

Node subsampling for multilevel meshfree elliptic PDE solvers

This chapter serves as a summary of the work published in [57] in collaboration with Drs. Bengt Fornberg and Morten E. Nielsen, which describes a method for subsampling quasi-uniform, variable density node sets while preserving relative local node densities.

4.1 Introduction

The need for efficient numerical methods for solving partial differential equations (PDEs) is a cornerstone in computational mathematics. Traditional methods often rely on mesh-based approaches, which can be computationally intensive and complex. In contrast, meshfree methods offer flexibility and simplicity, particularly for problems involving complex geometries or moving boundaries. This chapter summarizes the work work published in [57], in collaboration with Drs. Bengt Fornberg and Morten E. Nielsen, which investigates a node subsampling strategy to enhance multilevel meshfree solvers for elliptic PDEs.

4.1.1 Background and Related Work

Node subsampling has a wide range of application areas beyond multilevel methods, including polynomial approximation [18, 80], numerical integration [84, 96], machine learning [53, 78, 92, 94], and computer graphics [17, 27, 111]. However, most subsampling routines from these applications are unsuited to multilevel methods for meshfree PDE solvers.

Meshfree methods, such as radial basis function-generated finite differences (RBF-FD) meth-

ods, provide alternatives to mesh-based techniques. These methods allow for high geometric flexibility making them particularly advantageous for handling irregular domains. However, the quasi-uniform[1] node sets which discretize complex domains are likely to contain high density variations. Additionally the underlying node sets must meet certain quality constraints in order to ensure stability and accuracy of the solution [37, 38, 66]. Robust algorithms for generating such node sets exist [36, 98, 104] and are utilised in this chapter.

Multilevel techniques are well-established for improving computational efficiency by solving a given problem on a hierarchy of discretizations [31, 108]. Combining multilevel approaches with RBF-FD methods yields robust solvers capable of tackling large-scale problems accurately and efficiently [112, 109]. However, existing methods are either inflexible or use algebraic multigrid methods[2].

Therefore, Section 4.2 of this chapter focuses on node subsampling strategies for variable-density, quasi-uniform node sets which maintain the quality of the initial node set. Then, Section 4.4 demonstrates the use of the best subsampling algorithm in conjuction with a multilevel RBF-FD solver for elliptic PDEs.

4.2 The Subsampling Algorithm

The primary contribution of this section is the development of a node subsampling technique based on a moving front method. It is shown to out-perform three other techniques from the literature which are relevant to subsampling variable density, quasi-uniform node sets.

[1] Quasi-uniform is here used to refer to node sets which may vary in density globally yet are near-uniform locally, i.e. there exists some $c > 0$ such that for any discretization of the domain, $h^{(n)}_{max}/h^{(n)}_{min} < c$ where $h^{(n)}_{max}$ and $h^{(n)}_{min}$ are the maximum and the minimum distances, respectively, to the nearest neighbor for some discretization n.

[2] Algebraic multigrid methods take more iterations to converge [109], require more time to set-up [107] [105], and are sensitive to parameter choice and node set variation [105]

4.2.1 The Contenders

Each algorithm to be later compared is discussed here. The novel method is described in detail and for each other algorithm, the reader is referred to the primary source. Implementations for all algorithms can be found on the author's GitHub, [55].

4.2.1.1 Moving Front

A novel subsampling was inspired by the 'moving front' strategies found in the node generation algorithms of [104] and [36]. Algorithm 1 describes the method in detail. First, all of the fine nodes are sorted and their k nearest neighbors are found. Then, for each node, working in the direction of sort, if a node has not already been marked, mark each node that is both above the present node in the sort and within a distance $c * D_1$ (for example $c = 1.5$) of the present node, where D_1 is the distance to the first nearest neighbor. All marked nodes are then removed to produce the coarse node set.

The nodes are searched based on a k-d tree. For the efficiency of the algorithm, the k nearest neighbors are searched, where $k = H(n+1) - 1 = 3n(n+1)$, $H(n)$ is the n^{th} centered hexagonal number, and $n = \lceil c \rceil$. This value of k represents the maximal number of nodes which can be packed into a ball of radius $n * D_1$ excluding the central node such that no more than k nodes could be within a distance of $n * D_1$ of the center node.

The moving front algorithm generalizes immediately to any number of space dimensions: nodes are sorted based on a direction, nearest neighbors are found, and nodes are marked based on proximity. The code for this implementation can be found on the author's GitHub, [55].

It should be noted that the moving front algorithm natural contains a directional bias. The effects of this directional bias are insignificant, however, as shown in the Sections 4.3.2 and 4.4.2. Yet, in the event that the subsampling is repeatedly applied, the direction of the algorithm should be changed at each iteration, using as many different directions as possible.

Algorithm 1 Moving Front Algorithm

function MFSub($X\_fine = \{\boldsymbol{x_1}, ..., \boldsymbol{x_N}\}, c$)
    $k = H(\lceil c \rceil)$
    Sort the nodes in X_fine
    Find the indices and distances of the k nearest neighbors for each point in X_fine
    for $i = 1 : N$ **do**
        if the node $\boldsymbol{x_i}$ has not already been marked **then**
            Determine which nearest neighbors are within a radius of c times the distance to the nearest neighbor
            Of those, determine which are 'above' $\boldsymbol{x_i}$ in the sort order
            Mark these nodes
        end if
    end for
    Remove the marked nodes from X_fine to produce X_coarse **return** X_coarse
end function

4.2.1.2 Other Methods

Three additional methods are compared against the moving front method described above. The weighted subsampling is based on the work presented in [111], but modified for variable density node sets. The Poisson disk subsampling [9] is similar to the thinning method presented in [98] and the Poisson thinning method presented in [58]. The generalized Diversity Subsampling algorithm can be found in [92] for a distribution based on the distance to the nearest neighbor of each node.

4.2.2 Comparisons of Subsampling Methods

The performance of each algorithm is compared based on an iterative coarsening of the dithered trui image[3], as shown in the first column of Figure 4.1. This initial node set serves as an upper bound of sorts in the context of complex, highly variable quasi-uniform domain upon which PDEs might need to be solved. The trui image is also a good test problem for heuristic evaluations.

4.2.2.1 Heuristic Comparison

The first comparison involves each algorithm subsampling the dithered trui image twice, as seen in Figure 4.1. The original node set contains 36303 nodes, the first subsample contains 10553 nodes, and the second subsample contains 3404 nodes. Each algorithm, excluding the generalized diversity subsampling algorithm[4], requires a parameter, c, to control the level of coarsening. To reproduce the subsamples in Figure 4.1, the values of c used in each algorithm are listed in Table 4.1.

[3]The dithering was performed by the algorithm in [104].

[4]The generalized diversity subsampling algorithm explicitly requires a target number of nodes as input rather than a parameter.

Method	First Subsampling	Second Subsampling
MF	1.50976	1.515
W	3.44	3.1
PD	1.51	1.521

Table 4.1: The parameters, c, for reproducing the node sets in Figure 4.1 for the moving front (MF), weighted (W), and Poisson disk (PD) subsampling algorithms. The generalized diversity subsampling algorithm explicitly relies on the desired number of nodes in the coarse node set and thus has no parameter listed here. These values are arbitrary, having been chosen only to achieve the same node count across methods.

4.2.2.2 Node Quality Measures

The second comparison is based on quantitative metrics. Unfortunately, quantitative measurements for how well a subsampling algorithm preserves original node quality are very hard to find. Some metrics which describe the quality of spatially variable node densities do exist [104] and they will serve as inspiration for the comparative measures that follow. One useful quality metric of a variable density node set is that of the local regularity as determined by the distribution of distances to the nearest k neighbors $\delta_{i,j}$, $i = 1, 2, ...k$ for each node x_j.

Two novel measures of subsampling quality are presented here. They are based on measures of local regularity which are commonly used for evaluating the quality of variable density node sets. The novel metrics are referred to as measures of comparative local regularity (CLR). These CLRs contrast the average or standard deviation of distances to the k nearest neighbors of the fine node set from those of the coarse node set.

For each node set X, the Euclidian distance between each node $x_j \in X$ and its k nearest neighbors is calculated as $\delta_{i,j} = \|x_j - x_{i,j}\|$. The average $\overline{\delta}_j$ or standard deviation σ_j of these distances is then found for each node $x_j \in X$. These are typical measures of local regularity. However, the present goal is measure how similar the initial and subsampled node sets are. To this

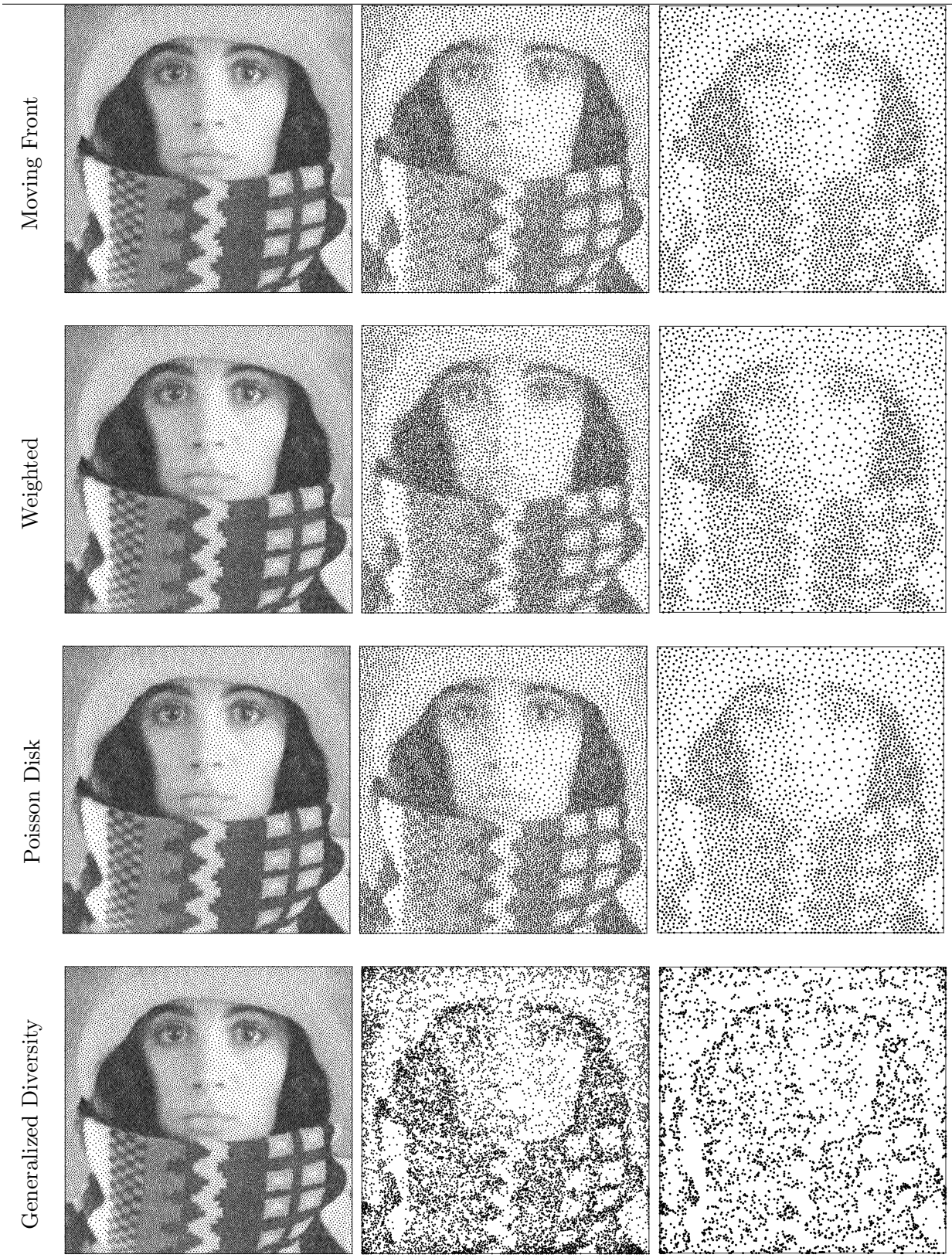


Figure 4.1: A visualization of the initial dithered trui image (36303 nodes), the first subsampling (10553 nodes), and the second subsampling (3404 nodes). From top to bottom, the rows demonstrate the moving front, weighted, Poisson disk, and generalized diversity subsampling algorithms. The first column of the figure repeats the same image of the initial dithering for convenience in comparison.

end, the distributions of $\overline{\delta}_j^{\text{fine}}$ and $\overline{\delta}_j^{\text{coarse}}$ over each of the fine and coarse node sets, respectively, are first normalized to be between 0 and 1. Then the difference between these values is calculated at each of the nodes which is collocated in both the coarse node set and the fine node set[5]. Finally, the standard L_2 norm is taken to produce a measure of CLR, which will depend on k. The same process can be applied using the standard deviation σ_j of those distances $\delta_{i,j}$.

The CLRs, as presented above, measure the difference between a given initial node set and a subsampling of it. Therefore, a better subsampling routing will have lower CLR values. Additionally, the values should only be compared between subsamplings of the same size and from the same initial node set.

4.2.2.3 Computational Cost

Finally, the algorithms are compared by their computational costs. The moving front, weighted, and Poisson disk subsampling algorithms as they are presented in this chapter were coded in MATLAB. The generalized diversity subsampling algorithm is coded in Python as presented in [92]. A comparison of the computational complexity can be found in Figure 4.3. The average execution times in Figure 4.3 are calculated based on ten repetitions of each algorithm subsampling from the node set size of the previous data point to the node set size for the given point using the `timeit` commands native to MATLAB and Python. The computations were performed on an AMD Milan, 2.8 GHz processor with Linux operating system.

4.2.3 Discussion

The heuristic comparison primarily establishes the visual goodness of the first three algorithms over the generalized diversity subsampling algorithm. Among the remaining three, the moving front and Poisson disk algorithms preserve a higher level of clarity in the woman's features (nostrils and mouth) and scarf patterns when compared to the weighted subsampling algorithm.

[5]Effectively, $\overline{\delta}_j^{\text{fine}}$ needs only be calculated at those nodes x_j in the fine node set which also belong to the coarse node set

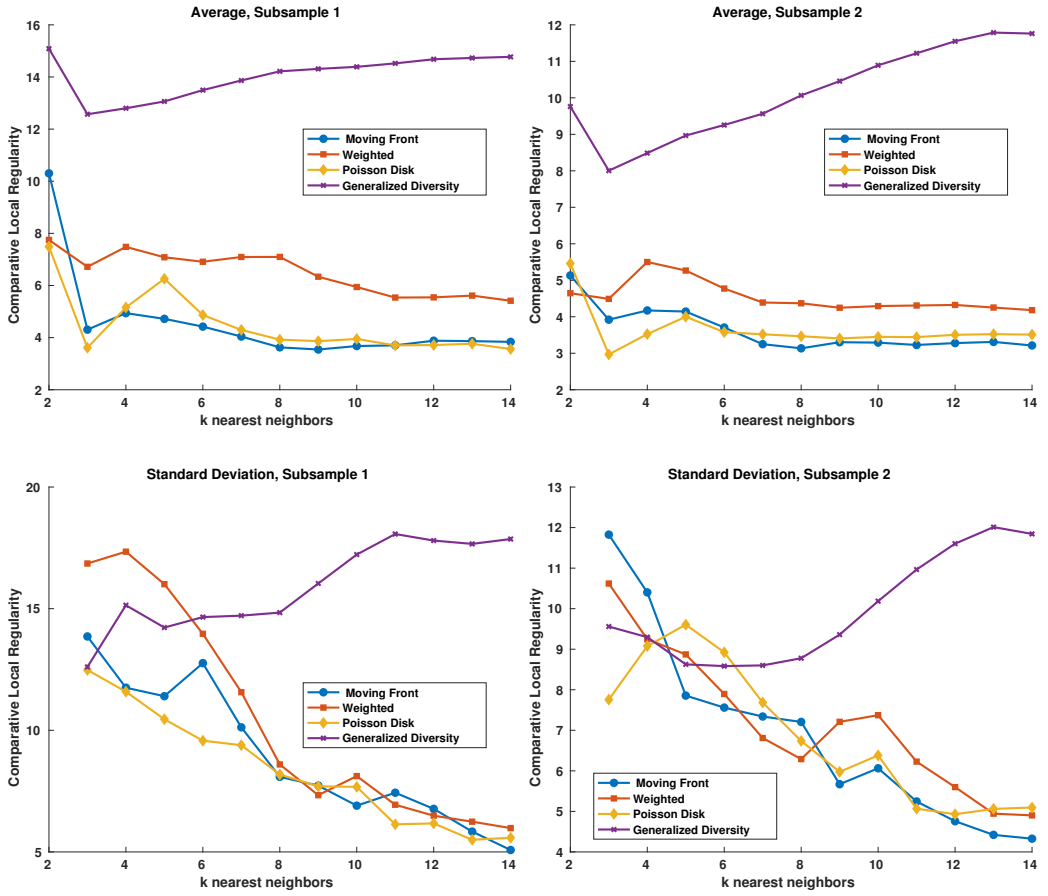


Figure 4.2: The comparative local regularity (CLR) of the average distance and standard deviation of distances for $k = 2, 3, ..., 14$ nearest neighbors of various subsampling methods (weighted, moving front, Poisson disk, and generalized diversity subsampling) applied once and twice to the dithered trui image. For both measures of CLR, a lower value is better.

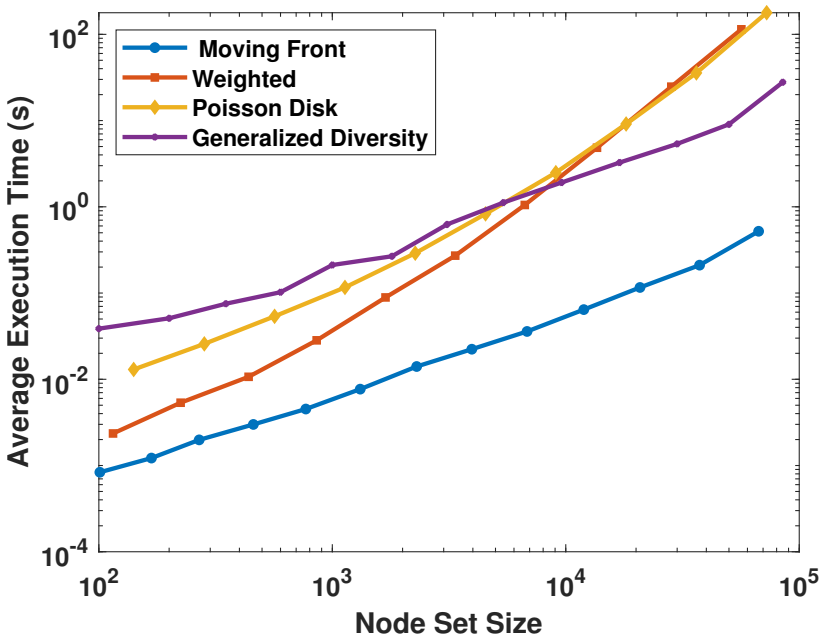


Figure 4.3: The computation time of each subsampling algorithm. The node set size is that of each resultant subsample. Each subsampling is serial in nature such that the coarse node set of the previous iteration is the fine node set of the next. The execution time was averaged over ten iterations of the moving front, weighted, Poisson disk, and generalized diversity subsampling algorithms. Note the logarithmic scales. The moving front algorithm is significantly faster than any of the other algorithms.

Further, the moving front and Poisson disk subsampling algorithms each better preserve the density disparities between areas of low and high node density in the original dithering[6] than do the weighted or generalized diversity subsampling algorithms. It should be noted that no direction bias of the moving front algorithm is visible.

Based on the results in Figure 4.2, it is clear that the moving front and Poisson disk algorithms preserve the characteristics of the initial dithered trui image better than the other two algorithms. This behavior is consistent across various sizes of node sets. Between them, however, it is not clear which one is quantitatively better.

The moving front algorithm is clearly the fastest algorithm of those compared in Figure 4.3. The significant computational cost savings secures the moving front algorithm as the best overall performing algorithm.

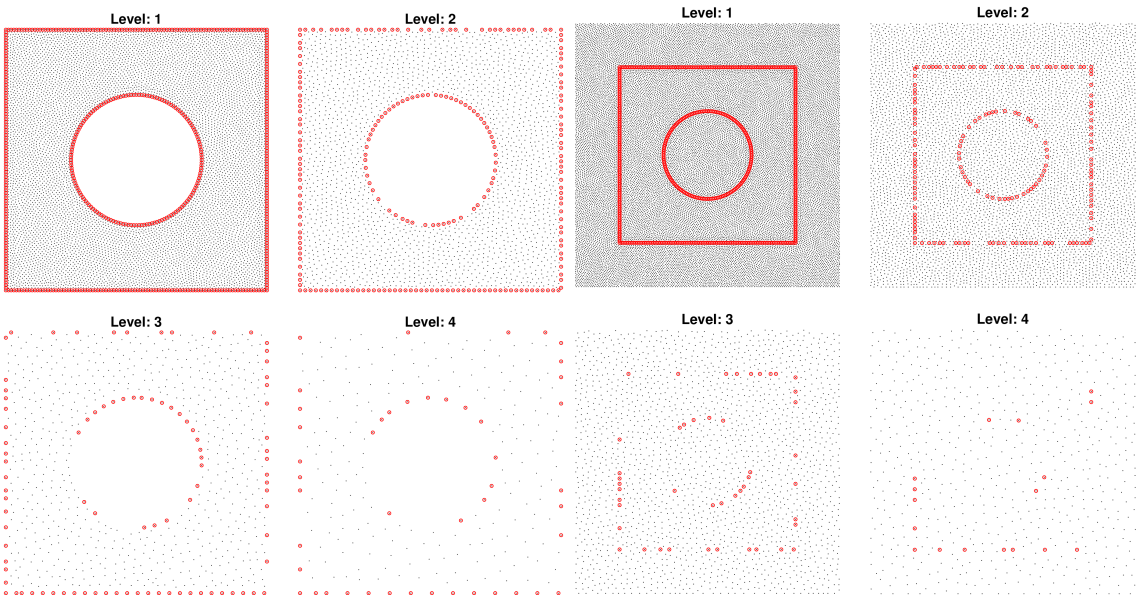
4.3 Boundary Considerations

The moving front subsampling algorithm was shown in Section 4.2 to outperform the other included algorithms. However, before it can be applied to test problems in Section 4.4, some potential pitfalls must be accounted for. Specifically, this section will demonstrate that boundary nodes must be subsampled separately from the interior nodes. For this section, initial node sets are generated by [104] and nodes near the boundary have been repelled, following the repel methodology described in [36], prior to any subsampling.

4.3.1 Subsample Boundary with Domain

While the directionality of the moving front algorithm is negligible for large 2D domains, the boundaries to those domains suffer more obvious effects, as can be seen in Figure 4.4(a). Such inconsistencies can be reduced by including nodes interior and exterior to each boundary as seen in Figure 4.4(b).

[6] The disparity is most notable between the woman's left cheek and hair and between the light stripe in the scarf on the woman's right and any of the surrounding regions.



(a) Naive algorithm

(b) Nodes included interior and exterior to boundaries

Figure 4.4: The moving front subsampling algorithm applied to a test node set with two boundaries. The initial node set is subsampled three times. The boundary node set is included in the domain node set such that they are subsampled collectively and simultaneously. Subsampling performance along the boundary is improved by including nodes interior and exterior to all boundaries. The method parameters for this example are $k = 10$ and $c = 1.5$.

Another way to reduce the inconsistencies in subsampling which may be due to the inherent directional bias of the moving front algorithm is to alternate the direction between subsampling iterations. This technique of alternating direction is unsatisfactory, however, because an ideal method would be effective independent any inherent directional bias.

4.3.2 Subsample Boundary Separately

To further improve robustness of the moving front method in the presence of boundaries, boundaries are subsampled separately. First, the given boundary nodes are subsampled. Then, any domain nodes within a prescribed distance of the boundary nodes are removed. Finally, the domain nodes are subsampled. The effects of this two-step process can be seen in Figure 4.5. Figure 4.5(a) alternates direction of the moving front algorithm while Figure 4.5(b) does not. A significant improvement in how consistently the algorithm behaves across the node set is apparent, independent of any direction bias in the subsampling algorithm.

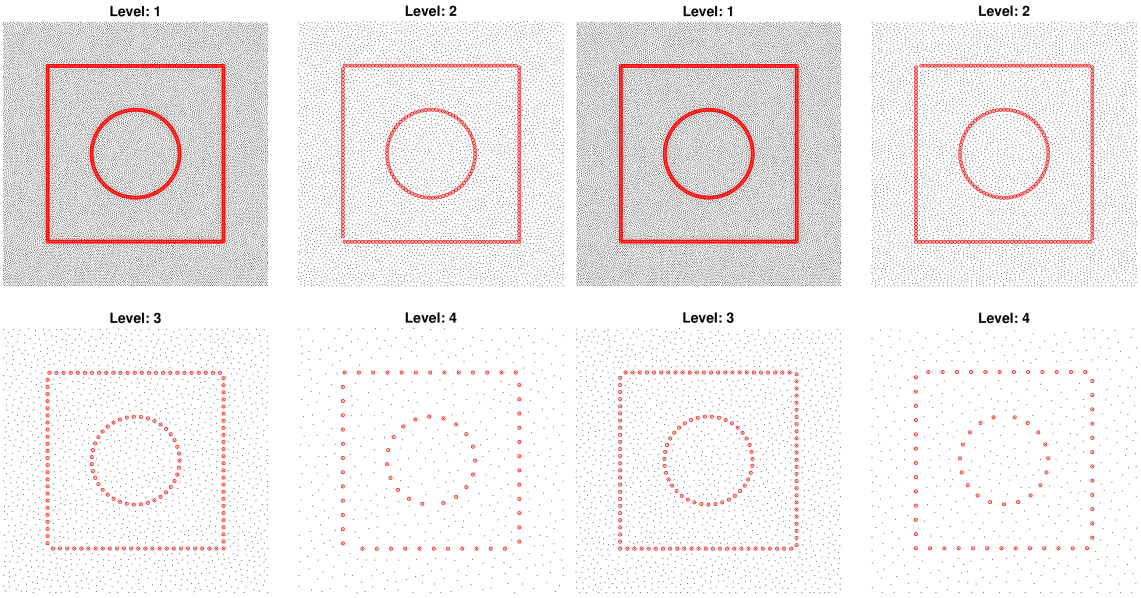
4.4 Meshfree multilevel RBF-FD solver

Finally, the moving front subsampling method can be applied to a multilevel RBF-FD solver. This section illustrates how to utilize the geometric flexibility of RBF-FD in combination with the proposed node subsampling strategy to setup a geometric multigrid method (GMM).

Consider the two-dimensional Poisson problem,

$$\begin{aligned} \nabla^2 u &= f, \quad \boldsymbol{x} \in \Omega \\ u &= g, \quad \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{4.1}$$

where $u = u(\boldsymbol{x}) = u(x, y) \in \mathbb{R}$ is the exact solution in a disk with unit diameter, i.e., $\Omega = \{(x, y), x^2 + y^2 \leq 0.5\}$, $g = g(\boldsymbol{x}) \in \mathbb{R}$ specifies Dirichlet boundary conditions on $\partial\Omega$ and $f = f(\boldsymbol{x}) \in \mathbb{R}$ specifies the source term. Discretizing the Laplacian with RBF-FD using polyharmonic splines and appended polynomials (PHS+poly) returns the following system which can be solved using a



(a) Alternating direction

(b) No alternating direction

Figure 4.5: The moving front subsampling algorithm applied to a test node set with two boundaries. The initial node set is subsampled three times. In these figures, the boundary node set is subsampled separately from the domain node set. Subsampling performance along the boundary is improved by subsampling boundary nodes independently. Additionally, no directional bias is detectable even when the algorithm does not alternate direction. The method parameters for this example are $k = 10$ and $c = 1.5$.

multilevel method:

$$L\boldsymbol{u} = \boldsymbol{f}. \tag{4.2}$$

4.4.1 Geometric multilevel elliptic solver

The meshfree GMM introduced here is based on similar ideas to those of the meshfree GMM found in [109] which is used for solving PDEs on surfaces. However, a few key differences should be noted. In this study, no Krylov subspace methods are used to increase the rate of convergence [47], the coarse grid difference operators are computed explicitly on each node set level, and all restriction operations are performed as injection (i.e. directly using values from the fine node set).

The basis of the GMM is the geometric multilevel V-cycle, which is described in Algorithm 2, where u_1 is the current approximation at the finest node set level, $L = \{L_j\}_{j=1}^{p}$ are the difference operators computed for each level, $I = \{I_{j+1}^{j}\}_{j=1}^{p-1}$ are the interpolation operators for each level and $R = \{R_j^{j+1}\}_{j=1}^{p-1}$ are the restriction (injection) operators for each level. During the V-cycles, pre- and post smoothing operations are performed using (ν_1, ν_2) Gauss-Seidel relaxations, respectively. At the coarsest node set level, a sparse LU solver is used.

A pseudocode for the proposed GMM is given in Algorithm 3. The multilevel solver performs up to i_{max} iterations (V-cycles) unless the relative residual $||r_1^i||_2/||r_1^0||_2 = ||f_1 - L_1u_1^i||_2/||f_1 - L_1u_1^0||_2$ of the ith iteration becomes less than a predefined tolerance tol.

The pseudocode for performing the geometric multilevel preprocessing, i.e., establishing all the different subsets of nodes, $X = \{X_j\}_{j=1}^{p}$, and the discrete operators L, I, R, is given in Algorithm 4. The first two inputs for this algorithm are the node set $\{X_{bg}$ which describes the fine level scattered node set covering the domain Ω and the node set $X_b\}$ which includes the fine level boundary nodes on $\partial\Omega$. The final input for Algorithm 4, $N_{\min}$, sets the minimum number of boundary nodes in the coarsest node set level.

Algorithm 2 Geometric multilevel V-cycle

function MLVCYC($u_1, f_1, L, I, R, \nu_1, \nu_2$)
  $u_1 \leftarrow \text{RELAX}(u_1, f_1, L_1, \nu_1)$
  $r_2 \leftarrow R_1^2(f_1 - L_1 u_1)$
  for $j = 2$ to $p - 1$ **do**
    $e_j \leftarrow \text{RELAX}(0, r_j, L_j, \nu_1)$
    $r_{j+1} \leftarrow R_j^{j+1}(r_j - L_j e_j)$
  end for
  $e_p = \text{LUSOLVE}(L_p, r_p)$
  for $j = p - 1$ to 2 **do**
    $e_j \leftarrow e_j + I_{j+1}^j e_{j+1}$
    $e_j \leftarrow \text{RELAX}(e_j, r_j, L_j, \nu_2)$
  end for
  $u_1 \leftarrow u_1 + I_2^1 e_2$
  $u_1 \leftarrow \text{RELAX}(u_1, f_1, L_1, \nu_2)$
  return u_1
end function

Algorithm 3 Geometric multilevel solver

function MLSOLVER($u_1^0, f_1, L, I, R, \nu_1, \nu_2, tol_1, tol_2, i_{max}, i_{conv}$)
  $i \leftarrow 0$
  $r_1^0 \leftarrow ||f_1 - L_1 u_1^0||_2$
  $counter \leftarrow 0$
  while $i < i_{max}$ and $||r_1^i||_2 > ||r_1^0||_2 \cdot tol_1$ and $counter < i_{conv}$ **do**
    $i \leftarrow i + 1$
    $u_1^i \leftarrow \text{MLVCYC}(u_1^{i-1}, f_1, L, I, R, \nu_1, \nu_2)$
    $r_1^i \leftarrow ||f_1 - L_1 u_1^i||_2$
    if $|\ln r_1^i - \ln r_1^{i-1}| < tol_2$ **then**
      $counter \leftarrow counter + 1$
    else
      $counter \leftarrow 0$
    end if
  end while
  return u_1^i
end function

Algorithm 4 Geometric multilevel preprocessing

function MLPRE($X_{bg}, X_b, N_{\min}$)
  $[X, R, p] \leftarrow \text{MLMFSUB}(X_{bg}, X_b, N_{\min})$
  $L_1 \leftarrow \text{RBFFD}(X_1, X_1)$
  for $j = 1$ to $p - 1$ **do**
    $I_{j+1}^j \leftarrow \text{RBFFD}(X_{j+1}, X_j)$
    $L_{j+1} \leftarrow \text{RBFFD}(X_{j+1}, X_{j+1})$
  end for
  return X, L, I, R, p
end function

4.4.2 Test Problems

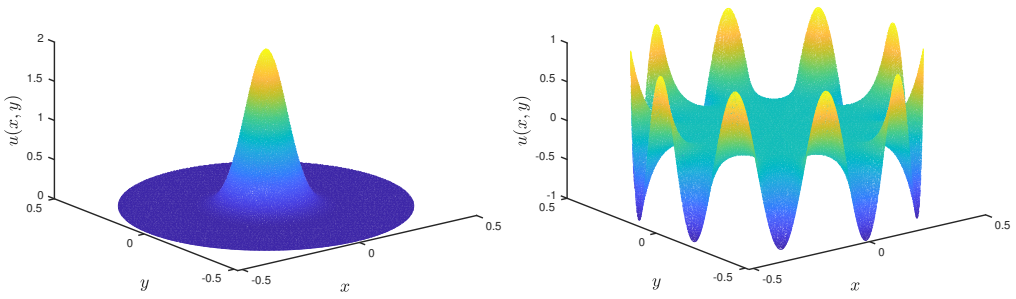
The numerical test setups have been chosen to showcase the computational efficiency of the GMM and the accuracy of the RBF-FD PHS+poly. The expectation for any multigrid or multilevel solver is that the wall clock time scales linearly with the number of nodes [10, 102, 109]. The high-order accuracy of RBF-FD PHS+poly is expected to be dictated by the degree of the augmented polynomials as shown, e.g., in [6].

For each test, the node size and polynomial degree will vary. The polynomial degree m_L ranges from 2 to 8. The other parameters for the discretized Laplacian operators are as follows: the PHS order is 3, the stencil size is 2ℓ where $\ell = (m+1)(m+2)/2$ is the number of monomial terms in a bivariate polynomial of degree m. The interpolation operators, I, are constructed using polynomials of degree 0, a degree 1 PHS, and a stencil size of 5. The parameters for the interpolation operators are kept fixed for all choices of m_L. Finally, the multilevel solver settings are defined as $(\nu_1, \nu_2, tol_1, tol_2, i_{max}, i_{conv}) = (2, 1, 10^{-16}, 10^{-1}, 50, 5)$ for both test problems. All reported wall clock times include only the execution of the multilevel schemes. They do not include those actions which are considered pre-processing, including the generation of the coarse levels and coarse level operators[7].

Two different problems are solved using variable density node sets in order to test the applicability of the proposed node subsampling strategy. The first problem considered is the Poisson problem for which $g = 0$ and $f = 200e^{-100r}\left(100r - 1\right)/r$ such that the solution given in polar coordinates is $u(r, \theta) = 2\exp(-r/0.01)$, while the other is a Laplace problem (i.e. $f = 0$) for which $g = \cos(10\theta)$ such that the solution given is $u(r, \theta) = 1024\cos(10\theta)r^{10}$ (see Figure 4.6).

The node sets for each example have been generated with variable densities based on the characteristics of their respective solutions, as illustrated in Figure 4.7, where $N_{\min} = 60$ for the Poisson problem and $N_{\min} = 120$ for the Laplace problem. The node densities in this study are

[7]The calculation of all RBF-FD operators is considered a preprocessing step which can be fully parallelized



(a) $u(r,\theta) = 2\exp(-r/0.01)$ (b) $u(r,\theta) = 1024\cos(10\theta)r^{10}$

Figure 4.6: The analytical solutions used for the (a) Poisson and (b) Laplace test problems.

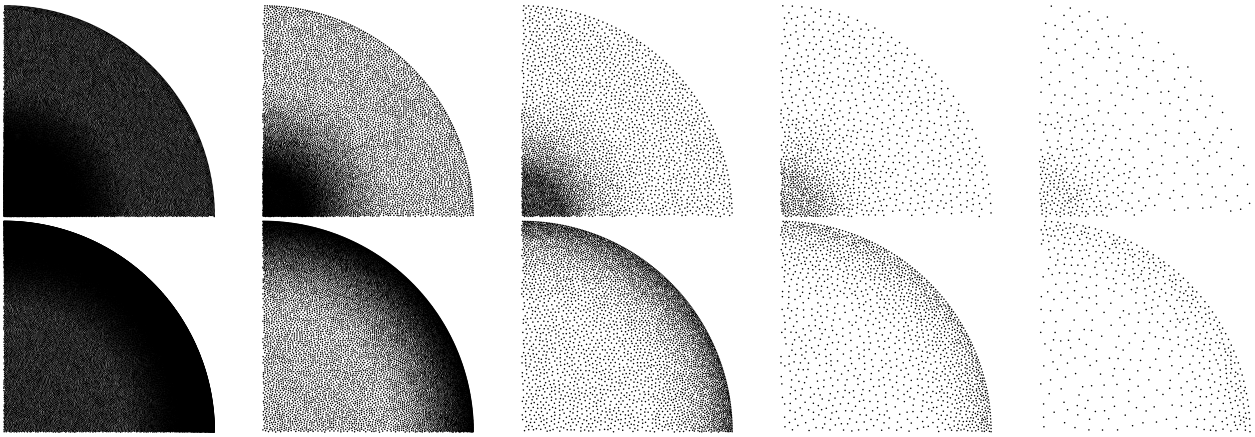


Figure 4.7: Example of the multilevel node subsampling process for the Poisson problem node set (top) and the Laplace problem node set (bottom). Only nodes within the first quadrant of the Cartesian coordinate system are shown.

defined by a linear transition between two prescribed node densities as,

$$\rho(d) = \begin{cases} \rho_1, & d < d_{\text{lim}} \\ \rho_1 + (\rho_2 - \rho_1)(d - d_{\text{lim}})/d_{\text{bl}}, & d_{\text{lim}} \leq d \leq d_{\text{lim}} + d_{\text{bl}} \\ \rho_2, & \text{otherwise} \end{cases} \tag{4.3}$$

where $d = ||\boldsymbol{x}||_2$ is the distance to the origin according to Figure 4.6, ρ_1 is the node density in region 1, d_{lim} is a distance within which ρ_1 is kept constant, whereas d_{bl} is the distance over which ρ_1 linearly blends into ρ_2. The nodes in the vicinity of boundary have been adjusted by means of repulsion only at the finest node set level [36].

4.4.2.1 Poisson Equation Test Problem

First, it can be seen from Figure 4.8 that the wall clock time[8] scales linearly with the number of nodes as expected. Furthermore, the maximum relative error ($||u - u_h||_\infty / ||u||_\infty$) decreases as

[8]For these tests, the algorithm achieved a relative residual within the prescribed tolerance ($tol = 10^{-16}$) before the maximum number of iterations was reached. The latter would produce a trivially linear plot of wall clock time versus number of nodes.

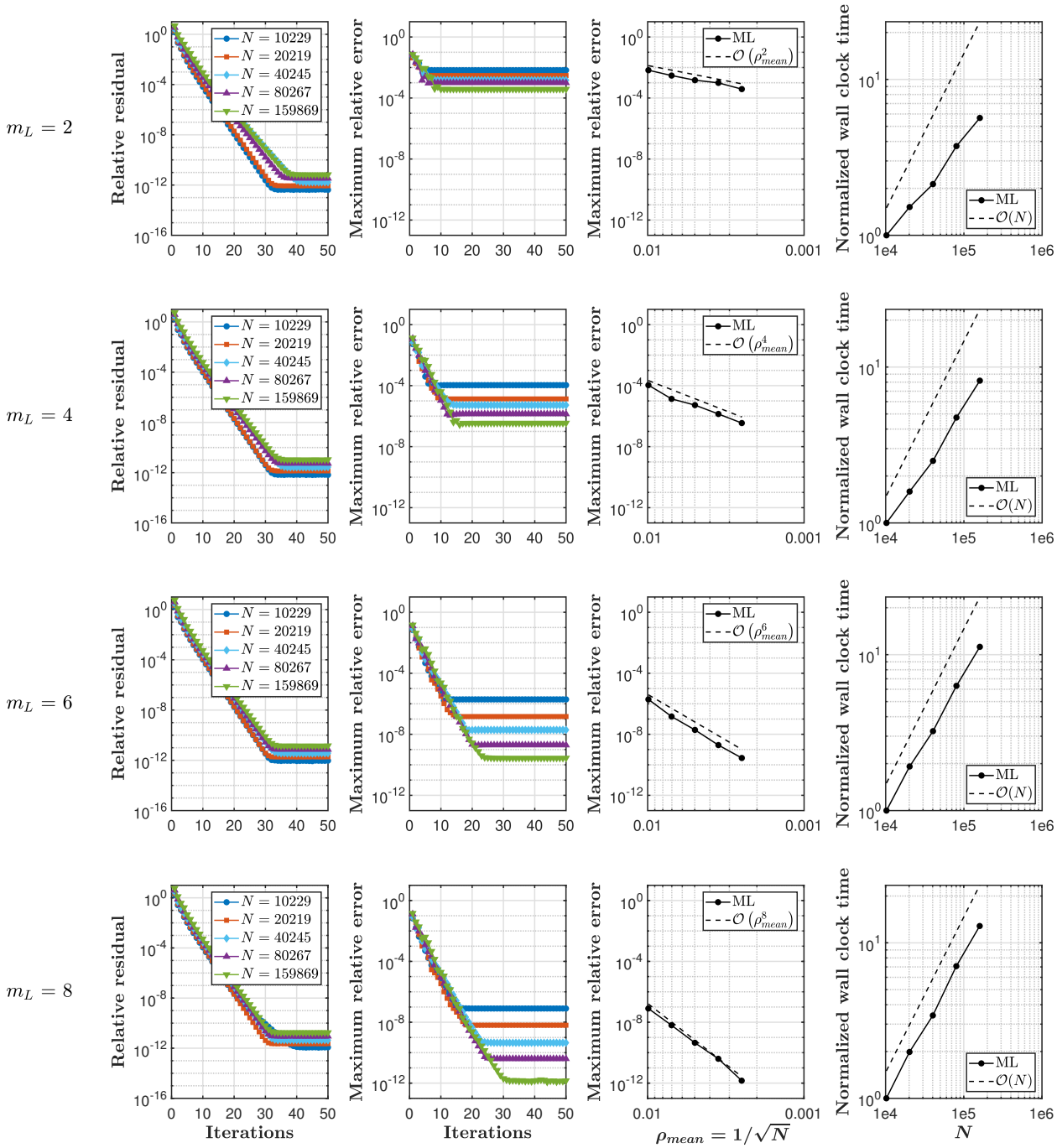


Figure 4.8: Poisson problem performance indicators of the implemented GMM for polynomial degrees of $m_L = \{2, 4, 6, 8\}$ from top to bottom. The mean node density is defined as $\rho_{mean} = 1/\sqrt{N}$.

function of node set resolution ($\rho_{mean} = 1/\sqrt{N}$) and the slope is dictated by the polynomial degree of the difference operator, m_L. This is in agreement with previous RBF-FD studies [6].

Finally, the proposed multilevel solver should provide solutions without any directional bias. Hence, to identify whether any directional bias is present, the relative errors for all node set resolutions have been normalized and depicted in Figure 4.9. Thus, the scale factors used in Figure 4.9 refer to the plateau of the maximum relative error plots in Figure 4.8. As no particular directional pattern is noticed in Figure 4.9, it can be concluded that the subsampling process used for setting up the multilevel solver does not introduce any directional bias.

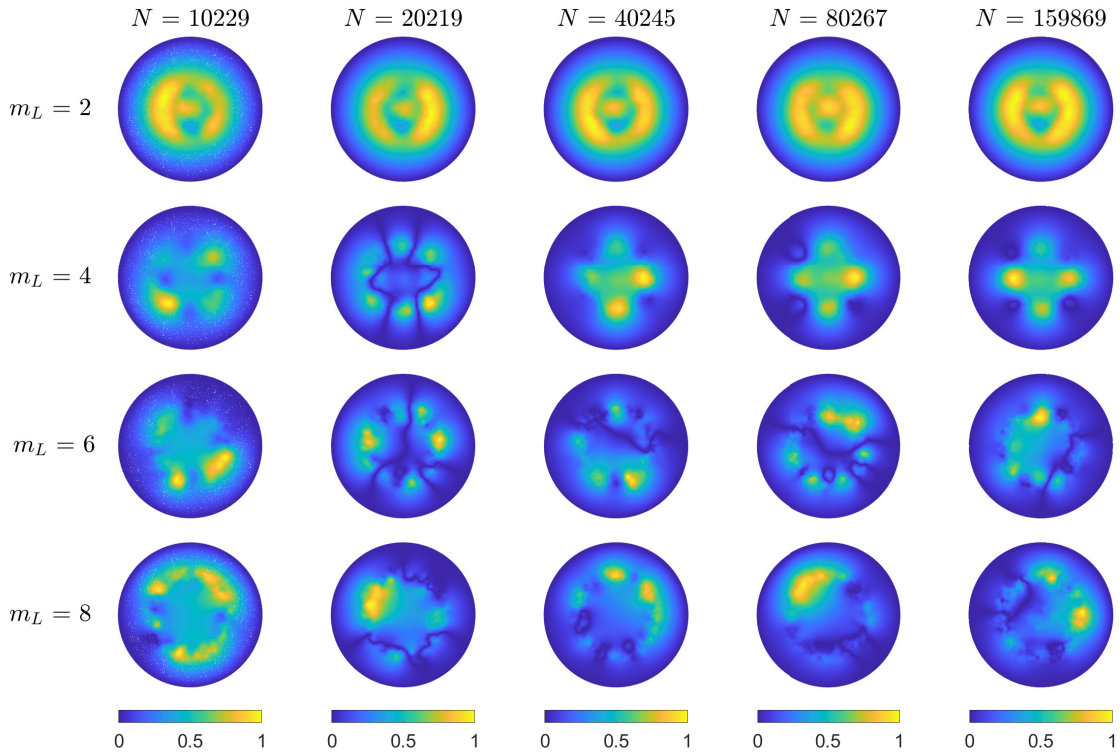


Figure 4.9: Normalized relative error distributions for various orders of the difference operators and node set resolutions for the Poisson problem. The color scale factors refer to the plateau of the maximum relative error plots in figure 4.8.

4.4.2.2 Laplace Equation Test Problem

The performance indicators of the multilevel solver for the Laplace problem are illustrated in Figure 4.10. The same overall conclusions that were made for the Poisson problem can be made for the Laplace problem, i.e. high-order accuracy and linear scaling of the computation time[9].

For $m_L = 8$, note that the convergence factor ($||r_1^i||_2/||r_1^{i-1}||_2$) of the multilevel solver is less for $N = 14419$ and $N = 28279$ compared with the other values of N. The decrease in convergence factors is most likely caused by a stencil size that is too large as compared to the relatively low node density near the boundary at the most coarse level since localized error peaks are present for both $N = 14419$ and $N = 28279$ ($m_L = 8$) in Figure 4.11. This decrease in convergence factor does not occur if the node resolution is increased to $N = 55966$ or above. With respect to the wall clock times for $N = 14419$ and $N = 28279$ given $m_L = 8$, the times are similar to those for $m_L = 6$ despite the lower rate of convergence. This similarity is due to fact that for $N = 14419$ and $N = 28279$ and $m_L = 6$ the relative residuals achieve the prescribed tolerance just before the maximum number of iterations is reached. Alternately, for $N = 14419$ and $N = 28279$ and $m_L = 8$, the maximum number of iterations is reached[10].

The normalized relative error distributions in Figure 4.11 illustrate that no directional bias seems to be introduced by the subsampling process, which is the same conclusion as for the Poisson problem.

[9] For these tests, the algorithm achieved a relative residual within the prescribed tolerance ($tol = 10^{-16}$) before the maximum number of iterations was reached except in the case of $m_L = 8$. For $m_L = 8$ and $N = 14419$ or $N = 28279$, the maximum number of iterations was reached before the relative residuals reached the prescribed tolerance.

[10] In these examples, the maximum number of iterations was 50.

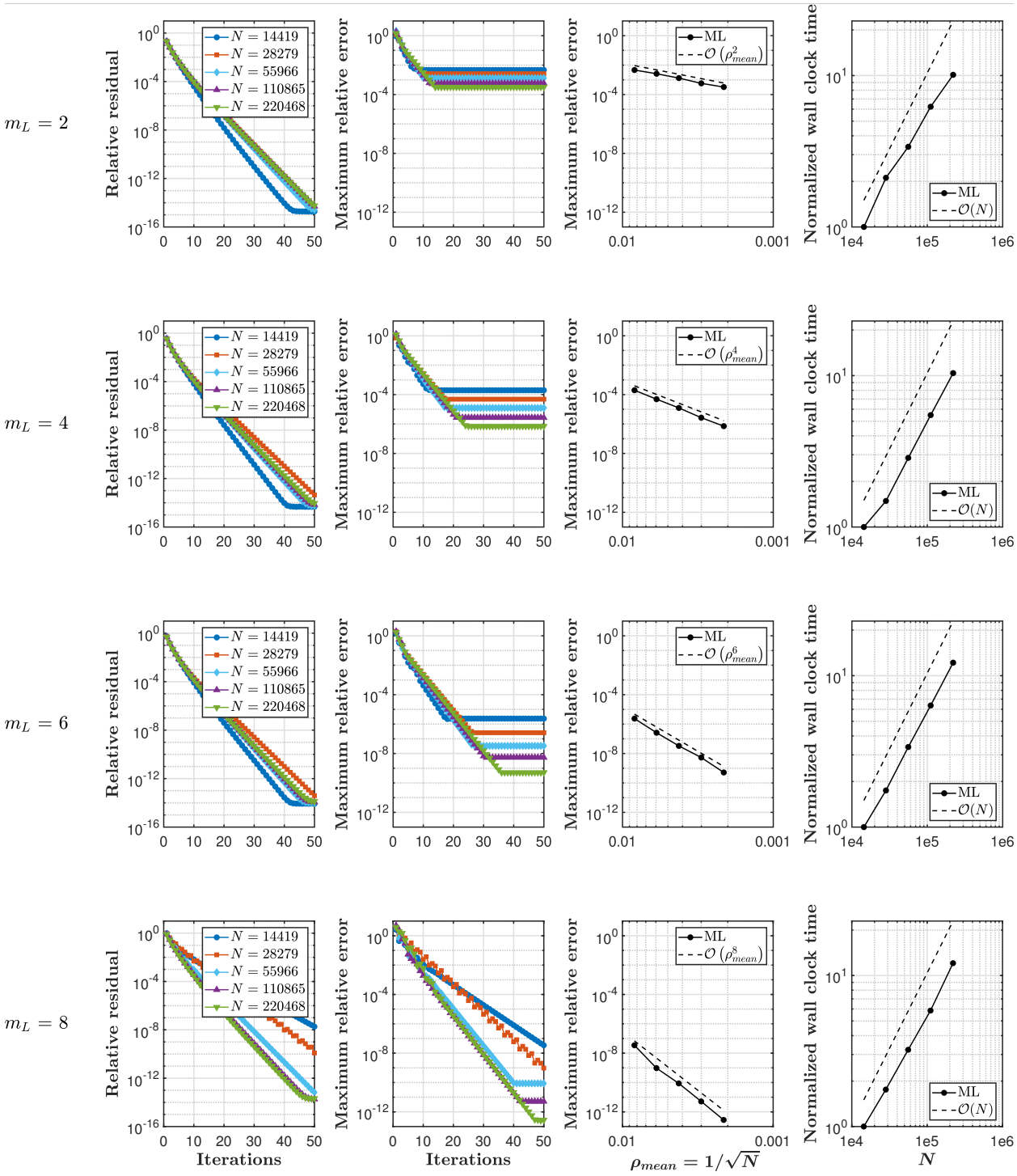


Figure 4.10: Laplace problem performance indicators of the implemented GMM for polynomial degrees of $m_L = \{2, 4, 6, 8\}$ from top to bottom. The mean node density is defined as $\rho_{mean} = 1/\sqrt{N}$.

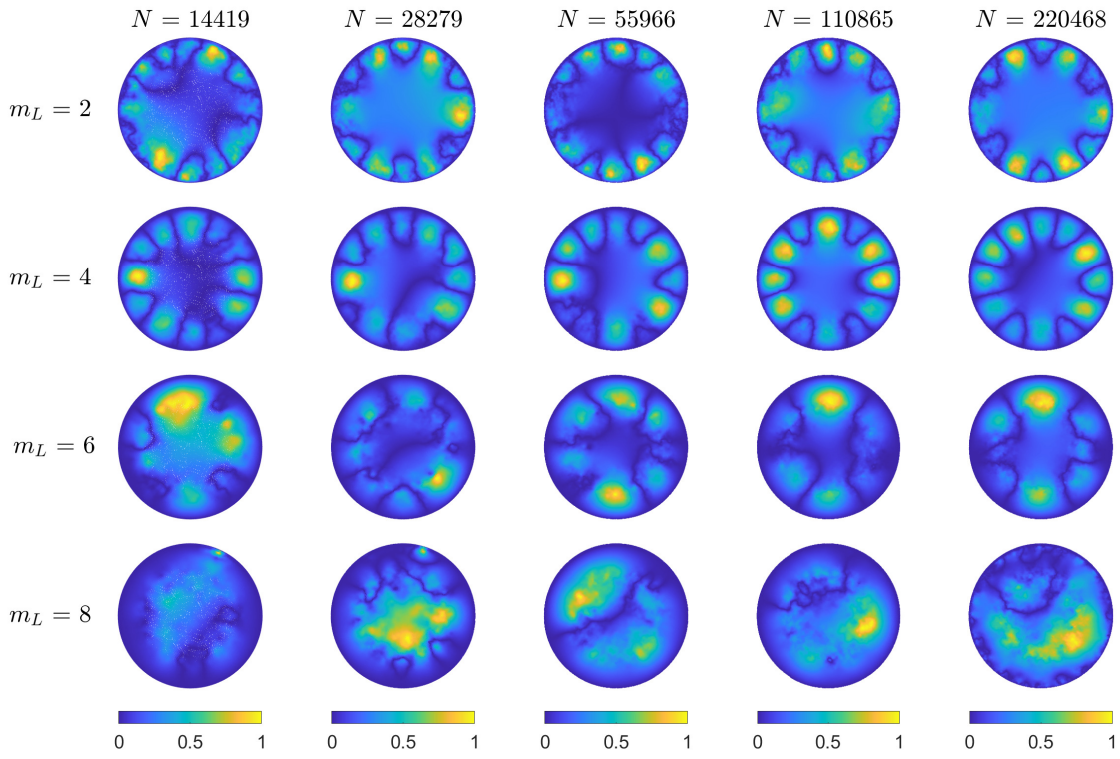


Figure 4.11: Normalized relative error distributions for various orders of the difference operators and node set resolutions for the Laplace problem. The color scale factors refer to the plateau of the maximum relative error plots in figure 4.10.

4.5 Conclusion

This chapter presents a novel node subsampling strategy for multilevel meshfree solvers, offering a robust and efficient approach to solving elliptic PDEs. The proposed method combines the flexibility of meshfree techniques with the computational cost advantages of multilevel frameworks, paving the way for future research and applications in numerical analysis and scientific computing. In particular, this method has been combined with Lagrange boundary constraints and domain decomposition method in Chapter 5.

Chapter 5

Domain Decomposition with Multilevel Meshfree Elliptic PDE Solvers using Extrapolation and Boundary Constraints

The work in this chapter was performed in collaboration with Drs. Bengt Fornberg and Morten E. Nielsen. A journal manuscript is being prepared for submission.

5.1 Introduction

5.1.1 General Background

Partial differential equations (PDEs) play a pivotal role in modeling and understanding diverse phenomena in various scientific disciplines. Further, the use of numerical methods to solve PDEs enables a much broader and more useful application of PDEs to the real world. Early and still commonly used methods for numerically solving PDEs are the finite difference method (FD), the finite element method (FEM) and the finite volume method (FVM). These methods require that the domain over which the PDE is being solved to be discretized into a finite number of nodes or elements through a process referred to as meshing. The use of meshes to discretize the computational domain of PDEs is not so simple, however. Methods which require meshing may face challenges in handling complex geometries and robust scalability [95].

Other popular methods for numerically solving PDEs are spectral element methods (SEMs) [87], discontinuous Galerkin methods (DGMs) [16], and radial basis functions-generated finite differences (RBF-FD) [38]. These methods generally provide higher-order accuracy when compared with traditional FEM or FVM. In particular, SEMs and DGMs still rely on meshed domains,

though DGMs provide suitable basis functions for more flexible discretizations [106, 88]. On the other hand, RBF-FD provides more flexibility in handling irregular geometries by avoiding a mesh entirely. Known as a meshless or meshfree method, RBF-FD discretizes the domain with scattered nodes. Such meshfree methods more easily handle nonuniform resolution requirements. RBF-FD is, as such, particularly suitable for problems involving intricate boundaries while still maintaining high-order accuracy [37, 32, 6].

5.1.2 Present Novelties

The RBF-FD method is typically implemented using node sets that are locally quasi-uniform and boundary-fitted. Such restrictions can be inconvenient in the context of time-evolutionary boundaries as any initially fitted node set may no longer fit an evolved boundary. Recent work has shown that RBF-FD extrapolation may be used to build constraints which satisfy the boundary conditions to a high-order [81]. In this method, abbreviated as RBF-FD-λ, Lagrange multipliers are introduced on the boundary to seamlessly enforce boundary conditions without the need for an adaptive method to produce interior nodes clustered near the boundaries yet carefully repelled from one another. While other methods have been recently developed to relax the need for fitted node sets [99, 100], assumptions must be made as to the continuity of the solution outside of the domain and new parameters must be tuned according to the given problem.

Being able to achieve high-order, geometrically flexible solutions to PDEs on unfitted node sets with RBF-FD-λ, an important follow-up issue is how to reduce computational cost when solving such problems. Incorporating a multilevel method into an RBF-FD solver improves the computational efficiency to $\mathcal{O}(N)$ in some cases, where N is the number of nodes in the domain. The theory of multilevel solvers has been extensively covered elsewhere and will not be reviewed in this manuscript. Further details on multilevel methods can be found in literature on the topics of multilevel approximation [31, 108] and multilevel solvers [112]. Multilevel algorithms have been employed in efficient methods for solving PDE on meshfree domains using RBF-FD [56, 109].

However, in the context of a time-dependent boundary, the computational cost of the meshfree multilevel solver no longer scales linearly with N. This increased cost is due to the fact that some of the computational costs which were otherwise overhead costs, e.g. computing the RBF-FD weights, are now recurrent costs as the domain changes. Therefore, it is desirable to introduce a decomposition method to treat the changing near-boundary domain separately from the unchanging core of the domain where a multilevel method can be used.

By decomposing the domain into a near-boundary region and a core region, the updated boundary operators only need to be recalculated and included in the linear systems for the near-boundary region. The most basic domain decomposition methods (DDMs) use overlapping subdomains and include the alternating[1] [63, 8, 79] and parallel[2] Schwarz methods [91, 30, 77]. The alternating method solves the problem on the subdomains sequentially, using the solution from the previous domain(s) to inform the solution in the next domain(s). In a corresponding manner, the parallel Schwarz method solves the problem on each subdomain simultaneously and then aggregates the solutions to produce a solution over the whole domain. Parallel DDMs introduce the possibility for the code to be parallelized thereby increasing computational efficiency, especially for large-scale problems [101, 28]. Both alternating and parallel methods update solution approximations and repeat iteratively until some convergence criteria is met. The domain decompositions present in this chapter are adaptations of the alternating and parallel Schwarz methods to RBF-FD operators.

The convergence rate for a standard Schwarz method is a function of the size of overlap between subdomains and the transmission conditions which describe how the subdomains influence one another. For elliptic problems, Schwarz algorithms work only for overlapping domain decompositions [28]. The most basic of transmission conditions include establishing artificial boundary conditions in and on the interface between subdomains. There exist optimized Schwarz methods which increase convergence by augmenting the transmission conditions [43, 28]. Non-overlapping

[1] Also referred to as the multiplicative Schwarz method and a case of general class of the successive subspace correction methods.

[2] Also known as the additive Schwarz method and a case of the general class of parallel subspace correction methods.

DDMs, also called iterative substructuring methods, are able to achieve convergence with decreased domain size and thus decreased computational cost [29, 7]. For the purposes of this chapter, all domain decompositions overlap and the transmission conditions are based on RBF-FD stencils as discussed in Section 5.3.1.

This chapter introduces a novel form of meshfree PDE solver. The discretized operators are based on RBF-FD operators (Section 5.2.1) augmented with Lagrance multiplier enforced boundary conditions (Section 5.2.2). The method involves a decomposition of the domain into a near-boundary domain and a core domain. These domains are solved according to an alternating DDM (Section 5.3.1.1). In the core domain, a multilevel method is used to approximate the solution (Section 5.3.2). In the near boundary domain, a second decomposition breaks the near-boundary domain into subdomains. An approximate solution on each near-boundary subdomain is found simultaneously using a direct solver and combined according to a parallel DDM (Section 5.3.1.2). Finally, the performance of the method on various numerical examples (Section 5.4) are discussed (Section 5.5).

5.2 Background

Let $\Omega \in \mathbb{R}^d$ be a d-dimensional domain bounded by $\Gamma \in \mathbb{R}^{d-1}$, in which we seek a solution to a general elliptic PDE[3],

$$\begin{aligned} \mathcal{L}u &= f, \quad \boldsymbol{x} \in \Omega \\ \mathcal{B}_{\mathrm{D}}u = u &= g, \quad \boldsymbol{x} \in \Gamma_{\mathrm{D}} \\ \mathcal{B}_{\mathrm{N}}u = \frac{\partial u}{\partial n} &= h, \quad \boldsymbol{x} \in \Gamma_{\mathrm{N}} \end{aligned} \tag{5.1}$$

where $u = u(\boldsymbol{x}) \in \mathbb{R}$ is the solution to (5.1) in Ω, while Γ_{D} specifies the Dirichlet boundary and $\mathbf{n}$ is the outward pointing normal vector defined on the Neumann boundary Γ_{N} as depicted in Figure 5.1. Note that the exact shape of the domain is not important (such as convexity, corners, etc.). There are no restrictions placed on the domain, including no need for convexity.

[3]While the problem is readily stated in d-dimensions, our numerical examples will be limited to $d = 2$.

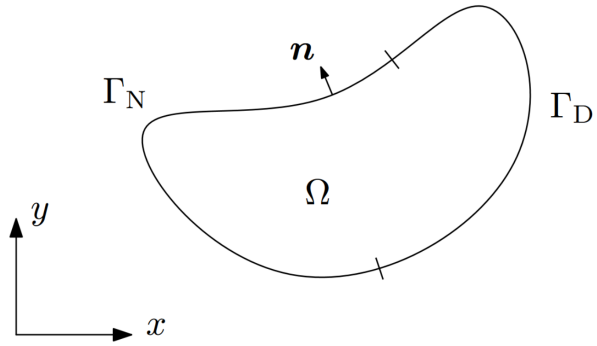


Figure 5.1: An example domain on which a PDE may be solved.

5.2.1 Radial basis function-generated finite differences

The method of RBF-FD seeks to find weights which apply to the function values at each of some $n-1$ nearest neighbors of a given center point $\boldsymbol{x}_c$. These weights approximate the effect of a general operator $\mathcal{L}$ on the solution at the center point such that

$$\mathcal{L}u(\boldsymbol{x}_c) \approx \sum_{i=1}^{n} w_i u(\boldsymbol{x}_i) \tag{5.2}$$

where $\boldsymbol{x_c} \in \{\boldsymbol{x}_i\}_{i=1}^n$. The weights in (5.2) are calculated by solving the following linear system:

$$\begin{bmatrix} \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|) & \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|) & \dots & \phi(\|\boldsymbol{x}_n - \boldsymbol{x}_2\|) \\ \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|) & \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_2\|) & \dots & \phi(\|\boldsymbol{x}_n - \boldsymbol{x}_2\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|\boldsymbol{x}_1 - \boldsymbol{x}_n\|) & \phi(\|\boldsymbol{x}_2 - \boldsymbol{x}_n\|) & \dots & \phi(\|\boldsymbol{x}_n - \boldsymbol{x}_n\|) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_1\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_2\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \vdots \\ \mathcal{L}\phi(\|\boldsymbol{x} - \boldsymbol{x}_n\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \end{bmatrix} \tag{5.3}$$

where the domain is discretized at node locations $x_1, ..., x_{N_B}$ on Γ and $x_{N_B+1}, ..., x_N$ in Ω, ϕ is a radial function, and $\|\cdot\|$ is the standard Euclidean distance formula.

A common variation of the system in (5.3) adds polynomial terms in combination with certain matching constraints. This augmented procedure is referred to as RBF-FD+Poly and, for linear polynomial terms in the 2-D case, becomes

$$\left[\begin{array}{ccc|ccc} & & & 1 & x_1 & y_1 \\ & A & & \vdots & \vdots & \vdots \\ & & & 1 & x_n & y_n \\ \hline 1 & \dots & 1 & & & \\ x_1 & \dots & x_n & & 0 & \\ y_1 & \dots & y_n & & & \end{array}\right] \left[\begin{array}{c} w_1 \\ w_2 \\ \vdots \\ w_n \\ \hline w_{n+1} \\ w_{n+2} \\ w_{n+3} \end{array}\right] = \left[\begin{array}{c} \mathcal{L}\phi(\|\boldsymbol{x}-\boldsymbol{x}_1\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \mathcal{L}\phi(\|\boldsymbol{x}-\boldsymbol{x}_2\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \vdots \\ \mathcal{L}\phi(\|\boldsymbol{x}-\boldsymbol{x}_n\|)\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \hline \mathcal{L}1\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \mathcal{L}x\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \mathcal{L}y\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \end{array}\right] \tag{5.4}$$

where A is the RBF coefficient matrix in (5.3). Only weights $w_1, w_2, ..., w_n$ should be used.

This polynomial augmented RBF-FD system also arises from a Lagrange multiplier-based minimization of the contribution from the RBF coefficient matrix A [110]. For strictly conditionally positive definite RBFs, such as polyharmonic splines (PHS), the constrained minimization problem is convex and has a unique solution leading to the important result that the polynomial part alone determines the accuracy order under node refinement [6, 5, 33, 34, 4]. This method is referred to as RBF-FD PHS+poly elsewhere in the literature. For conciseness in the remainder of this chapter, any reference to RBF-FD will be assumed to be RBF-FD PHS+poly.

The weights, which approximate the operator $\mathcal{L}$ at point $\boldsymbol{x}_c$, can then be found for each point $\boldsymbol{x}_c = \boldsymbol{x}_i$ in the domain such that $\boldsymbol{w}_c$ is the vector of weights corresponding to the stencil centered at x_c. The weight vectors can be assembled into an operator, $\tilde{L}$, which approximates $\mathcal{L}$ over the whole domain:

$$\tilde{L} = \begin{bmatrix} \boldsymbol{w_1}^T \\ \boldsymbol{w_2}^T \\ \vdots \\ \boldsymbol{w_n}^T \end{bmatrix}. \tag{5.5}$$

The RBF-FD approximated linear system can then be represented as

$$\tilde{L}\boldsymbol{u} = \boldsymbol{f} \tag{5.6}$$

where $\boldsymbol{u} = \{u_i\}_{i=1}^{\mathrm{N}}$ is a vector of function values such that $u_i = u(x_i)$ and $\boldsymbol{f} = \{f_i\}_{i=1}^{\mathrm{N}}$ where

$f_i = f(x_i)$ is a vector of the right hand side of the PDE. The set $\{\boldsymbol{x}_i\}_{i=1}^n$ contains all spatial points, including boundary points. Depending on the boundary conditions, the weights in $\tilde{L}$ may need to be adjusted. For example, Dirichlet boundary conditions are expressed by setting all of the weights on row i_b of $\tilde{L}$, which corresponds to boundary node x_{i_b}, to zero except for that weight on the diagonal which is one.

Alternatively, the node set $\{\boldsymbol{x}_i\}_{i=1}^n$ can be partitioned into interior nodes $\{\boldsymbol{x}_{\mathrm{I},i}\}_{i=1}^{N_\mathrm{I}}$, Dirichlet boundary nodes $\{\boldsymbol{x}_{\mathrm{D},j}\}_{j=1}^{N_\mathrm{D}}$, and Neumann boundary nodes $\{\boldsymbol{x}_{\mathrm{N},j}\}_{j=1}^{N_\mathrm{N}}$. Following the same derivation above for the corresponding operator of each node set, the operator $\tilde{L}$ becomes $L \in \mathbb{R}^{N_\mathrm{I} \times N}$ and dedicated boundary operators $B_\mathrm{D} \in \mathbb{R}^{N_\mathrm{D} \times N}$ and $B_\mathrm{N} \in \mathbb{R}^{N_\mathrm{N} \times N}$ are introduced such that they are the RBF-FD discretizations of the continuous boundary operators $\mathcal{B}_\mathrm{D}$ and $\mathcal{B}_\mathrm{N}$, respectively. Then the system from (5.6) becomes

$$\begin{bmatrix} L \\ B_\mathrm{D} \\ B_\mathrm{N} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_\mathrm{I} \\ \boldsymbol{u}_\mathrm{D} \\ \boldsymbol{u}_\mathrm{N} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix} \tag{5.7}$$

where $\boldsymbol{g}$ is the Dirichlet condition and $\boldsymbol{h}$ is the Neumann condition defined similarly to $\boldsymbol{f}$. Further details on the methods of RBF-FD and RBF methods in general have been discussed thoroughly in the literature [6, 32, 38].

5.2.2 RBF-FD with Lagrange Multiplier Enforced Boundary Conditions

A key choice when solving PDEs numerically is that of how to discretize the domain. The two most common choices are grid-based and quasi-uniform discretizations. Finite element methods and most RBF methods use locally quasi-uniform node sets. More specifically, they use quasi-uniform node sets that are aligned with the boundaries. However, the node sets need not conform to the boundary when Lagrange multipliers are introduced to enforce boundary constraints. Instead, a background node set that is independent of the boundary nodes can be used. An unfitted node set is simpler to generate, permits local refinement based purely on accuracy considerations, and

simplifies the handling of moving boundaries[4] . Figure 5.2 visualizes example square and butterfly domains with scattered background node sets which are independent of and not fitted to the respective boundaries. A high-accuracy implementation of RBF-FD using Lagrange multiplier enforced boundary conditions, referred to as RBF-FD-λ, on unfitted quasi-uniform node sets has been explored in [81] and is summarized here.

The Euler-Lagrange equations can be used to setup the Poisson problem in strong form with constraints [81]. The two-dimensional case is considered here for the purpose of simplicity. First, the Lagrangian density is assumed to be expressed as,

$$\mathcal{L}(x, y, \boldsymbol{u}, \boldsymbol{u}_x, \boldsymbol{u}_y) = \sum_{i=1}^{N_I} \frac{1}{2} u_{i,x}^2 + \frac{1}{2} u_{i,y}^2 + u_i f_i \tag{5.8}$$

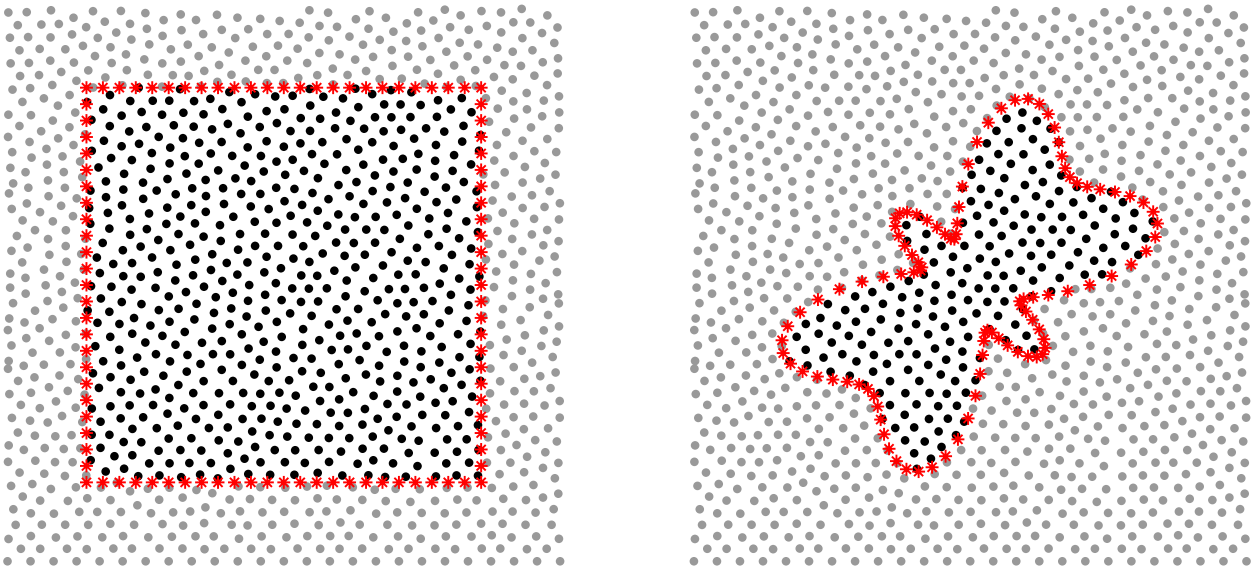
where $f_i = f(\boldsymbol{x}_i)$, $u_i = u(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}$ corresponds to the ith element of $\boldsymbol{u} \in \mathbb{R}^{N_I}$, whereas $u_{i,x} = \frac{\partial u(\boldsymbol{x})}{\partial x}\big|_{\boldsymbol{x}=\boldsymbol{x}_i}$ corresponds to the ith element of $\boldsymbol{u}_x \in \mathbb{R}^{N_I}$ and $u_{i,y} = \frac{\partial u(\boldsymbol{x})}{\partial y}\big|_{\boldsymbol{x}=\boldsymbol{x}_i}$ corresponds to the ith element of $\boldsymbol{u}_y \in \mathbb{R}^{N_I}$. In order to enforce the boundary conditions, the Lagrangian density is augmented by constraints,

$$\mathcal{L}'(x, y, \boldsymbol{u}, \boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{\lambda}_\mathrm{D}, \boldsymbol{\lambda}_\mathrm{N}) = \mathcal{L} - \sum_{j=1}^{N_\mathrm{D}} \lambda_{\mathrm{D},j} c_{\mathrm{D},j} - \sum_{k=1}^{\mathrm{N_N}} \lambda_{\mathrm{N},k} c_{\mathrm{N},k}, \tag{5.9}$$

where the constraints are defined as $c_{\mathrm{D},j} = \mathcal{B}_\mathrm{D}(u)|_{\boldsymbol{x}=\boldsymbol{x}_j} - g(\boldsymbol{x}_j)$ and $c_{\mathrm{N},k} = \mathcal{B}_\mathrm{N}(u)\big|_{\boldsymbol{x}=\boldsymbol{x}_k} - h(\boldsymbol{x}_k)$, and the jth element of $\boldsymbol{\lambda}_\mathrm{D}$ is defined as $\lambda_{\mathrm{D,j}}$ while the kth element of $\boldsymbol{\lambda}_\mathrm{N}$ is defined as $\lambda_{\mathrm{N,k}}$. The minimum of the augmented Lagrangian density can be found by solving the system of linear equations

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial u_i} - \frac{\partial}{\partial x}\left(\frac{\partial \mathcal{L}'}{\partial u_{i,x}}\right) - \frac{\partial}{\partial y}\left(\frac{\partial \mathcal{L}'}{\partial u_{i,y}}\right) = 0, \quad & \text{for } i = 1, 2, ..., N_I, \\ \frac{\partial \mathcal{L}'}{\partial \lambda_{D,j}} = 0, \quad & \text{for } j = 1, 2, ..., N_D, \\ \frac{\partial \mathcal{L}'}{\partial \lambda_{N,j}} = 0, \quad & \text{for } j = 1, 2, ..., N_N, \end{aligned} \tag{5.10}$$

[4] An independent background node set need not change as the boundary changes. Only a calculation of which nodes are in and which are out is necessary.



(a) Square domain: simple with sharp corners.

(b) Butterfly domain: smooth and non-convex.

Figure 5.2: Two example domains for which the background scattered node sets are independent of the boundaries. Each background node set, represented here by circular dots, is neither aligned with nor fit to the boundary, represented here by asterisks. Those nodes interior to the boundaries are shown as black and those exterior to the boundaries are shown as grey.

which, after evaluating some of the derivatives and rearranging terms, can be expressed as

$$u_{i,xx} + u_{i,yy} + \sum_{j=1}^{N_D} \lambda_{D,j} \frac{\partial c_{D,j}}{\partial u_i} + \sum_{k=1}^{N_N} \lambda_{N,k} \frac{\partial c_{N,k}}{\partial u_i} = f_i, \qquad \text{for } i = 1, 2, ..., N_I$$

$$\mathcal{B}_D(u)\big|_{\boldsymbol{x}=\boldsymbol{x}_j} = g(\boldsymbol{x}_j), \text{for } j = 1, 2, ..., N_D, \tag{5.11}$$

$$\mathcal{B}_N(u)\big|_{\boldsymbol{x}=\boldsymbol{x}_k} = h(\boldsymbol{x}_k), \text{for } k = 1, 2, ..., N_N.$$

The last step is to introduce a discretization of $u(\boldsymbol{x})$ using N_I collocation nodes in Ω, and insert it into (5.11), which finally ends up with the system of linear equations

$$\begin{bmatrix} L & B_{\mathrm{D}}^T & B_{\mathrm{N}}^T \\ B_{\mathrm{D}} & O_{\mathrm{D,D}} & O_{\mathrm{D,N}} \\ B_{\mathrm{N}} & O_{\mathrm{N,D}} & O_{\mathrm{N,N}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{\mathrm{I}} \\ \boldsymbol{\lambda}_{\mathrm{D}} \\ \boldsymbol{\lambda}_{\mathrm{N}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix} \tag{5.12}$$

$$\Lambda \boldsymbol{u} = \boldsymbol{F} \tag{5.13}$$

where $O_{\mathrm{D,N}}$ is a zero matrix of size $\mathrm{N_D} \times \mathrm{N_N}$, while $\boldsymbol{\lambda}_{\mathrm{D}} \in \mathbb{R}^{\mathrm{N_D}}$ and $\boldsymbol{\lambda}_{\mathrm{N}} \in \mathbb{R}^{\mathrm{N_N}}$ are the Lagrange multipliers. The matrix L is the same as above. The linear system in (5.12) is similar to that of (5.7) and can be solved using any linear solver.

It should be emphasized that the discrete operators, i.e. $L \in \mathbb{R}^{\mathrm{N_I} \times \mathrm{N_I}}$, $B_{\mathrm{D}} \in \mathbb{R}^{\mathrm{N_D} \times \mathrm{N_I}}$ and $B_{\mathrm{N}} \in \mathbb{R}^{\mathrm{N_N} \times \mathrm{N_I}}$, are collocated only on interior nodes as depicted with black nodes in Figure 5.2. Thus, both the Dirichlet and Neumann boundary conditions are essentially approximated by means of extrapolation from a subset of the interior nodes using the exact same algorithm for computing RBF-FD weights. While this method allows for the accurate use of an unfitted node set, it is possible to use a boundary-fitted node set as used in traditional RBF-FD methods.

5.3 Method

The novel method presented in this chapter solves (5.1) using RBF-FD-λ operators, alternating and parallel DDMs, and a multilevel method. First, a scattered, locally quasi-uniform

background node set is generated such that it fully encompasses the domain of the PDE. Then, reduced order RBF-FD operators are calculated on the background node set and its associated coarse levels without taking the domain into consideration. In addition, the boundary is discretized and high order RBF-FD boundary operators are calculated according to the domain, i.e., with stencils centered on boundary nodes and reaching into the domain. Next, the domain is decomposed into a near-boundary domain and a core domain in support of the alternating DDM. Subsequently, the near-boundary domain is further decomposed into subdomains in support of the parallel DDM. Finally, high order RBF-FD operators on the near-boundary nodes (which are permitted to reach into the core domain) are calculated and combined with the RBF-FD boundary operators into RBF-FD-λ operators.

The solution is solved for directly on each of the near-boundary subdomains, merged with solutions on the other subdomains, and updated according to the parallel DDM described in Section 5.3.1.2. Next, the solution on the core domain is approximated by a multilevel solver, as described in Section 5.3.2. The near-boundary and multilevel approximations influence one another, are synchronized, and are merged according to the alternating DDM as described in Section 5.3.1.1. The iterations are finished when the residual reaches a desired tolerance.

5.3.1 Boundary Decomposition

The domain on which the PDE is solved is decomposed twice. The first decomposition serves to preserve the computational cost of the multilevel method in the bulk of the domain. The second decomposition serves to decrease the computational cost and enable the parallelization of directly solving the PDE close to a time-dependent boundary. Consider a scattered, quasi-uniform background node set Ω_0 and a discretized boundary curve Γ which is independent of yet lies entirely within Ω_0. Let $\Omega \subset \Omega_0$ be the set of all background nodes which lie interior to Γ.

When using RBF-FD methods, the constituent stencils and weights are dependent upon the operator(s) and domain of the given PDE system. As such, for any system in which neither of those change (such as a Laplace problem with a stationary boundary), the calculation of the stencils and

weights is a pre-processing step and the computational cost is only incurred once. However, if either the operator or the domain is time-dependent[5], the RBF-FD weights and stencils may need to be recalculated each time step. Here it should be noted that the class of problems with moving boundaries serves as a primary motivation for the method presented in this chapter. However, only results for time-independent problems are considered explicitly in this chapter.

For example, given a stencil near a time-dependent boundary, if the boundary shifts to exclude one of the points from the stencil, a new stencil with new weights must be calculated. With RBF-FD-λ, only stencils that directly touch the boundary need updated weights in case the boundary moves; no interior nodes move. In general, in order to accurately resolve a moving boundary, the calculation of some RBF-FD weights will need to be done at each time step and thus cannot be considered a pre-processing step. Further, in the event that the entire interior domain is to be solved by a geometric multilevel scheme, then the coarse levels and the respective operators on those levels will also need to be recalculated at each time step. While the recalculation of the RBF-FD stencils and weights near the boundary cannot be avoided, the recurrent costs at each time step can be limited by introducing a domain decomposition.

5.3.1.1 Alternating Domain Decomposition Method

In order to keep any of the multilevel operators from changing due to an evolving-in-time domain boundary, consider the decomposition of the domain Ω into two overlapping regions: a near-boundary region Ω_1 and a core region Ω_2, where $\Omega_1 \cap \Omega_2 \neq \emptyset$ as seen in Figure 5.3. The core region is where the solution will be approximated by the multilevel method and, as such, must be chosen carefully. The near-boundary region will be solved directly[6] and thus does not have any special restrictions or constraints; it will be built to compliment the core region and kept small to reduce computational costs.

[5] The case of a time-dependent operator is not discussed in this manuscript.

[6] Technically the near-boundary region will be solved by a parallel domain decomposition within which each subdomain is solved directly, but the point remains that the construction of the near-boundary region does not need to be restricted or constrained in any way, other than to be kept small to reduce computational costs.

The core region is designed so as to be unaffected by any changes to the domain that might be caused by a moving boundary. Given that each node is directly affected by those nearest neighbors which make up its RBF-FD stencil, the core region must only contain those nodes with stencils which remain inside the boundary for all time. In practice, the dynamics of the boundary are usually not well-understood ahead of time and so a conservative estimate must be made. By constructing the core region this way, the multilevel RBF-FD operators will remain the same for all time thereby preserving the computational efficiency of the multilevel method in this region.

The near-boundary region can now be built around the core region. Given that this decomposition only involves two domains, the near-boundary region must, at minimum, contain the compliment of the core domain, Ω_2^c. However, as will be discussed later, the two regions must overlap. Therefore, layers of nearest neighbors, as determined by the background RBF-FD stencils, are added to Ω_2^c to construct the near-boundary region. Let $\mathcal{N}_k(\Omega_2^c)$ be the set of all points in Ω which are stencils nodes in the neighborhood of any and all points in $\mathcal{N}_{k-1}(\Omega_2^c)$, where $\mathcal{N}_0(\Omega_2^c) = \Omega_2^c$. Then, the final near-boundary domain $\Omega_1 = \mathcal{N}_{\delta_1}(\Omega_2^c)$ for some $\delta_1 \in \mathbb{N}$.

The boundary Γ must also be decomposed. However, the nature of the decomposition presented here is such that all boundary nodes are associated with Ω_1. Thus $\Gamma = \Gamma_1$ and $\Gamma_2 = \emptyset$.

Now that Ω has been decomposed into two domains, the near-boundary region Ω_1 on which the solution is approximated by a direct solution and the core region Ω_2 on which the solution is approximated by a multilevel solver, some scheme must be established for coordinating the solution between these two domains. An alternating DDM inspired by the alternating Schwarz method will be used here and is described in Algorithm 5. In this method the solution from one region influences the other through a transfer of information facilitated by both an overlap and the RBF-FD stencils. Any stencil which includes at least one node from either the overlapping region or the other domain contributes to this information transfer. Even if there is no overlap, the stencils closest to the boundary between regions will naturally "reach" into the adjacent region and allow for information transfer. While the solution may converge when no overlap is present[7], the

[7]For simple geometries, such as the square, the method converges for no overlap, though very slowly. For more complex

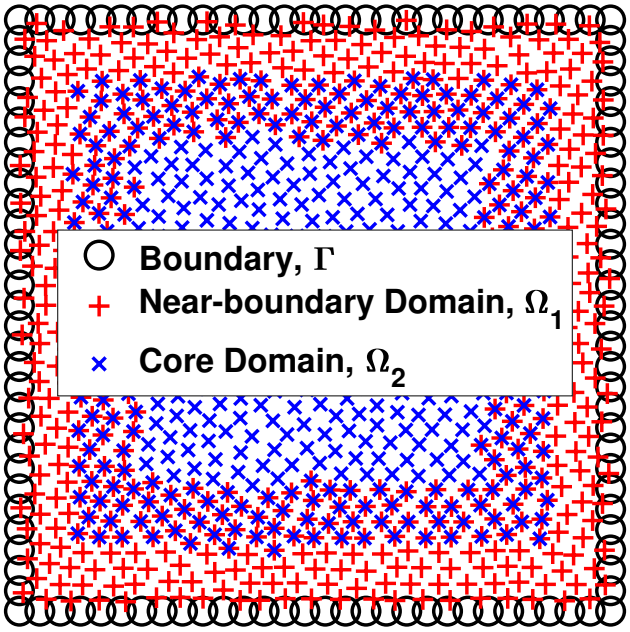


Figure 5.3: The domain decomposition of the interior node set into near-boundary and core domains.

convergence is tremendously improved by allowing the regions to overlap. Domain decompositions in which Ω_1 and Ω_2 do not overlap will not be considered presently.

Algorithm 5 Alternating Domain Decomposition Method

Input: $L1$: High order RBF-FD-λ operator on Ω_1, L_2: Reduced order RBF-FD operator on Ω_2, F_1: Right-hand side on Ω_1, F_2: Right-hand side on Ω_2, $\{\Omega_k\}_{k=1}^2$: Near-boundary and Core domains, u_0: Initial guess

Output: u: Approximate solution

$u \leftarrow u_0$

for $i = 1, 2, \ldots$ until convergence **do**

  for $k = 1, 2$ **do**

    $r \leftarrow F_k - L_k u$

    Solve $L_k c = r$

    $u \leftarrow u + c$ in Ω_k

  end for

end for

5.3.1.2 Parallel Domain Decomposition Method

Consider that a direct method is used to approximate the solution in the near-boundary region Ω_1. The cost will be $\approx \mathcal{O}(N_1^3)$ where N_1 is the number of nodes in Ω_1. The cost of approximating the solution in the near-boundary domain should ideally be balanced against the cost of the multilevel solve in the core domain, $\mathcal{O}(N_2)$ where N_2 is the number of nodes in the core region Ω_2. There is a lower bound on the size of the near-boundary region due to the limited size of the core domain (see Section 5.3.1.1). However, this lower bound may not be small enough to ensure that $N_1^3 \approx N_2$. For domain with 10,000 nodes, a balanced cost would imply $N_1 < 22$. Such a limitation is unrealistic given the size of the node set. It is made further unrealistic when considering that the near-boundary domain must overlap with the core domain, which is limited in how close it

geometries, such as the butterfly, convergence is not guaranteed.

can get to the boundary. A hypothetical $\mathcal{O}(N_1^2)$ cost of the near-boundary solve would only require limiting the near-boundary domain to fewer than 100 nodes, which is still unrealistic. Therefore, the near-boundary region will be divided into multiple subdomains and solved simultaneously to reduce computational cost.

Let $\{\tilde{\omega}_i\}_{i=1}^p$ be a set of a unique subdomains which comprise a non-overlapping decomposition of Ω_1 such that $\tilde{\omega}_i \subset \Omega_1$ for $i = 1, ..., p$, $\tilde{\omega}_i \cap \tilde{\omega}_j = \emptyset$ for all $i, j \in \{1, ..., p\}$, and $\cup_{i=1}^p \tilde{\omega}_i = \Omega_1$. As discussed in Section 5.3.1.1, simple DDMs, such as those presented here, are much aided by overlap between domains. Therefore, let $\{\omega_i\}_{i=1}^p$ be a set of non-unique subdomains which comprise an overlapping decomposition of Ω_1 such that $\omega_i \subset \Omega_1$ for $i = 1, ..., p$, $\tilde{\omega}_i \subset \omega_i$ for $i = 1, ..., p$, $\omega_i \cap \omega_j \neq \emptyset$ for all $i, j \in \{1, ..., p\}$, and $\cup_{i=1}^p \omega_i = \Omega_1$. Examples of the unique and overlapping subdomains are shown in Figure 5.4.

The construction of the unique near-boundary subdomains begins with the division of the boundary into p subsets, γ_i. Each of these subsets should be of equal or nearly equal size, and each should consist of neighboring points to preserve the local structure of the boundary. Next, for each node in the near-boundary domain $\boldsymbol{x} \in \Omega_1$, the nearest boundary node $\boldsymbol{y} \in \Gamma$ is determined. Then, the unique near-boundary subdomain $\tilde{\omega}_i$ is the set of those nodes whose nearest neighbor in the boundary also belongs to γ_i, or $\tilde{\omega}_i = \{\boldsymbol{x} \in \Omega_1 : \min_{\boldsymbol{y} \in \Gamma} \text{dist}(\boldsymbol{x}, \boldsymbol{y}) \in \gamma_i\}$.

Next, the overlapping subdomains are built by adding layers of nearest neighbors, as determined by the background RBF-FD stencils. The layers are added iteratively and consist of those nearest neighbors to a point in $\tilde{\omega}_i$ or any previous layer which are in Ω_1, but are not already in $\tilde{\omega}_i$ or a previous layer. Let $\mathcal{N}_k(\tilde{\omega}_i)$ be the set of all points in Ω_1 which are stencil nodes in the neighborhood of any and all points in $\mathcal{N}_{k-1}(\tilde{\omega}_i)$, where $\mathcal{N}_0(\tilde{\omega}_i) = \tilde{\omega}_i$. Next, let $\tilde{\gamma}_i = \{\boldsymbol{x} \in \Gamma : \exists \boldsymbol{y} \in \mathcal{N}_{\delta_2}(\tilde{\omega}_i) \text{ such that } \boldsymbol{x} = \arg\min_{\boldsymbol{z} \in \Gamma} (\text{dist}(\boldsymbol{y}, \boldsymbol{z})) \text{ and } \text{dist}(\boldsymbol{x}, \boldsymbol{y}) < 1.5\rho\}$ where ρ is the local mean node distance and $\delta_2 \in \mathbb{N}$ is the final number of layers. Then, the final near-boundary domain $\omega_i = \mathcal{N}_{\delta_2}(\tilde{\omega}_i) \cup \tilde{\gamma}_i$.

The results are subdomains which extend to overlap adjacent subdomains, but do not increase the overlap between the near-boundary and core domains. Note that the initial boundary subsets

are not necessarily fully represented in the subdomains. Only those boundary nodes which are close enough to an interior point in the subdomain are added. This may result in what seems like a poor representation of the boundary, but full inclusion of the subset boundary nodes into the subdomain negatively can result in divergence of the iterative methods.

The parallel DDM in Algorithm 6 is very similar to the restricted parallel Schwarz method. The primary difference being that this method facilitates the transfer of information between subdomains by way of RBF-FD stencils rather than boundary conditions. Nevertheless, approximate solutions are calculated on each of the p overlapping subdomains independently and combined simultaneously to approximate the solution on the entire near-boundary domain. The solution is iteratively refined by way of a residual correction. However, this method requires that the high order RBF-FD-λ operator be carefully restricted according to each subdomain.

When calculating the residual $r = F - \Lambda u$ and solving for the correction $\Lambda e = r$, it is important to recall the structure of RBF-FD operators. For the near-boundary operator Λ on domain Ω, row i corresponds to the stencil centered at $\boldsymbol{x}_i$. A nonzero entry in column j of row i is the stencil weight which applies to $u(\boldsymbol{x}_j)$. Therefore, in order to properly calculate the function values for a given set of rows, all non-zero columns from those rows must be included. Let $\Lambda_{k,*}$ represent the restricted Λ to only the rows $\{i : x_i \in \Omega_k\}$, but including all nonzero columns. For the calculation of the error, let $\Lambda_{k,k}$ represent the operator restricted to the rows $\{i : x_i \in \Omega_k\}$ and columns $\{j : x_j \in \Omega_k\}$.

Having decomposed the near-boundary domain into p subdomains, the computational cost to approximate the solution on Ω_1 has decreased. While extending $\tilde{\omega}_k$ to ω_k does increase the computational cost of each subdomain solve, the increased rate of convergence should offset most of the additional cost. However, there is no computational benefit gained by extending the overlap beyond halfway into the adjacent subdomains.

Given non-overlapping subdomains of size $N_{\tilde{\omega}} = N_1/p$, the maximum size of overlapping subdomains is thus $N_\omega = 2N_1/p$. Therefore the computational cost of approximating the solution in the near-boundary region using Algorithm 6 is $\mathcal{O}((2N_1/p)^3)$. The subdomain solves can be

computed in parallel, however this is not guaranteed to improve performance. If the overhead communication cost of including another processor outweighs the benefit of that processor, then parallelization will not improve performance. The cost-benefit analysis of parallelization will not be considered here.

Algorithm 6 Parallel Domain Decomposition Method

Input: L: RBF-FD-λ operator on Ω_1, F: Right-hand side on Ω_1, Ω_1: Near-boundary domain, $\{\omega_k\}_{k=1}^{p}$: Overlapping subdomains, $\{\tilde{\omega}_k\}_{k=1}^{p}$: Non-overlapping subdomains, u_0: Initial guess on Ω_1

Output: unb: Approximate solution on Ω_1

$u \leftarrow u_0$

for $i = 1, 2, \ldots$ until convergence **do**

  for $j = 1, 2, \ldots, p$ **in parallel do**

    Restrict rows of L to the subdomain ω_k to produce $L_{k,*}$

    Restrict F to the subdomain ω_k to produce $F_{k,*}$

    $r \leftarrow F_{k,*} - L_{k,*}u$

    Restrict rows and columns of L to the subdomain ω_k to produce $L_{k,k}$

    Solve $L_{k,k}c = r$

    $u \leftarrow u + c$ in ω_k

  end for

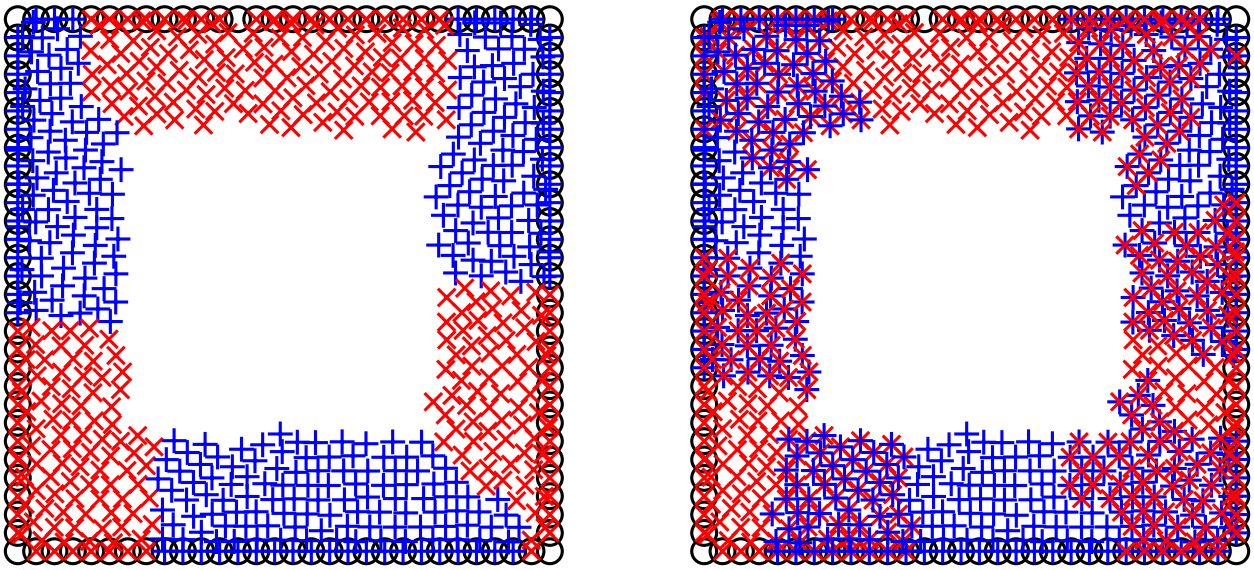
  Restrict subdomain solutions to non-overlapping subdomains

  Synchronize and merge restricted subdomain solutions into u on Ω_1

end for

5.3.2 Multilevel Method

While the near-boundary region is solved by a seris of direct solvers within in a parallel DDM, the core domain is solved using a multilevel solver. The meshfree geometric multilevel solver discussed here is identical to that presented by the authors previously [56]. It was demonstrated



(a) Unique near-boundary subdomains.

(b) Overlapping near-boundary subdomains.

Figure 5.4: Examples of the decomposition in the near-boundary region. The figure on the left shows the unique domains which do not overlap and the figure on the right shows the domains with overlap. The colors and markings are included only to distinguish between regions.

to solve PDEs on 2-D domains with the expected high order accuracy of RBF-FD PHS+poly and the low $\mathcal{O}(N)$ computational time of a multilevel method. For this solver, the background node set is coarsened according to the subsampling scheme found in [56]. No Krylov subspace methods are used to increase the rate of convergence [47]. Furthermore, the coarse grid difference operators are computed explicitly using RBF-FD on each node set level. Finally, all restriction operations are performed as injection (i.e. directly using values from the fine node set).

Given that $\Gamma_2 = \emptyset$, the discretized operators on the core domain are the RBF-FD PHS+poly operators and not RBF-FD-λ operators. Additionally, the RBF-FD matrices in the multilevel scheme can be significantly "degraded" and still achieve sufficiently accurate approximations for the overall method to converge. The exact details of the reduced order RBF-FD operators can be found in Section 5.4.1.

5.3.3 Stopping Criteria

The stopping criteria for the method is based on both the relative change in the solution $\frac{\|u^{[i]}-u^{[i-1]}\|_\infty}{\|u^{[i]}\|_\infty}$ and the relative change in residual $\frac{\|r^{[i]}-r^{[i-1]}\|_\infty}{\|f\|_\infty}$ where $r^{[i]} = f - Lu^{[i]}$. When either value is reduced below a given tolerance, the method is considered to have sufficiently converged. If the method exceeds a prescribed maximum number of iterations, the method is considered to have failed to converge. If either the relative change in solution or the residual grow for too many consecutive iterations, then the method is considered to have diverged.

5.4 Numerical Example

Let $\Omega \in \mathbb{R}^2$ be a 2-dimensional domain bounded by $\Gamma \in \mathbb{R}^1$, in which we seek a solution to the Poisson problem[8],

$$\begin{aligned} \nabla^2 u &= f, \quad \boldsymbol{x} \in \Omega \\ \mathcal{B}_{\mathrm{D}} u = u &= g, \quad \boldsymbol{x} \in \Gamma_{\mathrm{D}} \end{aligned} \tag{5.14}$$

[8]While the problem is readily stated in d-dimensions, our numerical examples will be limited to $d = 2$.

where ∇^2 is the continuous Laplace operator and Γ_{D} specifies the Dirichlet boundary. Let

$$\begin{aligned} f =& \frac{2}{x^2+y+1} - \frac{4x^2}{(x^2+y+1)^2} - \frac{1}{(x^2+y+1)^2} - \\ & 3x^2\sin(xy)^3 + 6x^2\sin(xy)\cos(xy)^2 - 3y^2\sin(xy)^3 + 6y^2\sin(xy)\cos(xy)^2, \quad \{x,y\} \in \Omega \\ g =& \sin(yx)^3 + \log(x^2+y+1), \qquad \{x,y\} \in \Gamma_{\mathrm{D}} \end{aligned}$$

The Laplace problem is solved on two domains: the square domain defined by $\Gamma = ([0,1] \times 0,1) \cup (0,1 \times [0,1])$ and the butterfly domain defined by

$$\Gamma(\theta) = \frac{5}{32}\left(2 + \sin(2\theta) - 0.01\cos(5\theta - \pi/2) + 0.63 * \sin(6\theta - 0.1)\right).$$

The corresponding convergence plots are shown in Figures 5.5 and 5.6, respectively. Two reference lines are included in each convergence plot. First, the expected order of convergence, according to the order of RBF-FD appended polynomial, is represented by a black, dashed line. Second, the error is shown by red asterisks for a direct solution to (5.14) using MATLAB's "\" backslash or `mldivide` command and a single, high order RBF-FD-λ matrix.

5.4.1 Parameters

The background node set is constructed for a range of mean node distance values. It is then coarsened by a factor of 1.2[9] to a minimum size of 80 nodes[10] for the coarsest node set. The near-boundary region overlaps the core region by two layers of nearest neighbors. The near-boundary region is decomposed into seven subdomains. Each subdomain is extended by two layers into the adjacent subdomains.

Each of the RBF-FD operators on the core domain is calculated using zero-degree appended polynomials (constants) and a 3rd order PHS. The differential operators use a 15 point stencil at the finest level and a 7 point stencil at the coarsest level while all interpolation operators use only a 5 point stencil. The near-boundary RBF-FD operators are calculated using appended polynomials

[9] According to the MFNUS algorithm presented in [56].

[10] A minimum node size which is too low causes the method to diverge for particular mean node distances.

of order 2, 4, 6, and 8, a 3rd order PHS and a stencil size equal to twice the number of polynomial terms.

The parallel DDM and the multilevel method are each iterated once for every iteration of the alternating DDM. That is, they do not iterate until some tolerance is achieved. Each near-boundary subdomain is solved using a sparse LU solver. The multilevel method includes two iterations each of pre- and post-smoothing. The geometric multilevel solver used in this method follows a V-cycle. During the V-cycles, pre- and post smoothing operations are performed using Gauss-Seidel relaxations. At the coarsest level of the core domain a sparse LU solver is used. The alternating DDM is performed until either stopping criteria is met with a tolerance of 10^{-13}.

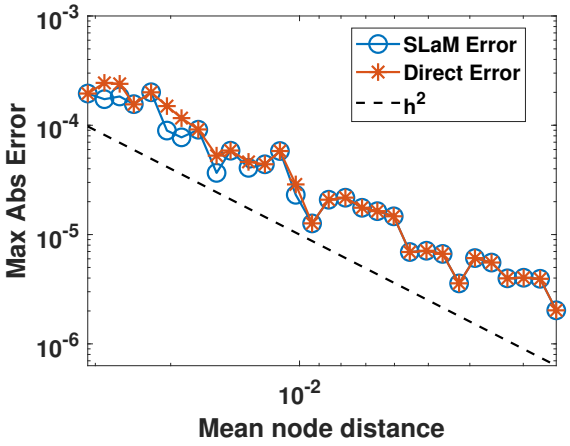
5.5 Discussion

5.5.1 Error

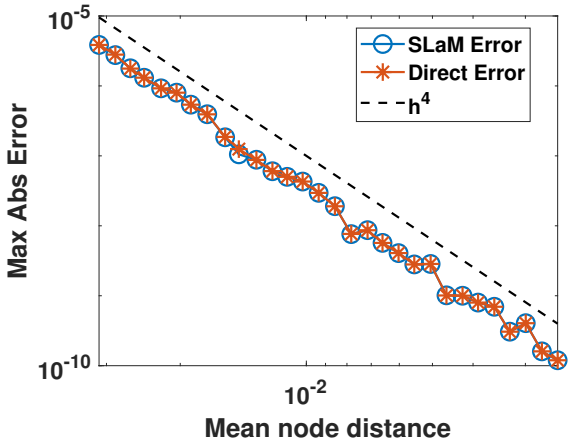
The convergence plots shown in Figures 5.5 and 5.6 demonstrate robust convergence. The method converges on simple domains with sharp corners and more complicated, non-convex domains and across a wide range of mean node distances. For each order of RBF-FD appended polynomial, the method achieves the expected order of convergence as demonstrated by the dashed, black lines in the convergence plots. The plateauing of the error at 10^{-13} is an artifact of the tolerance being set to 10^{-13}. It is interesting that the iterative method is occasionally more accurate than the direct method. This is likely due to ill-conditioning of the RBF-FD-λ matrices due to the complex geometry of the butterfly domain.

5.5.2 Computational Cost

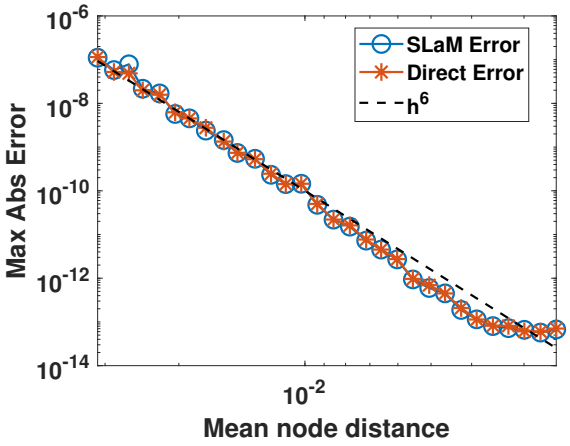
The computational cost of solving each of the near-boundary subdomains will be $\mathcal{O}(N_\omega^3)$ while the multilevel solver will be $\mathcal{O}(N_2)$, where each near-boundary subdomain contains $N_\omega \leq 2N_1/p$ nodes and the core subdomain contains N_2 nodes. The method demonstrates a slightly sublinear, $< \mathcal{O}(N)$, computational scaling with respect to the total number of nodes in the domain



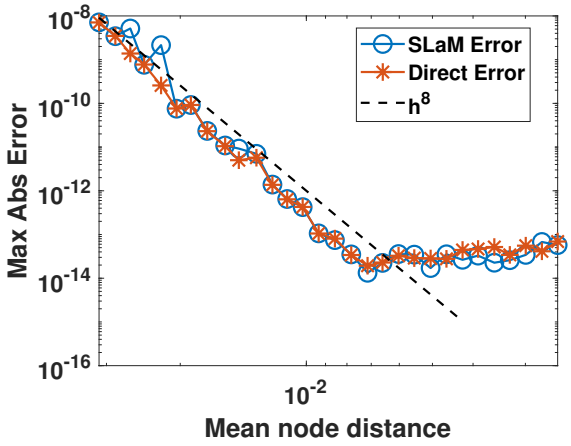
(a) RBF-FD appended polynomial of order 2



(b) RBF-FD appended polynomial of order 4

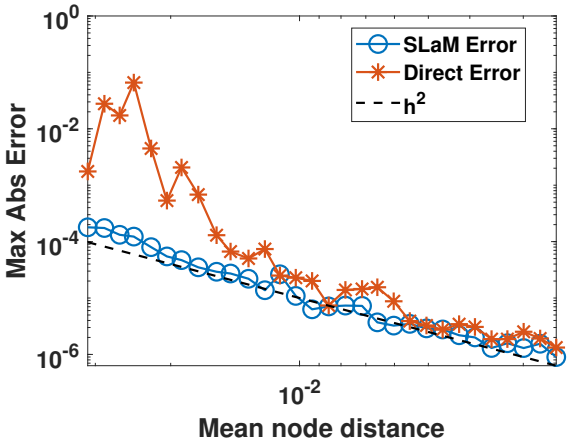


(c) RBF-FD appended polynomial of order 6

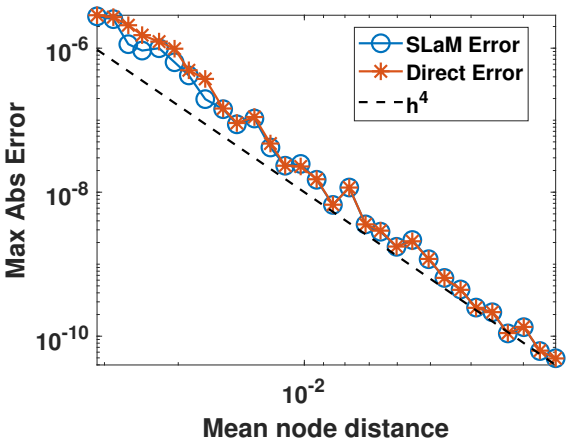


(d) RBF-FD appended polynomial of order 8

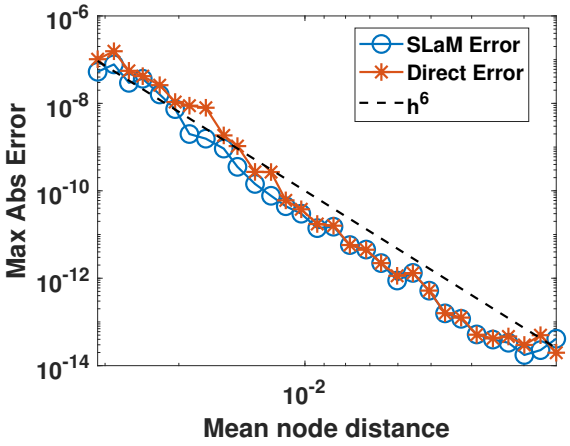
Figure 5.5: Error convergence for the method with RBF-FD appended polynomials of order 2, 4, 6, 8 when solving (5.14) on the square domain. The near-boundary region is decomposed into 7 subdomains, two layers of overlap are used between the core and near-boundary domain, and two layers of overlap are used between each near-boundary subdomain.



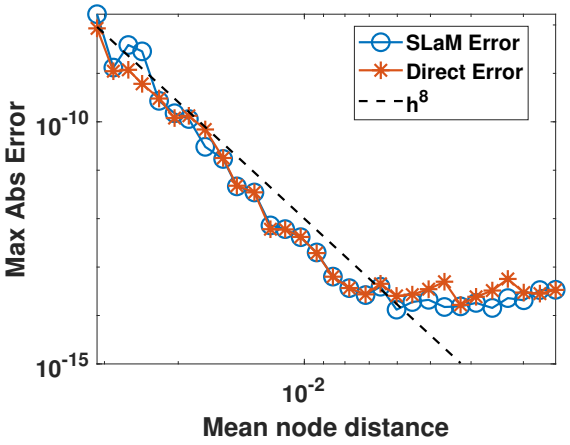
(a) RBF-FD appended polynomial of order 2



(b) RBF-FD appended polynomial of order 4



(c) RBF-FD appended polynomial of order 6



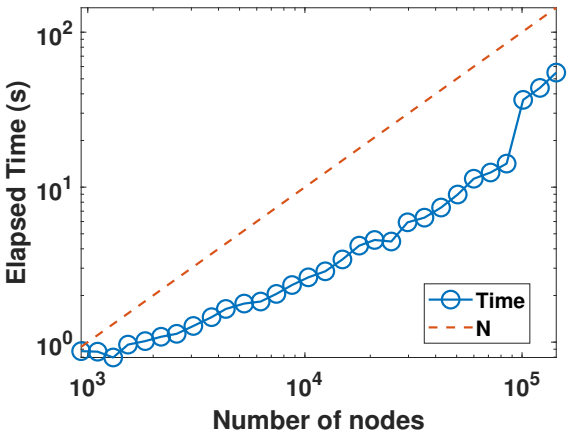
(d) RBF-FD appended polynomial of order 8

Figure 5.6: Error convergence for the method with RBF-FD appended polynomials of order 2, 4, 6, 8 when solving (5.14) on the butterfly domain. The near-boundary region is decomposed into 7 subdomains, two layers of overlap are used between the core and near-boundary domain, and two layers of overlap are used between each near-boundary subdomain.

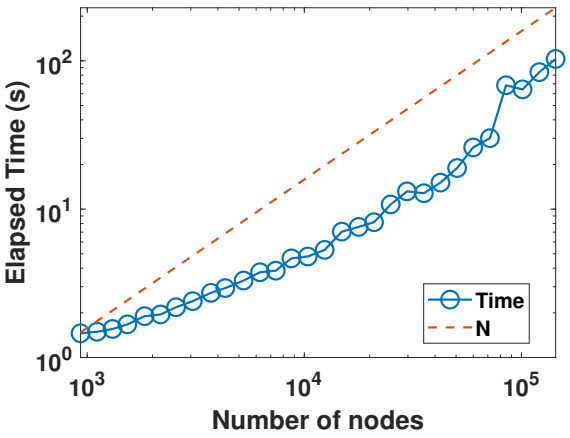
as demonstrated in Figures 5.7 and 5.8 on the square domain and the butterfly domain, respectively. It is important to note that there are upfront or overhead costs to solving these problem which are not included in the data presented here.

5.6 Conclusion

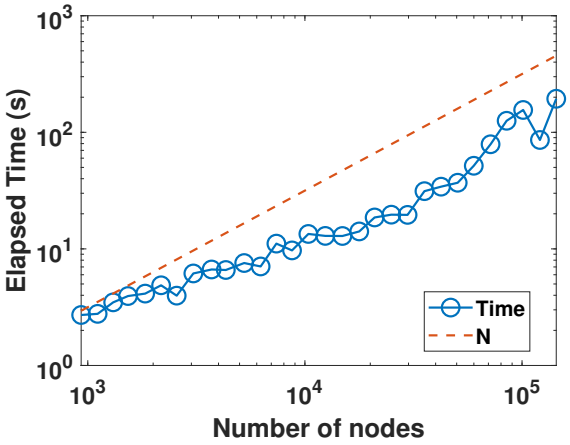
The method in this chapter was developed to serve as a precursor method to a time-dependent PDE solver with moving boundaries. In the time-independent case, the method is demonstrated to achieve high accuracy and $\mathcal{O}(N)$ computational cost. This method is tested on domains with sharp corners and non-convexity.



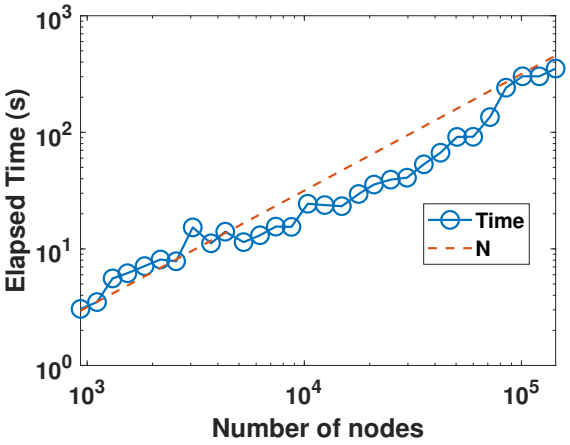
(a) RBF-FD appended polynomial of order 2



(b) RBF-FD appended polynomial of order 4

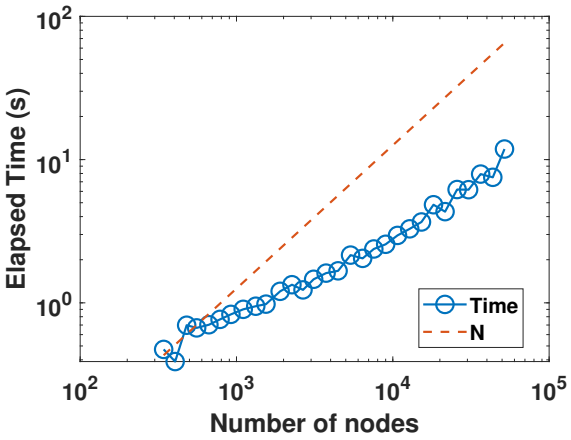


(c) RBF-FD appended polynomial of order 6

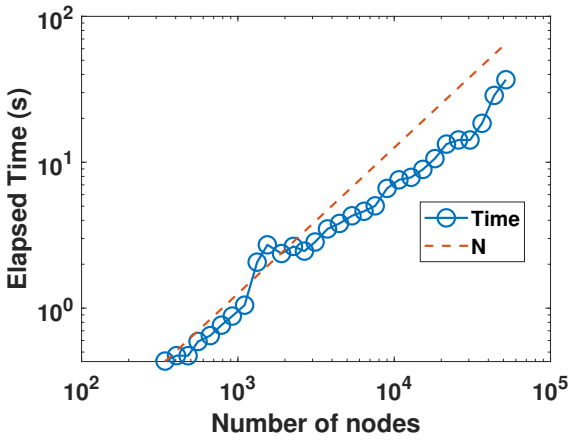


(d) RBF-FD appended polynomial of order 8

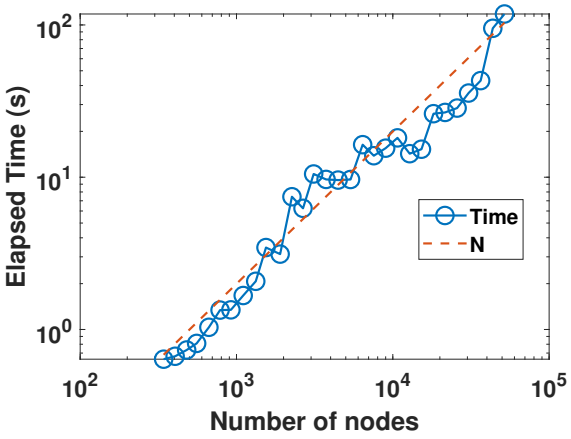
Figure 5.7: Computational cost for increasing numbers of interior nodes for the method with RBF-FD appended polynomials of order 2, 4, 6, 8 when solving (5.14) on a square domain. The near-boundary region is decomposed into 7 subdomains, two layers of overlap are used between the core and near-boundary domain, and two layers of overlap are used between each near-boundary subdomain.



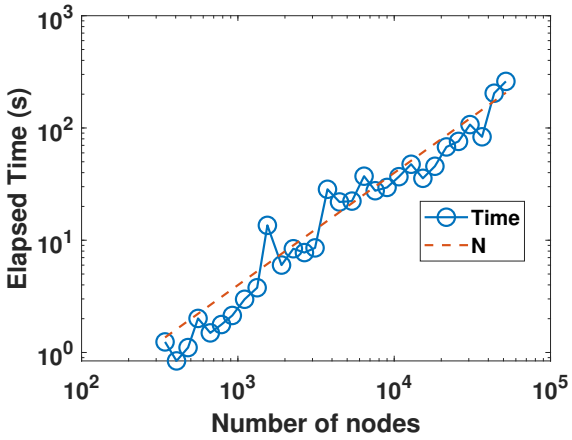
(a) RBF-FD appended polynomial of order 2



(b) RBF-FD appended polynomial of order 4



(c) RBF-FD appended polynomial of order 6



(d) RBF-FD appended polynomial of order 8

Figure 5.8: Computational cost for increasing numbers of interior nodesfor the method with RBF-FD appended polynomials of order 2, 4, 6, 8 when solving (5.14) on the butterfly domain. The near-boundary region is decomposed into 7 subdomains, two layers of overlap are used between the core and near-boundary domain, and two layers of overlap are used between each near-boundary subdomain.

Chapter 6

Future Work

This chapter contains possible directions for future research on the topics contained in Chapters 3 and 5.

6.1 Fractional Differential Equation Solvers

The method presented in Chapter 3 for calculating the FBQF has many areas ripe for further research. The limitation of the algorithm in it's current state to solve only those FODEs with initial condition $y(0) = 0$ is unsatisfactory. Therefore, more research should be done to determine the cause of this limitation and how to overcome it.

The initial nonlinear optimization step in this scheme requires the fitting of the data to a polynomial. While the change of variables renders the step size small enough that the data is well enough approximated by a polynomial (a polynomial of order 1, even), the authors suspect that a more robust fitting by generalized polynomials in this domain may improve the accuracy of the method. A better fitting may reduce the order of the monomial necessary to return optimal convergence. Separately, solving FODEs requires accounting for all data over all time, which can result in prohibitively large computational and memory costs when t is large. Therefore, in order to make the FBQF more robust for large t, further research must be done to reduce computational and memory requirements.

In the context of multi-term FODEs, some effective solution techniques result in very small values of α [20]. The FBQF scheme has demonstrated effectiveness in handling FODEs with for

small values of α. Applications of the FBQF method to FODEs with $\alpha > 1$, fractional derivatives in space, and fractional partial differential equations should also be explored.

6.2 Meshfree Multilevel PDE Solvers

There are many fruitful areas in which the method presented in Chapter 5 can be further researched. The simplest extension of this method is consider to PDEs with Neumann and Robin boundary conditions. The next most obvious extension of this method is to solving time-dependent PDEs, particularly those with evolving boundaries. Given that this case is the primary motivation for the method as it is currently presented, the extension to time-dependence is natural. Finally, it would be interesting to test this method on a more diverse array of PDEs such as Convection-Diffusion Equations or systems of PDEs such as the incompressible Navier-Stokes equations.

Bibliography

[1] T. Abdeljawad. On conformable fractional calculus. Journal of Computational and Applied Mathematics, 279:57–66, May 2015. ISSN 0377-0427. doi: 10.1016/j.cam.2014.10.016.

[2] N. H. Abel. Oplösning af et Par Opgaver ved Hjelp af bestemte Integraler (Solution of a couple of problems by means of definite integrals). Magazin for Naturvidenskaberne, pages 55–68, 1823.

[3] A. A. Alikhanov. A new difference scheme for the time fractional diffusion equation. Journal of Computational Physics, 280:424–438, Jan. 2015. ISSN 0021-9991. doi: 10.1016/j.jcp.2014.09.031.

[4] G. A. Barnett. A robust RBF-FD formulation based on polyharmonic splines and polynomials. PhD thesis, Citeseer, 2015.

[5] V. Bayona. An insight into RBF-FD approximations augmented with polynomials. Computers & Mathematics with Applications, 77(9):2337–2353, 2019.

[6] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs. Journal of Computational Physics, 332:257–273, 2017.

[7] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. I. Mathematics of Computation, 47(175):103–134, 1986.

[8] J. H. Bramble, J. E. Pasciak, J. P. Wang, and J. Xu. Convergence estimates for product iterative methods with applications to domain decomposition. Mathematics of Computation, 57(195):1–21, 1991.

[9] R. Bridson. Fast poisson disk sampling in arbitrary dimensions. SIGGRAPH sketches, 10(1): 1, 2007.

[10] W. L. Briggs, V. E. Henson, and S. F. McCormick. A Multigrid Tutorial. SIAM, second edition, 2000. doi: 10.1137/1.9780898719505.

[11] M. Cai and C. Li. Numerical approaches to fractional integrals and derivatives: A review. Mathematics, 8(1):43, Jan. 2020. ISSN 2227-7390. doi: 10.3390/math8010043.

[12] J. Cao, C. Li, and Y. Chen. High-order approximation to Caputo derivatives and Caputo-type advection-diffusion equations (II). Fractional Calculus and Applied Analysis, 18(3):735–761, 2015. ISSN 13110454. doi: 10.1515/fca-2015-0045.

[13] M. Caputo. Linear models of dissipation whose q is almost frequency independent–II. Geophysical Journal International, 13(5):529–539, Nov. 1967. ISSN 0956-540X, 1365-246X. doi: 10.1111/j.1365-246X.1967.tb02303.x.

[14] A. Cardone, D. Conte, B. Paternoster, and ,Dipartimento di Matematica, Università di Salerno, Fisciano (SA), Italy. Two-step collocation methods for fractional differential equations. Discrete & Continuous Dynamical Systems - B, 23(7):2709–2725, 2018. ISSN 1553-524X. doi: 10.3934/dcdsb.2018088.

[15] A. Cardone, D. Conte, and B. Paternoster. A MATLAB code for fractional differential equations based on two-step spline collocation methods. In A. Cardone, M. Donatelli, F. Durastante, R. Garrappa, M. Mazza, and M. Popolizio, editors, Fractional Differential Equations, pages 121–146, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-19771-6-9. doi: 10.1007/978-981-19-7716-9_8.

[16] B. Cockburn. Discontinuous Galerkin methods. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics, 83(11):731–754, 2003.

[17] R. L. Cook. Stochastic sampling in computer graphics. ACM Trans. Graph., 5(1):51–72, 01 1986. ISSN 0730-0301. doi: 10.1145/7529.8927.

[18] S. De Marchi, F. Piazzon, A. Sommariva, and M. Vianello. Polynomial meshes: Computation and approximation. In Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering, 2015.

[19] K. Diethelm. Generalized compound quadrature formulae for finite-part integrals. IMA Journal of Numerical Analysis, 17:479–493, July 1997. doi: 10.1093/imanum/17.3.479.

[20] K. Diethelm. Efficient solution of multi-term fractional differential equations using P(EC)^{m}E methods. Computing, 71(4):305–319, Nov. 2003. ISSN 1436-5057. doi: 10.1007/s00607-003-0033-3.

[21] K. Diethelm. The Analysis of Fractional Differential Equations: An Application-Oriented Exposition Using Differential Operators of Caputo Type, volume 2004 of Lecture Notes in Mathematics. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-14573-5 978-3-642-14574-2. doi: 10.1007/978-3-642-14574-2.

[22] K. Diethelm. Fundamental approaches for the numerical handling of fractional operators and time-fractional differential equations. In G. E. Karniadakis, editor, Numerical Methods, number Volume 3 in Handbook of Fractional Calculus with Applications, pages 1–22. De Gruyter, Apr. 2019. ISBN 978-3-11-057168-4. doi: 10.1515/9783110571684-001.

[23] K. Diethelm, N. J. Ford, and A. D. Freed. A predictor-corrector approach for the numerical solution of fractional differential equations. Nonlinear Dynamics, 29(1):3–22, July 2002. ISSN 1573-269X. doi: 10.1023/A:1016592219341.

[24] K. Diethelm, N. J. Ford, and A. D. Freed. Detailed error analysis for a fractional Adams method. Numerical Algorithms, 36(1):31–52, May 2004. ISSN 1572-9265. doi: 10.1023/B:NUMA.0000027736.85078.be.

[25] K. Diethelm, J. M. Ford, N. J. Ford, and M. Weilbeer. Pitfalls in fast numerical solvers for fractional differential equations. Journal of Computational and Applied Mathematics, 186 (2):482–503, Feb. 2006. ISSN 03770427. doi: 10.1016/j.cam.2005.03.023.

[26] K. Diethelm, V. Kiryakova, Y. Luchko, J. A. T. Machado, and V. E. Tarasov. Trends, directions for further research, and some open problems of fractional calculus. Nonlinear Dynamics, 107(4):3245–3270, Mar. 2022. ISSN 1573-269X. doi: 10.1007/s11071-021-07158-9.

[27] M. A. Z. Dippé and E. H. Wold. Antialiasing through stochastic sampling. SIGGRAPH Comput. Graph., 19(3):69–78, 07 1985. ISSN 0097-8930. doi: 10.1145/325165.325182.

[28] Dolean, Victorita and Jolivet, Pierre and Nataf, Frédéric. An Introduction to Domain Decomposition Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015. doi: 10.1137/1.9781611974065. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611974065.

[29] M. Dryja and O. B. Widlund. Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems. New York University, Courant Institute of Mathematical Sciences, 1989.

[30] M. Dryja and O. B. Widlund. Some domain decomposition algorithms for elliptic problems. In Iterative Methods for Large Linear Systems, pages 273–291. Elsevier, 1990.

[31] G. E. Fasshauer. Meshfree Approximation Methods with MATLAB, volume 6. World Scientific, 2007.

[32] N. Flyer, G. A. Barnett, and L. J. Wicker. Enhancing finite differences with radial basis functions: Experiments on the Navier-Stokes equations. Journal of Computational Physics, 316:39–62, 2016. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.02.078.

[33] N. Flyer, G. A. Barnett, and L. J. Wicker. Enhancing finite differences with radial basis functions: experiments on the Navier-Stokes equations. Journal of Computational Physics, 316:39–62, 2016.

[34] N. Flyer, B. Fornberg, V. Bayona, and G. A. Barnett. On the role of polynomials in RBF-FD approximations: I. Interpolation and accuracy. Journal of Computational Physics, 321:21–38, 2016.

[35] B. Fornberg. Improving the accuracy of the trapezoidal rule. SIAM Review, 63(1):167–180, Jan. 2021. ISSN 0036-1445. doi: 10.1137/18M1229353.

[36] B. Fornberg and N. Flyer. Fast generation of 2-D node distributions for mesh-free PDE discretizations. Computers & Mathematics with Applications, 69(7):531–544, 2015. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2015.01.009.

[37] B. Fornberg and N. Flyer. Solving PDEs with radial basis functions. Acta Numerica, 24: 215–258, 2015. doi: 10.1017/S0962492914000130.

[38] B. Fornberg and N. Flyer. A Primer on Radial Basis Functions with Applications to the Geosciences. SIAM, Philadelphia, PA, 2015. doi: 10.1137/1.9781611974041.

[39] B. Fornberg and A. Lawrence. Enhanced trapezoidal rule for discontinuous functions. Journal of Computational Physics, 491:112386, Oct. 2023. ISSN 00219991. doi: 10.1016/j.jcp.2023.112386.

[40] B. Fornberg and C. Piret. Complex Variables and Analytic Functions: An Illustrated Introduction. Number 165 in Other Titles in Applied Mathematics. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 2020. ISBN 978-1-61197-598-7.

[41] B. Fornberg and C. Piret. Computation of fractional derivatives of analytic functions. Journal of Scientific Computing, 96(3):79, July 2023. ISSN 1573-7691. doi: 10.1007/s10915-023-02293-4.

[42] B. Fornberg and J. A. Reeger. An improved Gregory-like method for 1-D quadrature. Numerische Mathematik, 141(1):1–19, Jan. 2019. ISSN 0029-599X, 0945-3245. doi: 10.1007/s00211-018-0992-0.

[43] M. J. Gander. Optimized schwarz methods. SIAM Journal on Numerical Analysis, 44(2):699–731, 2006.

[44] G.-h. Gao, Z.-z. Sun, and H.-w. Zhang. A new fractional numerical differentiation formula to approximate the Caputo fractional derivative and its applications. Journal of Computational Physics, 259:33–50, Feb. 2014. doi: 10.1016/j.jcp.2013.11.017.

[45] R. Garrappa. Trapezoidal methods for fractional differential equations: Theoretical and computational aspects. Mathematics and Computers in Simulation, 110:96–112, Apr. 2015. ISSN 03784754. doi: 10.1016/j.matcom.2013.09.012.

[46] R. Garrappa. Numerical solution of fractional differential equations: A survey and a software tutorial. Mathematics, 6(2):16, Jan. 2018. ISSN 2227-7390. doi: 10.3390/math6020016.

[47] A. Ghai, C. Lu, and X. Jiao. A comparison of preconditioned Krylov subspace methods for large-scale nonsymmetric linear systems. Numerical Linear Algebra with Applications, 26(1):e2215, 2019. doi: https://doi.org/10.1002/nla.2215. e2215 nla.2215.

[48] R. Gorenflo and F. Mainardi. Fractional calculus. In A. Carpinteri and F. Mainardi, editors, Fractals and Fractional Calculus in Continuum Mechanics, pages 223–276. Springer, Vienna, 1997. ISBN 978-3-7091-2664-6. doi: 10.1007/978-3-7091-2664-6_5.

[49] J. Gregory. Letter to J. Collins (23 November 1670). James Gregory Tercentenary Memorial Volume, pages 118–137, 1939.

[50] B.-Y. Guo, J. Shen, and L.-L. Wang. Generalized Jacobi polynomials/functions and their applications. Applied Numerical Mathematics, 59(5):1011–1028, May 2009. ISSN 0168-9274. doi: 10.1016/j.apnum.2008.04.003.

[51] P. M. Guzmán, G. E. Langton, L. M. Lugo Motta Bittencurt, J. Medina, and J. E. Napoles Valdes. A new definition of a fractional derivative of local type. Journal of Mathematical Analysis, Feb. 2018. ISSN 2217-3412.

[52] M. Ishteva. Properties and applications of the Caputo fractional operator. Master's thesis, Department of Mathematics, University of Karlsruhe, Karlsruhe, 2005.

[53] R. W. Kennard and L. A. Stone. Computer aided design of experiments. Technometrics, 11 (1):137–148, 1969. doi: 10.1080/00401706.1969.10490666.

[54] R. Khalil, M. Al Horani, A. Yousef, and M. Sababheh. A new definition of fractional derivative. Journal of Computational and Applied Mathematics, 264:65–70, July 2014. ISSN 03770427. doi: 10.1016/j.cam.2014.01.002.

[55] A. P. Lawrence. Subsampling algorithms. URL https://github.com/andplaw/Subsampling.

[56] A. P. Lawrence, M. E. Nielsen, and B. Fornberg. Node subsampling for multilevel meshfree elliptic PDE solvers. Computers and Mathematics with Applications, 2024.

[57] A. P. Lawrence, M. E. Nielsen, and B. Fornberg. Node subsampling for multilevel meshfree elliptic PDE solvers. Computers & Mathematics with Applications, 164:79–94, June 2024. ISSN 08981221. doi: 10.1016/j.camwa.2024.03.022.

[58] S. Le Borne and M. Wende. Multilevel interpolation of scattered data using H-matrices. NUMERICAL ALGORITHMS, 85(4):1175–1193, 12 2020. ISSN 1017-1398. doi: 10.1007/s11075-019-00860-1.

[59] C. Li and M. Cai. High-order approximation to Caputo derivatives and Caputo-type advection–diffusion equations: Revisited. Numerical Functional Analysis and Optimization, 38(7):861–890, July 2017. ISSN 0163-0563. doi: 10.1080/01630563.2017.1291521.

[60] C. Li and F. Zeng. Numerical Methods for Fractional Calculus. Chapman and Hall/CRC, 0 edition, May 2015. ISBN 978-0-429-07563-6. doi: 10.1201/b18503.

[61] C. Li, F. Zeng, and F. Liu. Spectral approximations to the fractional integral and derivative. Fractional Calculus and Applied Analysis, 15(3):383–406, Sept. 2012. ISSN 1314-2224. doi: 10.2478/s13540-012-0028-x.

[62] H. Li, J. Cao, and C. Li. High-order approximation to Caputo derivatives and Caputo-type advection–diffusion equations (III). Journal of Computational and Applied Mathematics, 299:159–175, June 2016. ISSN 0377-0427. doi: 10.1016/j.cam.2015.11.037.

[63] P.-L. Lions et al. On the Schwarz alternating method. I. In First International Symposium on Domain Decomposition Methods for Partial Differential Equations, volume 1, page 42. Paris, France, 1988.

[64] J. Liouville. Sur le calcul des differentielles á indices quelconques. Journal L'École Polytechnique, 13:71–162, 1832.

[65] J. Liouville. Mémoire Sur Quelques Questions de Géométrie et de Mécanique, et Sur Un Nouveau Genre de Calcul Pour Résoudre Ces Questions. 1832.

[66] T. Liu and R. B. Platte. Node generation for RBF-FD methods by QR factorization. Mathematics, 9(16), 2021. ISSN 2227-7390. doi: 10.3390/math9161845.

[67] C. Lubich. Runge-Kutta theory for Volterra and Abel integral equations of the second kind. Mathematics of Computation, 41(163):87–102, 1983. ISSN 0025-5718, 1088-6842. doi: 10.1090/S0025-5718-1983-0701626-6.

[68] C. Lubich. On the stability of linear multistep methods for Volterra convolution equations. IMA Journal of Numerical Analysis, 3(4):439–465, 1983. ISSN 0272-4979, 1464-3642. doi: 10.1093/imanum/3.4.439.

[69] C. Lubich. Fractional linear multistep methods for Abel-Volterra integral equations of the second kind. Mathematics of Computation, 45(172):463–469, 1985. ISSN 0025-5718, 1088-6842. doi: 10.1090/S0025-5718-1985-0804935-7.

[70] C. Lubich. Discretized fractional calculus. SIAM Journal on Mathematical Analysis, 17(3): 704–719, May 1986. ISSN 0036-1410, 1095-7154. doi: 10.1137/0517050.

[71] C. Lubich. Convolution quadrature and discretized operational calculus. I. Numerische Mathematik, 52(2):129–145, Jan. 1988. ISSN 0945-3245. doi: 10.1007/BF01398686.

[72] C. Lubich. Convolution quadrature and discretized operational calculus. II. Numerische Mathematik, 52(4):413–425, July 1988. ISSN 0945-3245. doi: 10.1007/BF01462237.

[73] C. Lubich. Convolution quadrature revisited. BIT Numerical Mathematics, 44(3):503–514, Aug. 2004. ISSN 1572-9125. doi: 10.1023/B:BITN.0000046813.23911.2d.

[74] Y. Luchko. Fractional derivatives and the fundamental theorem of fractional calculus. Fractional Calculus and Applied Analysis, 23(4):939–966, Aug. 2020. ISSN 1311-0454, 1314-2224. doi: 10.1515/fca-2020-0049.

[75] W.-H. Luo, C. Li, T.-Z. Huang, X.-M. Gu, and G.-C. Wu. A high-order accurate numerical scheme for the Caputo derivative with applications to fractional diffusion problems. Numerical Functional Analysis and Optimization, 39(5):600–622, Apr. 2018. ISSN 0163-0563. doi: 10.1080/01630563.2017.1402346.

[76] C. Lv and C. Xu. Error analysis of a high order method for time-fractional diffusion equations. SIAM Journal on Scientific Computing, 38(5):A2699–A2724, Jan. 2016. ISSN 1064-8275. doi: 10.1137/15M102664X.

[77] A. M. Matsokin and S. V. Nepomnyashchikh. The Schwarz alternation method in a subspace. Sov. Math., 29(10):78–84, 1985. ISSN 0197-7156.

[78] X. Meng, Y. Li, D. Shi, S. Hu, and F. Zhao. A method of power flow database generation base on weighted sample elimination algorithm. Frontiers in Energy Research, 10, 2022.

[79] S. G. Mikhlin. Über den Schwarzschen Algorithmus. Dokl. Akad. Nauk SSSR, n. Ser., 77: 569–571, 1951. ISSN 0002-3264.

[80] A. Narayan and D. Xiu. Constructing nested nodal sets for multivariate polynomial interpolation. SIAM Journal on Scientific Computing, 35(5):A2293–A2315, 2013. doi: 10.1137/12089613X.

[81] M. E. Nielsen and B. Fornberg. High-order numerical method for solving elliptic partial differential equations on unfitted node sets. to be submitted.

[82] K. B. Oldham and J. Spanier. The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order. Dover Books on Mathematics. Dover Publ, Mineola, NY, dover ed edition, 2006. ISBN 978-0-486-45001-8.

[83] G. M. Phillips. Gregory's Method for Numerical Integration. The American Mathematical Monthly, 79(3):270–274, 1972. ISSN 0002-9890. doi: 10.2307/2316623.

[84] F. Piazzon, A. Sommariva, and M. Vianello. Caratheodory-Tchakaloff subsampling. Dolomites Research Notes on Approximation, 10, 11 2016.

[85] C. Piret and E. Hanert. A radial basis functions method for fractional diffusion equations. Journal of Computational Physics, 238:71–81, Apr. 2013. ISSN 0021-9991. doi: 10.1016/j.jcp.2012.10.041.

[86] I. Podlubny. Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications. Academic Press, 1998.

[87] C. Pozrikidis. Introduction to Finite and Spectral Element Methods Using MATLAB. CRC press, 2005.

[88] J.-F. Remacle, J. E. Flaherty, and M. S. Shephard. An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems. SIAM review, 45 (1):53–72, 2003.

[89] S. G. Samko, A. A. Kilbas, and O. I. Marichev. Fractional Integrals and Derivatives: Theory and Applications. Gordon and Breach Science Publishers, Switzerland ; Philadelphia, Pa., USA, 1993. ISBN 978-2-88124-864-1.

[90] R. Scherer, S. L. Kalla, Y. Tang, and J. Huang. The Grünwald–Letnikov method for fractional differential equations. Computers & Mathematics with Applications, 62(3):902–917, Aug. 2011. ISSN 0898-1221. doi: 10.1016/j.camwa.2011.03.054.

[91] H. A. Schwarz. Gesammelte Mathematische Abhandlungen, volume 260. American Mathematical Soc., 1972.

[92] B. Shang, D. W. Apley, and S. Mehrotra. Diversity subsampling: Custom subsamples from large data sets. INFORMS Journal on Data Science, 2(2):161–182, 2023.

[93] J. Shen and C. Sheng. Spectral methods for fractional differential equations using generalized Jacobi functions. In G. E. Karniadakis, editor, Numerical Methods, pages 127–156. De Gruyter, Apr. 2019. ISBN 978-3-11-057168-4. doi: 10.1515/9783110571684-005.

[94] A. L. Silveira and P. J. S. Barbeira. A fast and low-cost approach for the discrimination of commercial aged cachaças using synchronous fluorescence spectroscopy and multivariate classification. Journal of the Science of Food and Agriculture, 102(11):4918–4926, 2022. doi: https://doi.org/10.1002/jsfa.11857.

[95] J. P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. J. Mavriplis. CFD vision 2030 study: a path to revolutionary computational aerosciences. Technical report, Natl. Aeronaut. Space Admin., Washington, DC, 2014.

[96] A. Sommariva and M. Vianello. Computing approximate Fekete points by QR factorizations of Vandermonde matrices. Computers & Mathematics with Applications, 57(8):1324–1336, 2009. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2008.11.011.

[97] M. Stynes. Singularities. In G. E. Karniadakis, editor, Numerical Methods, pages 287–306. De Gruyter, Apr. 2019. ISBN 978-3-11-057168-4. doi: 10.1515/9783110571684-011.

[98] P. Suchde, T. Jacquemin, and O. Davydov. Point cloud generation for meshfree methods: An overview. Archives of Computational Methods in Engineering, pages 1–27, 2022.

[99] I. Tominec and E. Breznik. An unfitted RBF-FD method in a least-squares setting for elliptic PDEs on complex geometries. Journal of Computational Physics, 436:110283, 2021.

[100] I. Tominec, P.-F. Villard, E. Larsson, V. Bayona, and N. Cacciani. An unfitted radial basis function generated finite difference method applied to thoracic diaphragm simulations. Journal of Computational Physics, 469:111496, 2022.

[101] A. Toselli and O. Widlund. Domain Decomposition Methods - Algorithms and Theory. Springer-Verlag, New York, 2004.

[102] U. Trottenberg, C. W. Oosterlee, and A. Schuller. Multigrid. Elsevier, 2000.

[103] D. Valério, M. D. Ortigueira, and A. M. Lopes. How many fractional derivatives are there? Mathematics, 10(5):737, 2022.

[104] K. van der Sande and B. Fornberg. Fast variable density 3-D node generation. SIAM Journal on Scientific Computing, 43(1):A242–A257, 2021.

[105] K. N. Volkov, V. N. Emel'yanov, and I. V. Teterina. Geometric and algebraic multigrid techniques for fluid dynamics problems on unstructured grids. Computational Mathematics and Mathematical Physics, 56:286–302, 2016.

[106] T. C. Warburton, S. J. Sherwin, and G. E. Karniadakis. Basis functions for triangular and quadrilateral high-order elements. SIAM Journal on Scientific Computing, 20(5):1671–1695, 1999.

[107] K. Watanabe, H. Igarashi, and T. Honma. Comparison of geometric and algebraic multigrid methods in edge-based finite-element analysis. IEEE transactions on magnetics, 41(5):1672–1675, 2005.

[108] H. Wendland. Scattered Data Approximation. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539.

[109] G. B. Wright, A. Jones, and V. Shankar. MGM: A meshfree geometric multilevel method for systems arising from elliptic equations on point cloud surfaces. SIAM Journal on Scientific Computing, 45(2):A312–A337, 2023.

[110] Z.-m. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. IMA journal of Numerical Analysis, 13(1):13–27, 1993.

[111] C. Yuksel. Sample elimination for generating Poisson disk sample sets. Computer Graphics Forum, 34:25–32, 5 2015. ISSN 1467-8659. doi: 10.1111/CGF.12538.

[112] R. Zamolo, E. Nobile, and B. Šarler. Novel multilevel techniques for convergence acceleration in the solution of systems of equations arising from RBF-FD meshless discretizations. Journal of Computational Physics, 392:311–334, 2019.

[113] M. Zayernouri and G. E. Karniadakis. Fractional Sturm–Liouville eigen-problems: Theory and numerical approximation. Journal of Computational Physics, 252:495–517, Nov. 2013. ISSN 0021-9991. doi: 10.1016/j.jcp.2013.06.031.

[114] M. Zayernouri and G. E. Karniadakis. Exponentially accurate spectral and spectral element methods for fractional ODEs. Journal of Computational Physics, 257:460–480, Jan. 2014. ISSN 0021-9991. doi: 10.1016/j.jcp.2013.09.039.

[115] X. Zhao and Z.-Z. Sun. Time-fractional derivatives. In G. E. Karniadakis, editor, Numerical Methods, pages 23–48. De Gruyter, Apr. 2019. ISBN 978-3-11-057168-4. doi: 10.1515/9783110571684-002.