PADOVA UNIVERSITY PRESS

Dolomites Research Notes on Approximation

Special Issue *ATMA2021 - Approximation: Theory, Methods, and Applications*, Volume 15 · 2022 · Pages 8–23

A Radial Basis Function - Finite Difference and Parareal Framework for Solving Time Dependent Partial Differential Equations

Nadun Dissanayake Kulasekera Mudiyanselage[a] · Jacob Blazejewski[b] · Benjamin Ong[b] · Cécile Piret[b]

Abstract

The Radial Basis Function–Finite Difference (RBF–FD) method is a mesh-less method for discretizing differential operators in ordinary differential equations (ODEs) and partial differential equations (PDEs). To solve a time-dependent PDE, it is common to use the method of lines approach (MOL), where one discretizes the spatial differential operator using RBFs, converting the PDE into a system of ODEs. The system of ODEs is solved using an appropriate numerical ODE solver. The RBF–FD approach has the advantage of leading to a differentiation matrix that is sparse. However, spurious eigenvalues could lead to unstable algorithms and resolving the issue could increase the computational cost. Typically, solving an ODE is usually thought of as a set of sequential steps. In this work, we propose to use a parallel ODE solver called the Parareal method that offers the ability to compute expensive steps in parallel. We also introduce a new strategy to utilize coarse and fine differentiation matrices along with coarse and finer ODE solvers in the Parareal algorithm to further speed up the computations. This strategy also notably mitigates the effects that the spurious eigenvalues have on the computations and reduces the computational cost of solving the system of ODEs with a standard approach.

1 Introduction

To construct numerical solutions to time-dependent partial differential equations (PDEs), one often applies the method of lines (MOL) approach, where the spatial derivatives are discretized, leading to a system of differential equations which are subsequently solved using standard (sequential) time integrators. A popular mesh-free approach for discretizing the spatial derivatives in complicated domains or on surfaces is to generate finite difference stencils derived from radial basis functions (RBF–FD) [1]. In contrast to global RBF approaches, the RBF–FD method formulates a "local collocation problem", leading to a sparse differentiation matrix while preserving high-order algebraic convergence [1]. Indeed, the RBF–FD has been successfully used to solve partial differential equations in many areas, including computational fluid dynamics [2], geology [3, 4, 5], and physics [6].

A significant limitation of the RBF–FD method is the time step restriction imposed by the differentiation matrix. While this is not an issue specific to the RBF–FD method (in general, one has to take increasingly smaller time steps as a spatial mesh is refined), the issue can be exacerbated depending on the choice of the collocation nodes. Further, the differentiation matrices can contain spurious eigenvalues [1, 7, 8, 9] i.e., eigenvalues located on the right half of the complex plane with positive real parts. These spurious eigenvalues can cause the RBF-FD method to be unstable and numerical approximations to grow unbounded over time. In order to resolve this issue, in instances where the positive real part is small in magnitude, one could take the time step to be significantly small so that the eigenvalues are inside the stability region. There are ODE solvers whose stability region stretches out to the right-hand side of the complex plane, such as the fourth order Runge Kutta method. However, it increases the overall computational cost and still, the pseudospectrum may not necessarily be inside the stability region. Also, one could add a very small artificial hyperviscosity term [1, 7] to the RBF-FD differentiation matrices, although too much of it could be counterproductive.

Parallel-in-time (PinT) methods have been popular in recent times due to their fast convergence properties and the ability to provide significant computational speed up. The Parareal algorithm is one such PinT algorithm developed by Lions, Maday, and Turinici in [10]. Since then the method has been used in many applications such as geology [11], fluid dynamics [12, 13] and chemistry [14]. This manuscript seeks to overcome the limitation of the RBF–FD method by using the parareal time-integration framework to control multiple time levels with varying basis and finite-difference stencils. The overall construction and intuition are straightforward: one uses a coarse finite-difference stencil (which gives rise to a differentiation matrix with desirable eigenvalues) along with a cheap time-integrator on the coarse parareal level, and a more accurate finite-difference stencil/time-integrator on the finer parareal levels.

The manuscript proceeds as follows. In Section 2, we begin with a brief introduction to the RBF method and its variants, followed by an introduction to the Parareal algorithm. In Section 3, we provide numerical results for direct implementation of the RBF-FD and parareal combination for several PDEs and then in Section 4, we present novel modifications to the RBF-FD method along with numerical results. In Section 5, we present a numerical study about how the eigenvalues of the RBF-FD differentiation matrices were affected by the new framework.

[a]Department of Mathematical Sciences, Appalachian State University, Boone, NC, 28608
[b]Department of Mathematical Sciences, Michigan Technological University, Houghton, MI, 49931

2 RBF–FD and Parareal background

In Section 2.1 we present an explanation of the Radial Basis Function–Finite Difference (RBF–FD) method as a way of discretizing differential operators in space. This is followed by Section 2.2 where we outline the details of the Parareal algorithm as a Parallel-in-time (PinT) method for solving systems of differential equations.

2.1 RBF–FD method

In the following subsections we will review how to form an interpolant using radial basis functions and show how this idea can be extended to the RBF-FD method for descritizing differential operators in space.

2.1.1 RBF interpolant

Consider the following scattered data points $f(\vec{x}_i)$, $i = 1,\dots,N$ where $\vec{x}_i$ are the locations of the data in a given coordinate system and N is the total number of nodes. For example, $\vec{x}_i$ could be ordered pairs in $\mathbb{R}^2$ or ordered triples in $\mathbb{R}^3$. We define an RBF interpolant as

$$s(\vec{x}) = \sum_{j=1}^{N} \lambda_j \phi\left(\left\|\vec{x} - \vec{x}_j\right\|\right),$$

where ϕ is a radial function (described below), and $\|\cdot\|$ is a distance function (in general, the Euclidean norm). By definition our interpolant must satisfy $s(\vec{x}_i) = f(\vec{x}_i)$ for some unknown, $\{\lambda_j\}$. These conditions give rise to the linear system $A\vec{\lambda} = \vec{f}$,

$$\begin{bmatrix} \phi(\|\vec{x}_1 - \vec{x}_1\|) & \phi(\|\vec{x}_1 - \vec{x}_2\|) & \cdots & \phi(\|\vec{x}_1 - \vec{x}_N\|) \\ \phi(\|\vec{x}_2 - \vec{x}_1\|) & \phi(\|\vec{x}_2 - \vec{x}_2\|) & \cdots & \phi(\|\vec{x}_2 - \vec{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\|\vec{x}_N - \vec{x}_1\|) & \phi(\|\vec{x}_N - \vec{x}_2\|) & \cdots & \phi(\|\vec{x}_N - \vec{x}_N\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f(\vec{x}_1) \\ f(\vec{x}_2) \\ \vdots \\ f(\vec{x}_N) \end{bmatrix}.$$

Notice that each row of A evaluates ϕ at a given central node $\vec{x}_i$ for all N nodes in the data set. So we can calculate our weights $\vec{\lambda} = A^{-1}\vec{f}$, which can then be used to form the interpolant.

There are two common classes of radial functions: infinitely smooth and piecewise smooth radial functions. The first four functions in Table 1 are examples of infinitely smooth radial functions. Each function contains a shape parameter ε, where a small ε value indicates a flatter radial function. The final two functions in Table 1 are examples of piecewise smooth radial functions, which do not include the additional ε parameter. Furthermore, piecewise smooth RBFs augmented with polynomials have been shown to reduce stagnation error; that is, the error is able to decrease to zero under node refinement [1, 15].

Table 1: There are two common types of RBFs: GA, MQ, IMQ, and IQ which are classified as infinitely smooth and all have a shape parameter ε and piecewise smooth RBFs such as PHS and TPS that do not have a shape parameter.

Type of Radial Function	Form of ϕ
Gaussian (GA)	$e^{-(\varepsilon r)^2}$
Multiquadric (MQ)	$\sqrt{1+(\varepsilon r)^2}$
Inverse Multiquadric (IMQ)	$\dfrac{1}{\sqrt{1+(\varepsilon r)^2}}$
Inverse Quadratic (IQ)	$\dfrac{1}{1+(\varepsilon r)^2}$
Polyharmonic Spline (PHS)	r^{2m+1} or $r^{2m}\log r$
Thin plate spline (TPS)	$r^2 \log r$

2.1.2 Finite difference weights of a differential operator – a global approach

Suppose that we wish to calculate $Lf(\vec{x}_i)$, $i = 1,\dots,N$, where L is a linear differential operator. Applying the operator to our interpolant from before yields

$$Ls(\vec{x}_i) = L\sum_{j=1}^{N} \lambda_j \phi\left(\left\|\vec{x}_i - \vec{x}_j\right\|\right) = \sum_{j=1}^{N} \lambda_j L\phi\left(\left\|\vec{x}_i - \vec{x}_j\right\|\right), \qquad i = 1,\dots,N$$

It is crucial to notice that an approximation of $Lf(x_i)$ is formed using the same weights as our interpolant of $f(x_i)$, but applied to $L\phi\left(\left\|\vec{x}_i - \vec{x}_j\right\|\right)$. We can now form a new linear system for the derivative of our interpolant as $L\vec{s} = B\vec{\lambda}$, where

$$B = \begin{bmatrix} L\phi(\|\vec{x}_1 - \vec{x}_1\|) & L\phi(\|\vec{x}_1 - \vec{x}_2\|) & \cdots & L\phi(\|\vec{x}_1 - \vec{x}_N\|) \\ L\phi(\|\vec{x}_2 - \vec{x}_1\|) & L\phi(\|\vec{x}_2 - \vec{x}_2\|) & \cdots & L\phi(\|\vec{x}_2 - \vec{x}_N\|) \\ \vdots & \vdots & \ddots & \vdots \\ L\phi(\|\vec{x}_N - \vec{x}_1\|) & L\phi(\|\vec{x}_N - \vec{x}_2\|) & \cdots & L\phi(\|\vec{x}_N - \vec{x}_N\|) \end{bmatrix}.$$

From our previous discussion we found $\vec{\lambda} = A^{-1}\vec{f}$, which implies

$$L\vec{s} = BA^{-1}\vec{f} = D\vec{f} \approx L\vec{f}.$$

We refer to the quantity BA^{-1} as our differentiation matrix D which discretizes the linear differential operator L. This method of approximating a differential operator is known as the RBF-Global method, because all the nodes in the data set are used to form the weights necessary for approximating $L\vec{f}$ at a node. Since we use all N nodes to form D the resulting differentiation matrix will be dense and computationally expensive for large node sets.

2.1.3 Radial basis function based finite difference (RBF-FD) method

Intuitively, approximated derivatives are not significantly enhanced with information that is far away from a central node since derivatives are inherently local. We can leverage this idea to sparsify the differentiation matrix. From our global data set of N nodes, we identify a set of n nearest neighbors for each node in the global set. This is efficiently accomplished via algorithms such as MATLAB's knnsearch. We then populate the differentiation matrix only with the weights at the nearest neighbors of a given node. This results in a much sparser differentiation matrix since for large node sets $N \gg n$.

2.1.4 Polyharmonic spline (PHS) + augmented polynomials

To avoid the nebulous task of selecting an appropriate shape parameter, a common choice for the RBF in the RBF-FD method is the polyharmonic spline (PHS),

$$\phi\left(\left\|\vec{x}-\vec{x}_j\right\|\right) = \left\|\vec{x}-\vec{x}_j\right\|^m, \qquad m \text{ is odd.}$$

This is augmented with polynomial terms to help reduce stagnation error and inevitably drive the order of convergence [1, 15, 16, 17, 18]. So our interpolant, formed with the n nearest neighbors, now looks like

$$s(\vec{x}_i) = \sum_{j=1}^{n} \lambda_j \left\|\vec{x}_i-\vec{x}_j\right\|^m + \sum_{\beta=1}^{B} \gamma_\beta P_\beta(\vec{x}_i), \qquad i = 1,\ldots,N, \tag{1}$$

where P_β is the β term of a ρ degree polynomial with a total of $B = \binom{d+\rho}{d}$ terms in $\mathbb{R}^d$ evaluated at the central node $\vec{x}_i$ and γ_β is its corresponding coefficient. For example a third degree polynomial in $\mathbb{R}^2$, where $\vec{x} = (x, y)$, is

$$P(\vec{x}) = \gamma_1 + \gamma_2 x + \gamma_3 y + \gamma_4 x^2 + \gamma_5 xy + \gamma_6 y^2 + \gamma_7 x^3 + \gamma_8 x^2 y + \gamma_9 x y^2 + \gamma_{10} y^3.$$

By construction, the interpolant, eq. (1), must satisfy

1. $s(\vec{x}_i) = f(x_i)$;
2. $\displaystyle\sum_{i=1}^{n} \lambda_i P_\beta(\vec{x}_i) = 0.$

As a general rule of thumb, we take $n = 2B$, i.e., twice as many neighbor nodes as the number of terms in the appended polynomial [1]. We can form our differentiation matrix following the same construction as before with our linear system $\vec{f} = A\vec{\lambda}$ now exhibiting the following block structure

$$\left[\begin{array}{cccc|cccc}
\phi(\|\vec{x}_1-\vec{x}_1\|) & \phi(\|\vec{x}_1-\vec{x}_2\|) & \cdots & \phi(\|\vec{x}_1-\vec{x}_n\|) & P_1(\vec{x}_1) & P_2(\vec{x}_1) & \cdots & P_\beta(\vec{x}_1) \\
\phi(\|\vec{x}_2-\vec{x}_1\|) & \phi(\|\vec{x}_2-\vec{x}_2\|) & \cdots & \phi(\|\vec{x}_2-\vec{x}_n\|) & P_1(\vec{x}_2) & P_2(\vec{x}_2) & \cdots & P_\beta(\vec{x}_2) \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\phi(\|\vec{x}_n-\vec{x}_1\|) & \phi(\|\vec{x}_n-\vec{x}_2\|) & \cdots & \phi(\|\vec{x}_n-\vec{x}_n\|) & P_1(\vec{x}_n) & P_2(\vec{x}_n) & \cdots & P_\beta(\vec{x}_n) \\
\hline
P_1(\vec{x}_1) & P_1(\vec{x}_2) & \cdots & P_1(\vec{x}_n) & & & & \\
P_2(\vec{x}_1) & P_2(\vec{x}_2) & \cdots & P_2(\vec{x}_n) & & & & \\
\vdots & \vdots & \ddots & \vdots & & & \mathbf{0} & \\
P_\beta(\vec{x}_1) & P_\beta(\vec{x}_2) & \cdots & P_\beta(\vec{x}_n) & & & &
\end{array}\right]
\left[\begin{array}{c} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \\ \hline \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_B \end{array}\right]
=
\left[\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_n \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right],$$

which can be written more compactly as

$$\left[\begin{array}{c|c} \Phi & \mathbf{P} \\ \hline \mathbf{P}^T & \mathbf{0} \end{array}\right]
\left[\begin{array}{c} \vec{\lambda} \\ \hline \vec{\gamma} \end{array}\right]
=
\left[\begin{array}{c} \vec{f} \\ \hline \vec{0} \end{array}\right]$$

The $\vec{\gamma}$ weights are discarded when populating the differentiation matrix, but are included for forming the local interpolant. The expected order of convergence of the RBF–FD method with PHS augmented with polynomials is $\mathcal{O}(h^{\rho-k+1})$, where k is the order of the differential operator being approximated [1], and h is the fill distance of the nodes.

2.2 Parareal algorithm

Consider the following initial value problem,

$$\begin{aligned} \frac{d\mathbf{u}(t)}{dt} &= \mathbf{f}(t,\mathbf{u}(t)), \qquad t \in (0,T] \\ \mathbf{u}(0) &= \alpha. \end{aligned} \tag{2}$$

If we subdivide the time domain into M intervals, $0 = T_0 < T_1 < T_2 < \cdots < T_M = T$, we can rewrite eq. (2) as a sequence of (coupled) initial value problems. For $m = 1, 2, \ldots M$,

$$\begin{aligned} \frac{d\mathbf{u}_m(t)}{dt} &= \mathbf{f}(t,\mathbf{u}_m(t)), \qquad t \in (T_{m-1},T_m] \\ \mathbf{u}_m(T_{m-1}) &= \mathbf{u}_{m-1}(T_{m-1}), \qquad \text{with } u_0(T_0) = \alpha. \end{aligned} \tag{3}$$

Here, $\mathbf{u}_m(t)$ denotes the solution in the time interval $(T_{m-1}, T_m]$. The Parareal framework decouples the system of IVPs, eq. (3), by rapidly solving (inaccurately) for the initial conditions in each time interval, and then iteratively correcting the initial conditions. For $k = 1, 2, \ldots$ and for $m = 1, 2, \ldots M$,

$$\begin{aligned} \frac{d\mathbf{u}_m^k(t)}{dt} &= \mathbf{f}\big(t,\mathbf{u}_m^k(t)\big), \qquad t \in (T_{m-1},T_m] \\ \mathbf{u}_m^k(T_{m-1}) &= \mathbf{U}_{m-1}^{k-1} \end{aligned} \tag{4}$$

where $U_0^0 = \alpha$ and $\{U_m^0\}_{m=1}^{M-1}$ are the initial conditions (also known as shooting parameters) in each time interval. Some parallel speedup can be observed if one defines and appropriately uses

1. a coarse propagation operator $\mathbf{G}(T_m, T_{m-1}, \mathbf{u}_{m-1})$, where $\mathbf{G}$ stands for the French translation of coarse: 'grossier';
2. a fine propagation operator $\mathbf{F}(T_m, T_{m-1}, \mathbf{u}_{m-1})$, where $\mathbf{F}$ stands for 'fine'.

The coarse and fine propagators approximate a solution to the differential equation $\mathbf{u}' = \mathbf{f}(t,\mathbf{u})$ at $T = T_m$ given an initial condition $\mathbf{u}(T_{m-1}) \approx \mathbf{u}_{m-1}$ for some time interval $(T_{m-1}, T_m]$.

As their names imply, the fine propagator computes a more accurate approximation of the solution than the coarse propagator. It is common to either take larger time steps or a lower-order integrator as the coarse propagator.

The Parareal algorithm is given in Algorithm 1. The algorithm begins by initializing $\mathbf{U}_0^0$ and solving eq. (4) for $\mathbf{U}_m^0$ for $m = 1, \ldots, M-1$ sequentially using the coarse propagator $\mathbf{G}$. The shooting parameters are then corrected over K iterates as outlined in the correction loop of Algorithm 1. It is worth noting that $\mathbf{F}\left(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}\right)$, $m = 1 \ldots M$ (line 8 of Algorithm 1) can be computed in parallel to provide parallel speedup.

Algorithm 1 The Parareal Algorithm

1: Set $\mathbf{U}_0^0 = \mathbf{u}(t_0)$
2: **for** $m = 1, 2, \ldots, M$ **do** ▷ Initial Coarse Approximation (0th level)
3: $\quad \mathbf{U}_m^0 = \mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^0)$
4: **end for**

5: **for** $k = 1, 2, \ldots, K$ **do** ▷ The correction loop
6: $\quad \mathbf{U}_0^k = \mathbf{u}(t_0)$
7: $\quad$ **for** $m = 1, \ldots, M$ **do**
8: $\quad\quad \mathbf{U}_m^k = \mathbf{F}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}) + \mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^k) - \mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1})$
9: $\quad$ **end for**
10: **end for**

The Parareal algorithm is assured to converge in at most $K = M$ correction iterations [19], often referred to as the finite-step convergence property of the Parareal framework. A full error analysis of the Parareal algorithm is given in [20]. However, there is no benefit to the Parareal framework if M correction iterations are used.

For efficiency, the Parareal algorithm must

1. converge with less iterations than the number of initial time intervals, i.e. $K \ll M$;
2. utilize M processors to compute $\mathbf{F}\left(T_{m+1}, T_m, \mathbf{U}_m^k\right)$, $m = 0 \ldots M-1$ (line 9) in parallel.

We overload our fine and coarse propagator function definition so that $\mathbf{F}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, M_F)$ denotes that the fine propagator takes M_F internal Euler time steps, and $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, M_G)$ denotes that the coarse propagator takes M_G internal time steps. To quantify parallel speedup, it is convenient to assume that the time subintervals are of equal length. Let γ_f and γ_c denote the computational time needed for each step in the fine/coarse propagator to solve the initial condition over one time subinterval, e.g., $\mathbf{F}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, M_F)$. Consider N_f and N_c are the number of steps for the fine and coarse propagator for $[T_m, T_{m+1}]$. Then, the parallel speedup (if M processors are used in the computation) is

$$\text{parallel speedup } = \frac{1}{(K+1)\frac{N_c}{N_f}\frac{\gamma_c}{\gamma_f} + \frac{K}{M}} \tag{5}$$

Further theoretical results regarding the computational speedup can be found in [21, 22].

3 Loose coupling of the RBF–FD + Parareal Framework

This section verifies that a loose coupling of the RBF–FD + Parareal is compatible. Specifically, the RBF-FD method with the PHS basis function/polynomial augmentation is used to discretize the spatial operators using a fixed number of spatial nodes in a method of lines approach; the resulting linear system of differential equations was solved using the Parareal algorithm. We used a combination of PHS and polynomials because it has been proven to be effective in the RBF-FD method [15, 16, 17]. Finite step convergence is exhibited regardless of the augmented polynomials degree, so we only present results for a 4th degree polynomial augmentation. In the following subsections we will explore the framework applied to three PDEs: the 2-D heat equation, the 2-D convection–diffusion equation, and the linearized shallow water equations.

3.1 2D heat equation

Consider the boundary value problem:

$$\begin{cases} \dfrac{\partial u}{\partial t} & = \Delta u + f(x,y,t), \quad (x,y)\in\Omega, \quad t\in[0,T] \\ u(x,y,t) & = g(x,y,t), \quad (x,y)\in\partial\Omega, \quad t\in[0,T] \\ u(x,y,0) & = h(x,y), \quad (x,y)\in\Omega \end{cases} \tag{6}$$

where $\Omega=[0,1]\times[0,1]$, and $t\in[0,0.2]$ and the chosen exact solution is: $u(x,y,t)=(y^3-1)(x^2-1)\,e^{-t}$.

- (Spatial Discretization): A seventh-order polyharmonic spline (PHS) augmented with a 4th degree polynomial was employed for the RBF–FD spatial discretization, with $N=1600$ equispaced nodes.
- (Time Discretization): We split the time domain into $M=100$ sub-intervals. The fine and coarse propagators are constructed using Backward Euler integrators with differing number of time steps. We present results for $M_F=\{40,80\}$ and $M_G=\{1,5\}$.

Figure 1 shows that the RBF-FD–Parareal algorithm converges in a finite number of iterations, significantly less than $M=100$. The error is computed by comparing the Parareal solution at $T=0.2$ to a reference solution generated by applying the fine propagator sequentially across the M time domains. The blue curve corresponds to a convergence study with a coarse solver consisting of one internal time step, $\mathbf{G}(T_m,T_{m-1},\mathbf{U}_{m-1}^{k-1},1)$. Since the fine solver consists of taking $M_F=40$ internal time steps, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f}=\frac{1}{40}$. Notice that since we use the same spatial discretization and the same ODE solver in both coarse and fine steps $\frac{\gamma_c}{\gamma_f}=1$. Convergence is achieved in $K\approx 14$ steps, so a theoretical speedup of approximately 2× (to machine precision) is possible, if 100 processors are used in the computation, see eq. (5). The red curve corresponds to simulations with a coarse solver consisting of five internal time steps, $G(T_m,T_{m-1},U_{m-1}^{k-1},5)$ and $M_F=80$. Now, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f}=\frac{1}{16}$. Convergence is achieved in $K\approx 7$ steps, so a theoretical speedup of approximately 1.7× (to machine precision) is possible according to eq. (5).

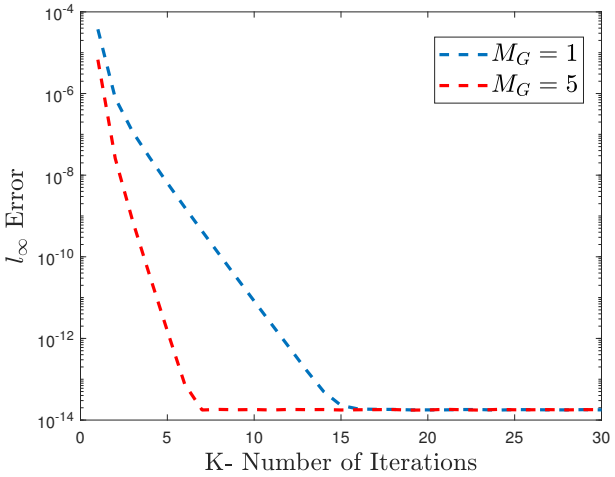

Figure 1: Parareal applied to a system of DEs, obtained by an RBF-FD discretization of the spatial operator in the 2-D heat equation. The blue curve corresponds to simulations with a coarse solver consisting of one internal time step, $\mathbf{G}(T_m,T_{m-1},\mathbf{U}_{m-1}^{k-1},1)$ and the red curve corresponds to simulations with a coarse solver consisting of five internal time steps, $\mathbf{G}(T_m,T_{m-1},\mathbf{U}_{m-1}^{k-1},5)$. In both cases, finite step convergence is observed. The more accurate coarse solver ($M_G=5$) has a superior rate of convergence, consistent with published error analysis in the literature [20].

The test case was also used to see if we can obtain the order of convergence (in space) stated in the literature when we use the Parareal algorithm. We can see from Figure 2 that the RBF–FD method yields an order of convergence consistent with the literature $\mathcal{O}(h^{\rho-k+1})$ [1], where ρ is the degree of the appended polynomial, k is the order of the differential operator in space. In fact, since the finite step convergence was already observed from Figure 1, we argue that the testing for space convergence is not

needed in general as the approximations from the parareal algorithm converge to the solution you would get from just applying the finer time integrator in the traditional way. Hence, if one knows that the RBF–FD can obtain a spatial order of convergence of $\mathcal{O}(h^{\rho-k+1})$ with a regular time integrator, showing finite step convergence will suffice to claim that the parareal method can also obtain the proper order of convergence.

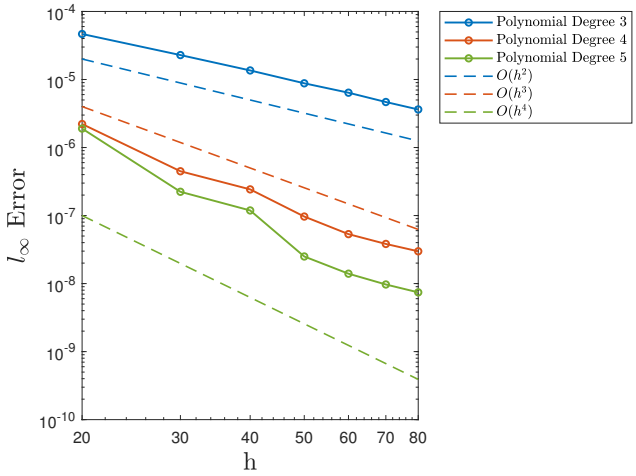

Figure 2: Spatial convergence for RBF–FD for the Parareal algorithm for $T = 0.2$. The RBF-FD method was applied to eq. (6) with varying augmented polynomials. Solid curves correspond to error decay with each polynomial degree, and the dashed curves are standard rates from $\mathcal{O}(h^2)$ to $\mathcal{O}(h^4)$. Overall, we can observe rate of convergence of $\mathcal{O}(h^{\rho-k+1})$ where, ρ is the degree of the polynomial, k is the order of the differential operator being approximated, and h is the fill distance of the nodes [1].

3.2 2D convection–diffusion equation

Consider the boundary value problem:

$$\begin{aligned} \frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} + \beta \frac{\partial u}{\partial y} &= \mu \Delta u && (x,y) \in \Omega, && t \in [0,T] \\ u(x,y,t) &= g(x,y,t), && (x,y) \in \partial\Omega, && t \in [0,T] \\ u(x,y,0) &= h(x,y), && (x,y) \in \Omega \end{aligned}$$

where $\Omega = [0,1] \times [0,1]$, and $t \in [0,T]$. The boundary and initial conditions are chosen so that the exact solution is

$$u(x,y,t) = \frac{1}{4t+1} \exp\left(\frac{-(x-\beta t-0.5)^2 - (y-\beta t-0.5)^2}{\mu(4t+1)} \right).$$

- (Spatial Discretization): A seventh-order polyharmonic spline (PHS) augmented with a 4th degree polynomial was employed for the RBF–FD spatial discretization, with $N = 1600$ equispaced nodes. The PDE parameters are chosen as $\mu = 0.01$ and $\beta = 0.8$.
- (Time Discretization): We split the time domain into $M = 100$ sub-intervals. The fine and coarse propagators are constructed using Backward Euler integrators with differing number of time steps. We present results for $M_F = 80$ and $M_G = \{1, 5\}$.

Figure 3 shows that the RBF-FD–Parareal algorithm converges in a finite number of iterations, significantly less than $M = 100$. The error is computed by comparing the Parareal solution at $T = 0.03$ to a reference solution generated by applying the fine propagator sequentially across the M time domains. The blue curve corresponds to a convergence study with a coarse solver consisting of one internal time step, $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, 1)$. Since the fine solver consists of taking $M_F = 80$ internal time steps, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f} = \frac{1}{80}$. Convergence is achieved in $K \approx 13$ steps, so a theoretical speedup of approximately $3\times$ (to machine precision) is possible, if 100 processors are used in the computation, see eq. (5). The red curve corresponds to simulations with a coarse solver consisting of five internal time steps, $G(T_m, T_{m-1}, U_{m-1}^{k-1}, 5)$ and $M_F = 80$. Now, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f} = \frac{1}{16}$. Convergence is achieved in $K \approx 7$ steps, so a theoretical speedup of approximately $1.7\times$ (to machine precision) is possible according to eq. (5).

3.3 Linearized shallow-water equations

The linearized shallow-water equations (SWEs), eq. (7), are useful for modeling turbulence when the flow depth is much smaller than the vertical scale.

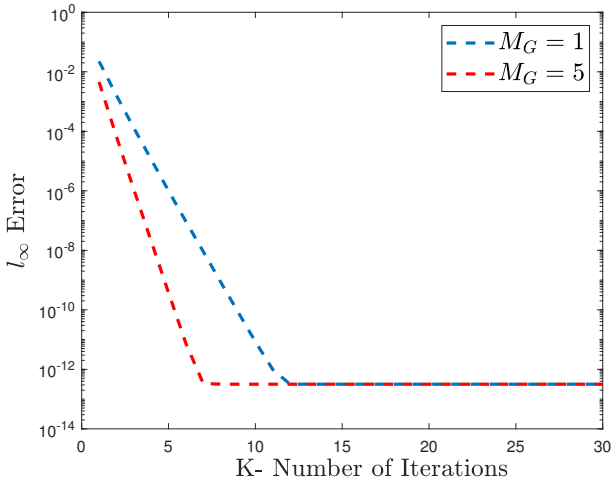

Figure 3: Parareal applied to a system of DEs, obtained by an RBF-FD discretization of the spatial operator in the 2-D convection-diffusion equation. The blue curve corresponds to simulations with a coarse solver consisting of one internal time step, $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, 1)$ and the red curve corresponds to simulations with a coarse solver consisting of five internal time steps, $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, 5)$. In both cases, finite step convergence is observed. The more accurate coarse solver ($M_G = 5$) has a superior rate of convergence, consistent with published error analysis in the literature [20].

$$\rho_t + F_x + G_y = S$$

where

- The vector of mass and momentum $\rho = (h, hu, hv)^T$.
- $h(x, y, t)$ is the total water depth and $(u(x, y, t), v(x, y, t))^T$ representing horizontal and vertical average fluid velocity.
- The flux vectors are $F = (hu, hu^2 + \frac{gh^2}{2}, huv)^T$, and $G = (hv, huv, hv^2 + \frac{gh^2}{2})^T$.
- The external force term $S = (0, -hb_x, -hb_y)^T$, and $b(x, y)$ has the information about the lower surface.
- g is the gravitational acceleration.

The linearized SWEs, eq. (7), can be obtained under the assumption that the source term S is negligible because the water surface wave elevation η is smaller than the mean water height H ($\eta \ll H$).

$$\begin{aligned} \frac{\partial \eta}{\partial t} + H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) &= 0 \\ \frac{\partial u}{\partial t} + g\frac{\partial \eta}{\partial x} &= 0, \\ \frac{\partial v}{\partial t} + g\frac{\partial \eta}{\partial y} &= 0 \end{aligned} \tag{7}$$

where η is the water surface height, H is the mean water height, (u, v) represents the horizontal and vertical fluid velocity, and g is the gravitational constant. We solve the 2D tidal wave problem [23] on the spatial domain $\Omega = [0, 1000\text{ km}] \times [0, 50\text{ km}]$. Notice $L = 1000$km. The average water depth H is 20 m while the initial water surface elevation η_0 is 1 m. T is the tidal-wave maker period, and $\omega = \frac{2\pi}{T}$.
Defining $k = \frac{\omega}{\sqrt{gH}}$, the exact solution is given by:

$$\begin{aligned} \eta(x, y, t) &= \eta_0 \cos(\omega t)\, \frac{\cos(k(L-x))}{\cos(kL)} \\ u(x, y, t) &= -\eta_0 \sin(\omega t)\, \frac{\sin(k(L-x))}{\cos(kL)} \\ v(x, y, t) &= 0 \end{aligned}$$

The boundary conditions are given by:

$$\eta(x, y, t) = \eta_0 \cos(\omega t), \qquad u(x, y, t) = 0, \qquad v(x, y, t) = 0, \qquad (x, y) \in \partial\Omega$$

- (Spatial Discretization): A seventh-order polyharmonic spline (PHS) augmented with a 6th degree polynomial was employed for the RBF–FD spatial discretization, with the total number of equispaced nodes as $N = 1600$. The solutions were approximated at $T = 2000$ seconds.

- (Time Discretization): We split the time domain into $M = 100$ sub-intervals. The fine and coarse propagators are constructed using Backward Euler integrators with differing number of time steps. We present results for $M_F = 100$ and $M_G = 5$.

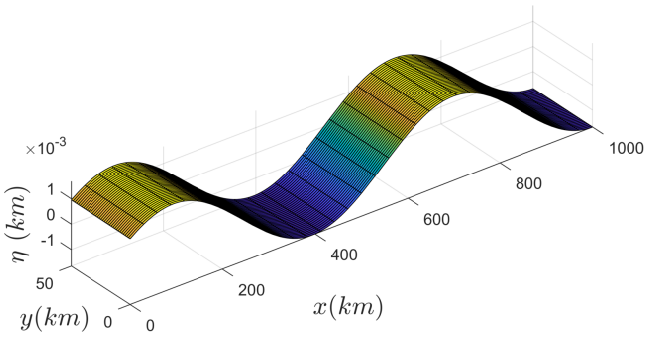

Figure 4: The analytical solution (η) of the eq. (7).

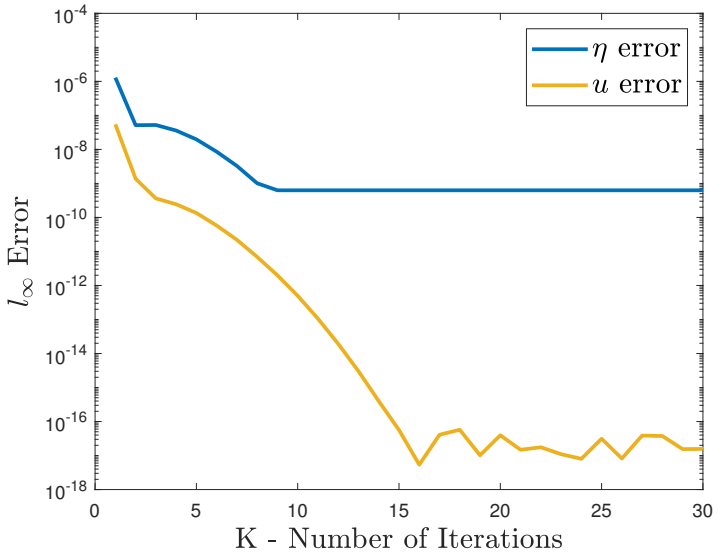

Figure 5: Parareal applied to a system of DEs, obtained by an RBF-FD discretization of the spatial operator in the 2-D shallow water equations. Both curves correspond to simulations with a coarse solver consisting of one internal time step, $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, 1)$. The blue curve represents error decay of η while the yellow curve represents the error decay of u. In the yellow curve, finite step convergence is observed. However, η seems not to have the desired finite step convergence. The accuracy of the η plot can be improved by increasing M_F. Further investigation on the reasoning behind this is ongoing work.

Figure 5 shows that the RBF-FD–Parareal algorithm converges in a finite number of iterations, significantly less than $M = 100$ for the dependent variable u. η does not converge to machine precision.

The Parareal algorithm is known to have stability issues when it is used to solve hyperbolic type PDEs [24]. If the PDE does not have diffusion, high wave numbers could cause high amplification, which leads to an unstable algorithm [25]. This is also exacerbated by the coarse solver. Several remedies are suggested in [26, 27, 28]. Further investigation of the convergence failure and adapting the methods stated in [26, 27, 28] is ongoing work. The error is computed by comparing the Parareal solution at $T = 2000$ seconds to a reference solution generated by applying the fine propagator sequentially across the M time domains. The blue curve corresponds to a convergence study of η while the yellow curve corresponds to u with a coarse solver consisting of five internal time steps, $\mathbf{G}(T_m, T_{m-1}, \mathbf{U}_{m-1}^{k-1}, 5)$. Since the fine solver consists of taking $M_F = 100$ internal time steps, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f} = \frac{1}{20}$. Convergence is achieved in $K \approx 8$ steps for η and $K \approx 13$ for u (machine

precision), so a theoretical speedup of approximately 1.8× (to 10 digits precision) for η and 1.2× (to machine precision) for u is possible, if 100 processors are used in the computation, see eq. (5).

4 Tighter coupling of the RBF–FD + Parareal framework

A key idea in the parareal algorithm is to use coarse and fine ODE solvers within a correction iteration. However, one can also apply spatial coarsening [29] within the fine and coarse propagators. In this section, we explore a tighter coupling between the RBF–FD + parareal framework using two strategies to construct a (cheaper) coarse differentiation matrix. One approach is to utilize a lower degree augmented polynomial to generate a coarse differentiation matrix with more sparsity due to the reduced number of neighbors. The second approach is to use a smaller number of spatial nodes to construct a coarse differentiation matrix. For instance, one could use 100 nodes to build a coarser differentiation matrix and 10000 nodes for the finer differentiation matrix. We also explore a combination of these two approaches. Overall, these modifications could further speed up the computations because these modifications yield sparser or smaller differentiation matrices. In addition, Section 5 shows that coarse differentiation matrices tend to have a better spectrum than differentiation matrices with a larger number of nodes and higher degree polynomial. However, there are two important criteria to be met with these modifications. We need to ensure that we can obtain finite step convergence, and that the spatial order of convergence remains the same. That is, the order of convergence is related to the degree of the augmented polynomial used in the finer differentiation matrix. In the following subsections, we will analyze each proposed modification.

4.1 Lower degree polynomials

The degree of the augmented polynomial constrains the order of convergence in space in the RBF–FD method using the polyharmonic spline augmented with polynomials. The RBF–FD method yields an order of convergence $\mathcal{O}(h^{\rho-k+1})$ [1] when approximating the kth derivative using a ρth degree polynomial. In addition, the differentiation matrices from the RBF-FD are $\mathcal{O}(nN)$ sparse where n is the number of neighbors while N is the total number of nodes. Typically, n is chosen to be as twice the number of terms in the augmented polynomial. Therefore, using a lower degree polynomial to construct the differentiation matrix in the coarse level in the parareal algorithm can save some computational cost if the sparsity structure of the differentiation matrix is exploited, but it is important to see if this has an impact on the overall order of convergence of the method. We implemented the above modifications to the 2D heat equation stated in eq. (6). For the finer differentiation matrix, we chose a 6th degree polynomial and the coarser differentiation matrix was created using polynomials of 3rd degree to 5th degree. Parareal parameters were chosen to be $M = 100$, $M_F = 20$,$M_G = 1$, and $K = 100$. It is also important to highlight that the same number of nodes were used to create both differentiation matrices for the tests in this subsection.

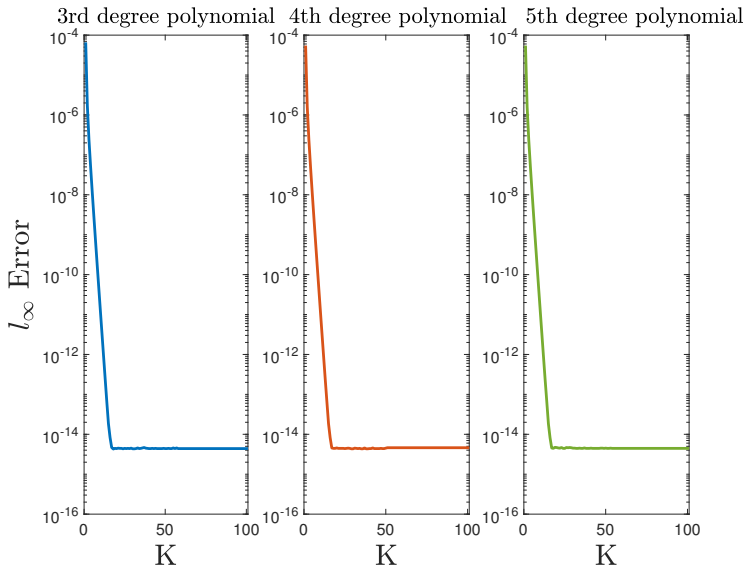

Figure 6: In all three graphs, the fine solver was created using a 6th degree polynomial. However, the polynomial degree for the coarse solve was varied. Finite step convergence was tested in three different cases where the coarse solver was varied as 3rd, 4th and 5th degree polynomials (left to right). All three figures display finite step convergence with $K \ll M$.

It is evident from Figure 6 that no matter the degree of the augmented polynomial used, finite step convergence can be observed. The effectiveness of the new framework is solidified from Figure 7 as we can see that the order of convergence in the finer differentiation matrix in space was still related to the augmented polynomial even for lower degrees ρ. It is also important to notice that one can run the algorithm with only a few iterations to obtain 7-8 digits of accuracy in the final solution. Not running all K correction iterations allows for an increase in computational efficiency.

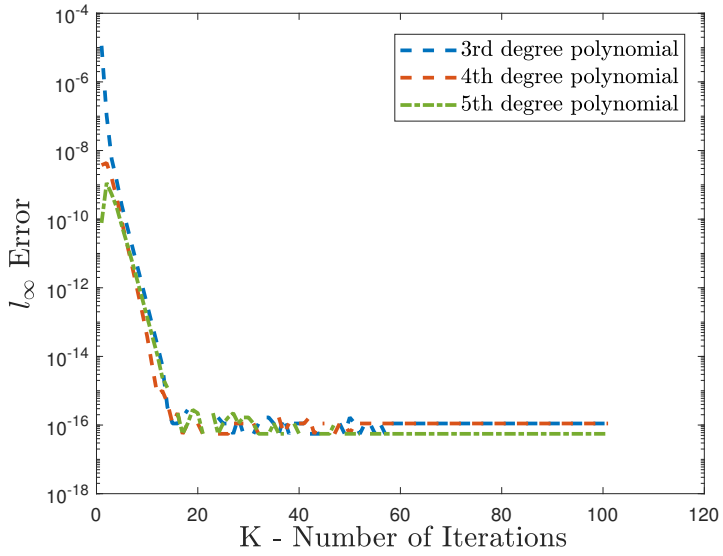

Figure 7: The heat equation was solved with 6th degree polynomial for both coarse and fine solver. Error was calculated by comparing these highly accurate solution with solutions using 3rd, 4th and 5th degree polynomials for a coarse solver while keeping the finer solver a 6th degree polynomial. This further solidifies that when one uses a lower degree polynomial the order of convergence still will remain related to the degree of the polynomial used with the fine solver.

4.2 Smaller number of nodes for the coarse solver

Another approach to construct a coarse differentiation matrix is to use a smaller number of nodes. For instance, for the tests implemented in this subsection, we used 10000 nodes to create a differentiation matrix for the finer level while at minimum we used 100 nodes to create a coarse differentiation matrix. In order to test finite step convergence, we varied the number of nodes used for the coarse differentiation matrix between 100 nodes to 5000 nodes. However, since the node distribution changes within the time integrator, an interpolation is necessary to switch between the data in the problem. Since the domain is fixed in all our test cases, we can pre-compute the interpolation matrices for efficiency. We used two approaches with interpolation. First a global RBF approach for interpolation with Gaussian RBFs and then an RBF-FD local approach. Polyharmonic Splines RBFs were used to construct differentiation weights with a 6th degree polynomial augmentation for both coarser and finer differentiation matrices. Parareal parameters were chosen to be $M = 100$, $M_F = 20$, $M_G = 1$, and $K = 100$.

Figure 8 shows finite step convergence, but at the cost of several digits of accuracy and a slightly higher number of iterations. We believe this is a result of the interpolation error and the choice of the shape parameter. When the shape parameter goes to zero the accuracy is expected to increase but at the cost of increasing the condition number of the interpolation matrix [30]. Hence, one needs to find an optimal shape parameter to get the best results. Therefore, in order to avoid choosing a suitable shape parameter, we used PHS RBFs and a local interpolation approach. Figure 9 shows that with the local interpolation, there is no loss of accuracy or increase in the number of iterations needed to observe finite step convergence. Therefore, the tests in this section show that one could use coarse differentiation matrices in the coarse level in Parareal algorithm and still observe finite step convergence, and obtain a computational speed up. In addition, the order of convergence in space remains related to the degree of the augmented polynomial used in the finer level despite using a node distribution with a smaller number of nodes or a lower degree polynomial at the coarse level.

Consider analyzing the theoretical speed-up that can be obtained using a lower degree polynomial and a smaller node set in the coarse solver. For the test cases in this section, the fine solver consists of taking $M_F = 20$ internal time steps, the ratio of the computational effort between $\mathbf{G}$ and $\mathbf{F}$ is $\frac{N_c}{N_f}\frac{\gamma_c}{\gamma_f} = \frac{1}{20} \times 0.0026$. Inverting the coarse differentiation matrix using MATLAB's backslash operator is 100 times faster than inverting the fine differentiation matrix to solve a single step in Backward Euler's method. Hence, we obtain the ratio as $\frac{\gamma_c}{\gamma_f} = 0.0026$. This is calculated by testing the time that it takes to solve a single step of Backward Euler with coarse and fine solver. The coarse differentiation matrix is constructed out of a 3rd degree polynomial with 100 nodes while the fine differentiation matrix was constructed with a 6th degree polynomial, with 10000 nodes. Convergence is achieved in $K \approx 18$ steps, so a theoretical speedup of approximately $5\times$ (to machine precision) is possible, if 100 processors are used in the computation, see eq. (5). This speed up is significant and useful when solving large system of equations, i.e., RBF discretizations with a large number of nodes.

5 Behavior of the eigenvalues of RBF–FD modifications to parareal

One drawback of using the RBF-FD spatial discretization in time-dependent PDEs is that the differentiation matrices can contain undesirable spurious eigenvalues, i.e., some spurious eigenvalues with positive real components requiring very small time steps to reside within the ODE solver's stability region, increasing the computational cost [1, 7, 8, 9].However, reducing the time step

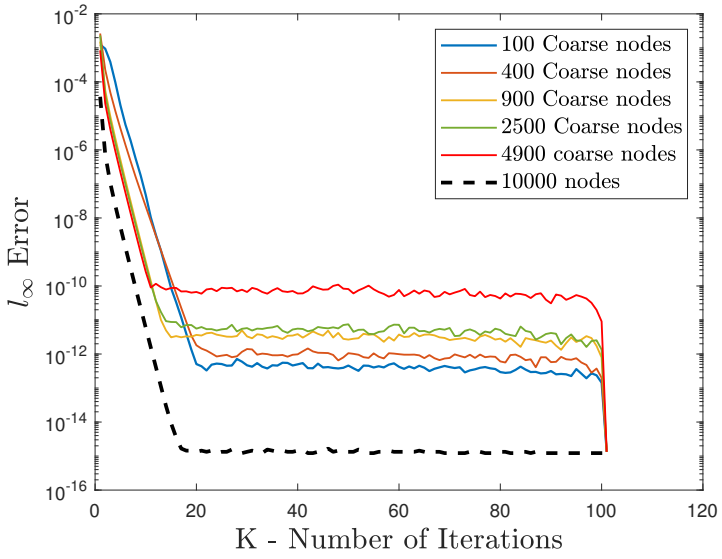

Figure 8: Finite-step convergence display when a small node distribution is used for the coarse solver. The black curve shows finite step convergence when 10000 nodes are used for both the coarse and fine solver. Other curves show finite step convergence when the number of nodes used for the coarse solver is less than 10000 while keeping the number of nodes of the fine solver fixed at 10000. There is an accuracy loss due to the choice of shape parameter in the global RBF interpolation.

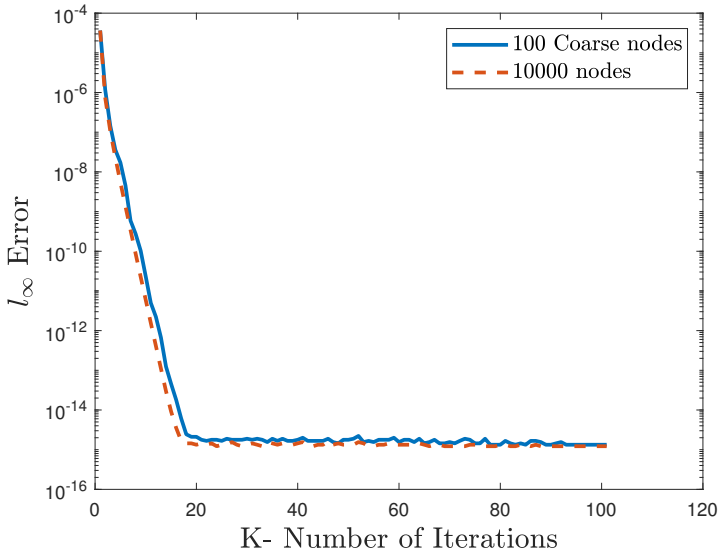

Figure 9: Finite-step convergence display when a small node distribution is used for the coarse solver but with RBF–FD for interpolation. Comparing to Figure 8 there is no accuracy lost as we used shape parameter independent local interpolation approach.

is not enough in many instances to address this issue. It is common to add a very small artificial hyperviscosity term to shift the eigenvalues to the left half of the complex plane. It has been observed that the differentiation matrix tends to have more spurious eigenvalues as we increase the number of nodes or increase the degree of the appended polynomial. Since the coarse differentiation matrices from parareal are constructed with a smaller number of nodes or a lower degree polynomial, the coarse differentiation matrix tends to have a better spectrum than the fine differentiation matrix. Therefore, our proposed framework of choosing a coarse differentiation matrix allows for larger time steps in the coarse level. Although the fine solver still requires smaller time steps due to the undesirable spurious eigenvalues, the computation in the finer level can be done in parallel. In this section, we provide a stability analysis of the RBF–FD and parareal algorithm using the existing theoretical results from the literature [20]. In addition, we use the same theoretical results to show why using a coarse differentiation matrix at the coarse level in parareal is better than using a finer differentiation matrix in both levels. We will first discuss the results for the heat equation and later for the transport equation.

5.1 Heat equation

Consider the one dimensional heat equation:

$$\begin{aligned} \frac{\partial u}{\partial t} &= u_{xx} && x \in \Omega, && t \in [0,T] \\ u(x,t) &= g(x,t), && x \in \partial\Omega, && t \in [0,T] \\ u(x,0) &= h(x), && x \in \Omega \end{aligned} \tag{8}$$

Theorem 5.1. *[20] Let $F(T_{m+1}, T_m, U_m^k)$ be the exact solution at T_{m+1} of eq. (8) with $u(T_m) = U_m^k$, and let $G(T_{m+1}, T_m, U_m^k)$ be a coarse solver with stability function R, such that $\sup_{x<0} |e^x - R(x)| = \rho_s$. Then the parareal algorithm applied to the heat equation satisfies the superlinear convergence bound*

$$\max_{1\le m\le\le M} \left\| u(t_m) - U_m^k \right\| \le \frac{\rho_s^k}{k!} \max_{1\le m\le M} \left\| u(T_m) - U_m^0 \right\|_2 \tag{9}$$

where $\|\cdot\|_2$ denotes the spectral norm in space and the constant ρ_s is universal, $\rho_s = 0.2036321888$. On unbounded time intervals, we have

$$\sup_{n>0} \left\| u(t_m) - U_m^k \right\| \le \rho_l^k \sup_{m>0} \left\|_u(T_m) - U_m^0 \right\|_2 \tag{10}$$

where the universal constant $\rho_l = 0.2984256075$.

We can rewrite eq. (8) as an ODE using Fourier transformation in space:

$$\widehat{u_t} = -\omega^2 \widehat{u}$$

We will call the constants ρ_s and ρ_l in eqs. (9) and (10) as superlinear and linear contraction factors respectively from now on. The derivation of these universal contraction factors is independent of the spatial discretization, i.e., they are purely based on the ODE theory. For further details on the proof, refer to [20]. Let R refer to the stability function of an ODE solver which provides its stability region in the complex plane. For the Backward Euler method, the upper bounds for contraction factors in eqs. (9) and (10) are

$$\rho_s = \sup_{z\in\mathbb{R}^-} \left| e^z - \frac{1}{1-z} \right|, \text{ and } \rho_l = \sup_{z\in\mathbb{R}^-} \frac{\left| e^z - \frac{1}{1-z} \right|}{1 - \left| \frac{1}{1-z} \right|},$$

where $z = -\omega^2 \Delta T$.

If we consider the system of ODEs $u_t = Du$ derived using the method of lines approach, we can show that the contraction factors are given by the complex values $z = \lambda \Delta T$, where $\lambda \in \mathbb{C}$ are the eigenvalues of the differentiation matrix formed from the discretization stencil. If we compare the RBF–FD differentiation matrix to the five-point stencil finite-difference method, the eigenvalues of the latter will be purely real and negative, and the universal constants stated in the theorem can be verified. However, the differentiation matrices from RBF–FD will have eigenvalues scattered over the entire complex plane, and therefore, an important question to investigate is whether the contraction factors from the above theorem are still valid for RBF–FD discretizations. In order to answer this problem, we take a closer look at the two complex functions used to find contraction factors:

$$f(z) = \left| e^z - \frac{1}{1-z} \right|, \qquad g(z) = \frac{\left| e^z - \frac{1}{1-z} \right|}{1 - \frac{1}{1-z}}.$$

Figure 10 shows the contours of $f(z)$ on the left half of the complex plane, and the spectrum of the RBF–FD discretized Laplace operator made out of equispaced 100 nodes augmented with a third-degree polynomial. The black mark indicates the maximum of $f(z)$ evaluated at the eigenvalues (hence eigenvalue that produce the contraction factor as maximum of $f(z)$ is the contraction factor). The first key observation is that the spectrum of the differentiation matrix does not have spurious eigenvalues, and it is densely arranged close to the real axis of the left half of the complex plane. This is because the eigenvalues are from a

Table 2: The maximum value of $g(z)$ found amongst the eigenvalues of the differentiation matrices discretizing the Laplacian operator formed from various combinations of total nodes and degree of polynomial. The nodes were arranged in a hexagonal grid to produce the best spectrum. When the total number of nodes and/or the polynomial degree is increased, the table shows that the contraction value gets larger than the universal value. This can be observed by looking at the right part upper and lower corner of the table. In addition, the contraction factor is less than one for all choices.

Polynomial degree	Total number of nodes				
	100	196	900	1600	3600
3	2.97E-01	2.98E-01	2.98E-01	2.98E-01	5.67E-01
4	2.98E-01	2.98E-01	4.65E-01	2.98E-01	2.98E-01
5	2.98E-01	2.98E-01	3.00E-01	1.46E+00	2.98E-01
6	3.07E-01	3.72E-01	5.79E-01	6.87E-01	3.00E-01

coarser differentiation matrix and a lower degree polynomial. However, if the differentiation matrix has spurious eigenvalues, they could fall into the regions where the iso-curves of $f(z)$ (or $g(z)$) would have larger values. As another test, we increased the number of nodes and the degree of the polynomial in coarser differentiation matrices and evaluated $g(z)$ at the eigenvalues of those respective matrices and recorded the maximum value of $g(z)$ found at amongst the eigenvalue evaluations in Table 2. It clearly shows that the linear contraction factor increases as we increase the number of nodes or the degree of the polynomial. For a smaller number of nodes or a lower degree of polynomial the contraction factor is closer to the universal value given in the theorem. It is also important to notice that in Theorem 5.1, the contraction values are related to the coarse solver and hence it is important to ensure that we have a differentiation matrix with the best spectrum at the coarse level. Therefore, to ensure the stability and efficiency of the algorithm, it is better to use a coarse differentiation matrix.

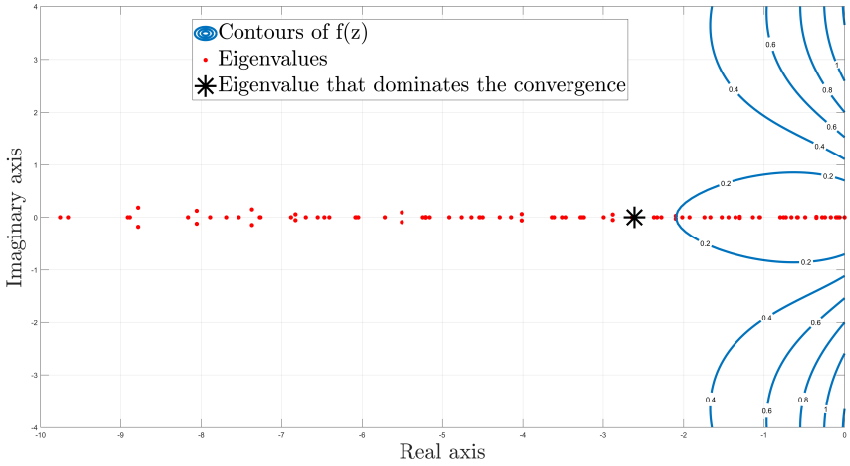

Figure 10: Spectrum of the RBF-FD discretized Laplacian operator using a seventh order PHS augmented with a 3rd degree polynomial overlayed with the contours of $f(z)$.

5.2 Transport equation

The issue of spurious eigenvalues can be severe when solving purely advective equations. A common practice among RBF researchers is to introduce artificial viscosity (hyperviscosity) to the model to ensure the stability of the algorithm [31]. In the upcoming analysis, we show that using a coarse differentiation matrix at the coarse level of the parareal algorithm is an excellent option compared to how large the contraction values can get when a finer differentiation matrix is used. For the numerical study, we again use a theorem from the parareal literature [20]. It is also important to notice that in Theorem 5.2, again the stability function R of the coarse solver is the key factor when it comes to choosing a contraction factor.

Theorem 5.2. *[20] Let $F(T_{m+1}, T_m, U_m^k)$ be the exact solution at T_{m+1} of eq. (8) with $u(T_m) = U_m^k$, and let $G(T_{m+1}, T_m, U_m^k)$ be a coarse solver with stability function R, such that $\sup_{x\in\mathbb{R}} \left|e^{ix} - R(ix)\right| = \rho_s$. Then the parareal algorithm applied to the transport equation satisfies the superlinear convergence bound*

$$\max_{1\leqslant m\leq\leqslant M} \left\|u(t_m) - U_m^k\right\| \leq \frac{\rho_s^k}{k!} \max_{1\leqslant m\leq\leqslant M} \left\|u(T_m) - U_m^0\right\|_2$$

where $\|.\|_2$ denotes the spectral norm in space and the constant ρ_s is universal, $\rho_s = 1.224353426$.

Figures 11 and 12 shows the spectrum of the differentiation matrices constructed with a 3rd degree polynomial and a 6th degree polynomial augmentation respectively. Figure 11 shows that the largest value of $f(z)$ is around 0.2 for the spectrum of a differentiation matrix made with a lower degree polynomial. This is well below the contraction factor stated in Theorem 5.2. It is also important to notice that when we compare Figures 11 and 12, we observe that the finer differentiation matrix has spurious eigenvalues. This is further validated when we look at Table 3, we see that the contraction factor gets larger when we increase the number of nodes. One could argue that for RBF–FD problems, 3600 nodes is a relatively small number of nodes. Therefore, one could use a coarse differentiation matrix at the coarse level that ensures the stability of the algorithm and use a finer differentiation matrix and a small-time step at the finer level since the computations at the finer level are done in parallel. In addition, despite $\rho_s > 1$ the parareal still has super-linear convergence $\left(\frac{1}{k!}\right)$, which allows it to solve advection equation with $K \ll M$.

Table 3: The maximum value of $f(z)$ found amongst the eigenvalues of the differentiation matrices discretizing the transport equation formed from various combinations of total nodes and degree of polynomial. The nodes were arranged in a Cartesian grid to produce the best spectrum. When the total number of nodes and/or the polynomial degree is kept low, the table shows that the contraction value is smaller and better than the universal value. This can be observed by looking at the left part upper and lower corner of the table. In addition, the method converges for all choices of node distributions and the polynomial degree.

Polynomial degree	Total number of nodes				
	100	196	900	1600	3600
3	1.80E-01	3.28E-01	8.65E-01	1.09E+00	1.27E+00
4	2.00E-01	3.81E-01	8.65E-01	1.17E+00	1.23E+00
5	1.90E-01	3.65E-01	9.48E-01	1.15E+00	1.27E+00
6	1.83E-01	3.45E-01	9.29E-01	1.14E+00	1.51E+00

6 Conclusion

In this manuscript, we explore RBF-FD + parareal as an approach to solve time dependent PDEs. There are several theoretical and practical implications of our study. The first is the ability to reduce computational cost in solving a time-dependent PDE by reducing the number of computational nodes for the differentiation matrix in the coarse solver and by computing segments of the fine solution in parallel. This reduction in computational cost is attractive, especially when solving PDEs with larger amount of discretization data. The second is a way to mitigate the effect of the spurious eigenvalues introduced by high order polynomials by using the parareal algorithm with coarse RBF-FD stencils and lower degree polynomial augmentation.We believe our proposed framework can be adopted to solve PDEs in three spatial dimensions, and this is part of our future work. Finally, our numerical study paves the way to theoretically analyze the behavior of spurious eigenvalues of RBF–FD differentiation matrices.

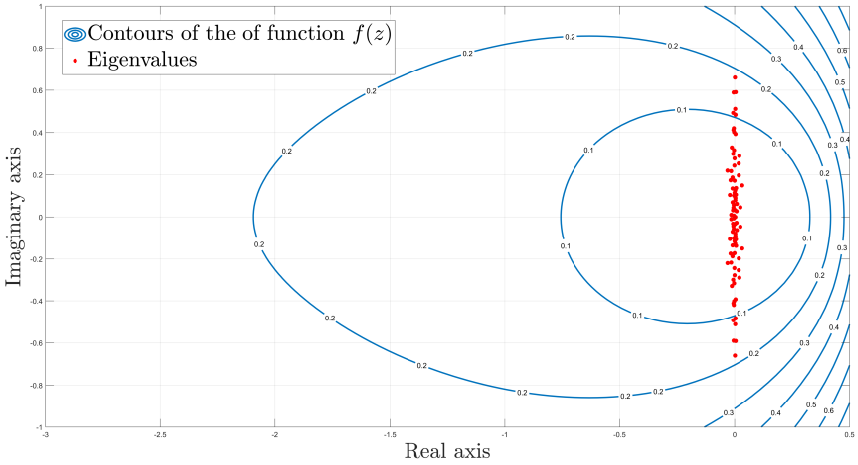

Figure 11: Spectrum of the RBF-FD discretized transport equation using a seventh order PHS augmented with a 3rd degree polynomial overlayed with the contours of $f(z)$.

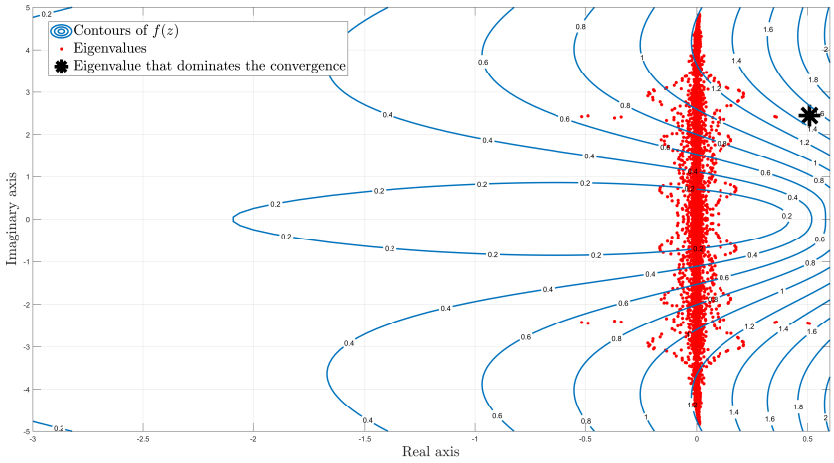

Figure 12: Spectrum of the RBF-FD discretized transport equation using a seventh order PHS augmented with a 6th degree polynomial overlayed with the contours of $f(z)$.

References

[1] B. Fornberg and N. Flyer. A Primer on Radial Basis Functions with Applications to the Geosciences. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015.

[2] R. Zamolo and L. Parussini. Analysis of geometric uncertainties in CFD problems solved by RBF-FD meshless method. Journal of Computational Physics, 2020.

[3] P. Cécile, N. Dissanayake, J. Gierke, and B. Fornberg. The Radial Basis Functions Method for Improved Numerical Approximations of Geological Processes in Heterogeneous Systems. Mathematical Geosciences, 52,08 2019.

[4] V. Bayona, N. Flyer, G. M. Lucas, and A. J. G. Baumgaertner. A 3-D RBF-FD solver for modeling the atmospheric global electric circuit with topography (GEC-RBFFD v1.0). Geoscientific Model Development, 8(10):3007-3020, 2015.

[5] N. Kulasekera Mudiyanselage, New Numerical Approximations of Geological Processes in Heterogeneous Systems Using radial Basis Functions. Open Access Dissertation, Michigan Technological University, 2021

[6] J. Močnik,and J. M. Berljavac, K. Mishra, J. Slak, and G. Kosec. RBF-FD analysis of 2D time-domain acoustic wave propagation in heterogeneous media. Computers & Geosciences, 153:104796, 2021.

[7] V. Shankar, and A. L. Fogelson. Hyperviscosity-Based Stabilization for Radial Basis Function-Finite Difference (RBF-FD) Discretizations of Advection-Diffusion Equations Journal of computational physics, 372:616-639, 2018.

[8] R. B. Platte and T. A. Driscoll. Eigenvalue Stability of Radial Basis Function Discretizations for Time-Dependent Problems. Computers & Mathematics with Application, 51:1251-1268, 2006

[9] V. Shankar, G. Wright, and A. Narayan. A Robust Hyperviscosity Formulation for Stable RBF-FD Discretizations of Advection-Diffusion-Reaction Equations on Manifolds. SIAM Journal on Scientific Computing, 42(4):A2371-A2401, 2020

[10] J. L. Lions, Y. Maday, and G. Turinici. Résolution d'edp par un schéma en temps pararéel. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics, 332(7):661-668, 2001.

[11] S. Razvan and I. M. Navon. POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model Journal of Computational Physics, 237:95-114, 2013

[12] C. Farhat and M. Chandesris. Time-decomposed parallel time-integrators: Theory and feasibility studies for fluid, structure, and fluid-structure applications Internat. J. Numer. Methods Engrg, 58:1397-1434, 2003

[13] J. M. F. Trindade and J. F. Pereira. Parallel-in-time simulation of the unsteady Navier-Stokes equations for incompressible flow International Journal for Numerical Methods in Fluids, 45(10):1123-1136, 2004

[14] M. Duarte, M. Massot, and S. Descombes. Parareal operator splitting techniques for multi-scale reaction waves: Numerical analysis and strategies. ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique, 45(5):825-852, 2011

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

[16] V. Bayona, An insight into RBF-FD approximations augmented with polynomials Computers and Methematics with Applications, 77:2337-2353, 2019

[17] V. Bayona, N. Flyer, B. Fornberg, G. Barnett. On the role of polynomials in RBF-FD approximations: II. Numerical solution of elliptic PDEs Journal of Computational Physics, 322:257-273, 2017

[18] V. Bayona, N. Flyer, B. Fornberg. On the role of polynomials in RBF-FD approximations: III. Behavior near domain boundaries Journal of Computational Physics, 380:378-399, 2019

[19] M. J. Gander. Analysis of the parareal algorithm applied to hyperbolic problems using characteristics Boletín de la Sociedad Española de Matemática Aplicada, 42:21-35, 2008

[20] M. J. Gander, and S. Vandewalle. Analysis of the Parareal Time-Parallel Time-Integration Method. SIAM Journal on Scientific Computing, 29(2):556-578, 2007.

[21] M. L. Minion. A hybrid parareal spectral deferred corrections method Communications in Applied Mathematics and Computer Science, 5(2):265-301, 2010

[22] A. Arteaga, D. Ruprecht and R. Krause. A stencil-based implementation of Parareal in the C++ domain specific embedded language STELLA Applied Mathematics and Computation, 267:727-741, 2015

[23] C. K. Chou, C. P. Sun, D. L. Young, J. Sladek, and V. Sladek. Extrapolated local radial basis function collocation method for shallow water problems. Engineering Analysis with Boundary Elements, 50:275-290, 2015.

[24] G.A. Staff, E.M. Rønquist. Stability of the Parareal Algorithm. Domain Decomposition Methods in Science and Engineering. Lecture Notes in Computational Science and Engineering. Springer, Berlin, Heidelberg, vol 40, 2005

[25] D. Ruprecht, Wave propagation characteristics of Parareal. Computing and Visualization in Science, 19:1-17, 2018

[26] He. Liping, J.S. Hesthaven, X. Zhu The reduced basis technique as a coarse solver for parareal in time simulations. J. Comput. Math, 28:676-692, 2010.

[27] D. Ruprecht, R. Krause Explicit parallel-in-time integration of a linear acoustic-advection system. Computers & Fluids, 59:72-83, 2012

[28] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. International journal for numerical methods in engineering, 67(5):697-724, 2006

[29] D. Ruprecht, Convergence of Parareal with spatial coarsening. Proceedings in Applied Mathematics and Mechanics, 14:1031-1034, 2014

[30] G. E. Fasshauer. Meshfree Approximation Methods with Matlab. World Scientific, 2007

[31] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. Computers & Mathematics with Applications, 48(5):853-867, 2004.