

Article

A Sequential, Implicit, Wavelet-Based Solver for Multi-Scale Time-Dependent Partial Differential Equations

Donald A. McLaren ¹, Lucy J. Campbell ² and Rémi Vaillancourt ^{1,*}

¹ Department of Mathematics and Statistics, University of Ottawa, 585 King Edward Ave., Ottawa, Ontario K1N 6N, Canada; E-Mail: dmcla047@uottawa.ca

² School of Mathematics and Statistics, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario K1S 5B6, Canada; E-Mail: campbell@math.carleton.ca

* Author to whom correspondence should be addressed; E-Mail: remi@uottawa.ca;
Tel.: +16-135-625-800 (ext. 3533); Fax: +16-135-625-776.

Received: 28 February 2013; in revised form: 13 March 2013 / Accepted: 8 April 2013 /

Published: 23 April 2013

Abstract: This paper describes and tests a wavelet-based implicit numerical method for solving partial differential equations. Intended for problems with localized small-scale interactions, the method exploits the form of the wavelet decomposition to divide the implicit system created by the time-discretization into multiple smaller systems that can be solved sequentially. Included is a test on a basic non-linear problem, with both the results of the test, and the time required to calculate them, compared with control results based on a single system with fine resolution. The method is then tested on a non-trivial problem, its computational time and accuracy checked against control results. In both tests, it was found that the method requires less computational expense than the control. Furthermore, the method showed convergence towards the fine resolution control results.

Keywords: wavelet; multiscale; partial differential equation; Rossby wave problem

Classification: MSC 65T60; 65M20

1. Introduction

Wavelets are a versatile tool for representing and analyzing functions. Wavelet transforms can be continuous, though we will focus on the discrete form (for both, see [1,2], or any other textbook on the subject). The discrete transform decomposes a function via families of wavelets and wavelet-related basis

functions. Wavelets can be orthogonal [3–5], and biorthogonal [6]. The greatest advantage of wavelet decompositions is that they combine multiple scales (much like different Fourier modes, or frequencies) with substantial localization. As a result, wavelets can be used directly to efficiently approximate the solution of partial differential equations, as in [7,8] (where wavelet decompositions are used in every spatial direction, as well as for the time discretization). The decomposition can also be used indirectly, as a tool to analyze a function and determine where greater resolution is necessary, like in [9,10]. Wavelets have been used to efficiently represent operators, as in [11]. Work has also been done adapting wavelets to the challenges inherent in solving partial differential equations, notably the introduction of second generation wavelets in [12]. Unlike first generation wavelets, these are not always equal to translates and scalings of a “mother” function, and so can be (more easily) made compatible with, for instance, the domain of a partial differential equation. Also, since the multi-resolution decomposition is usually the key property, multi-resolution representations have been developed that do not require a wavelet basis, see [13]. We will keep our model simple, and so use biorthogonal wavelets (symmetric, with finite support), with a few small modifications to satisfy any required boundary conditions.

The goal of this paper is to improve the efficiency of implicit methods for time-dependent partial differential equations. The basic premise is that the time step resulting from an implicit discretization can be calculated at multiple resolutions, with each resolution calculated separately (sequentially). If the problem requires fine resolution in small sub-domains, then this approach can significantly speed up implicit calculation of time steps. At its most basic, this method starts by calculating what we call the large-scale system: The time step over the entire domain Ω , at a coarse resolution. Next, some of those calculated wavelet coefficients are used in the small-scale system, a new set of calculations at a fine resolution in the sub-domain $\Lambda \subset \Omega$. Some of the coefficients from the small-scale system are then added to the large-scale system and the time step is complete.

The advantage to our approach is the ability to compute a time step at different resolutions, in different domains, sequentially. Solving a problem at multiple resolutions is an obvious way to exploit a multi-scale decomposition. As such, there are similar methods to be found throughout the literature. The most similar are those intended to be used to solve elliptic problems at multiple scales. One that is particularly connected to our own method, due to its use of biorthogonal wavelets, can be found in [14]. Others can be found in [15–17]. Our method is different from those in that it is intended to be used on time-dependent problems. More importantly, adaptive multi-scale solvers for elliptic problems use repeated iterations at each resolution to maintain consistency and reduce the error. Our method keeps the different levels of resolution, the different scales, consistent with each other. This means that no repeated iteration is needed for each time step, so the problem only needs to be calculated once for each scale. This is not an advantage in an elliptic solver, but when several thousand (if not million) time steps have to be calculated, it is advantageous to avoid having to repeatedly calculate individual time steps.

Another perspective on our method is that it allows, in effect, the problem to be solved in a different manner in the subdomain Λ than over the remainder of the full domain Ω . This difference is, at the moment, restricted to the use of a finer resolution, but it could well be that using a different time-discretization altogether could be an option (note that this is not yet tested). This means that our method has a property similar to that of domain decomposition methods. As with adaptive multi-scale solvers, domain decomposition methods typically require multiple iterations between the different

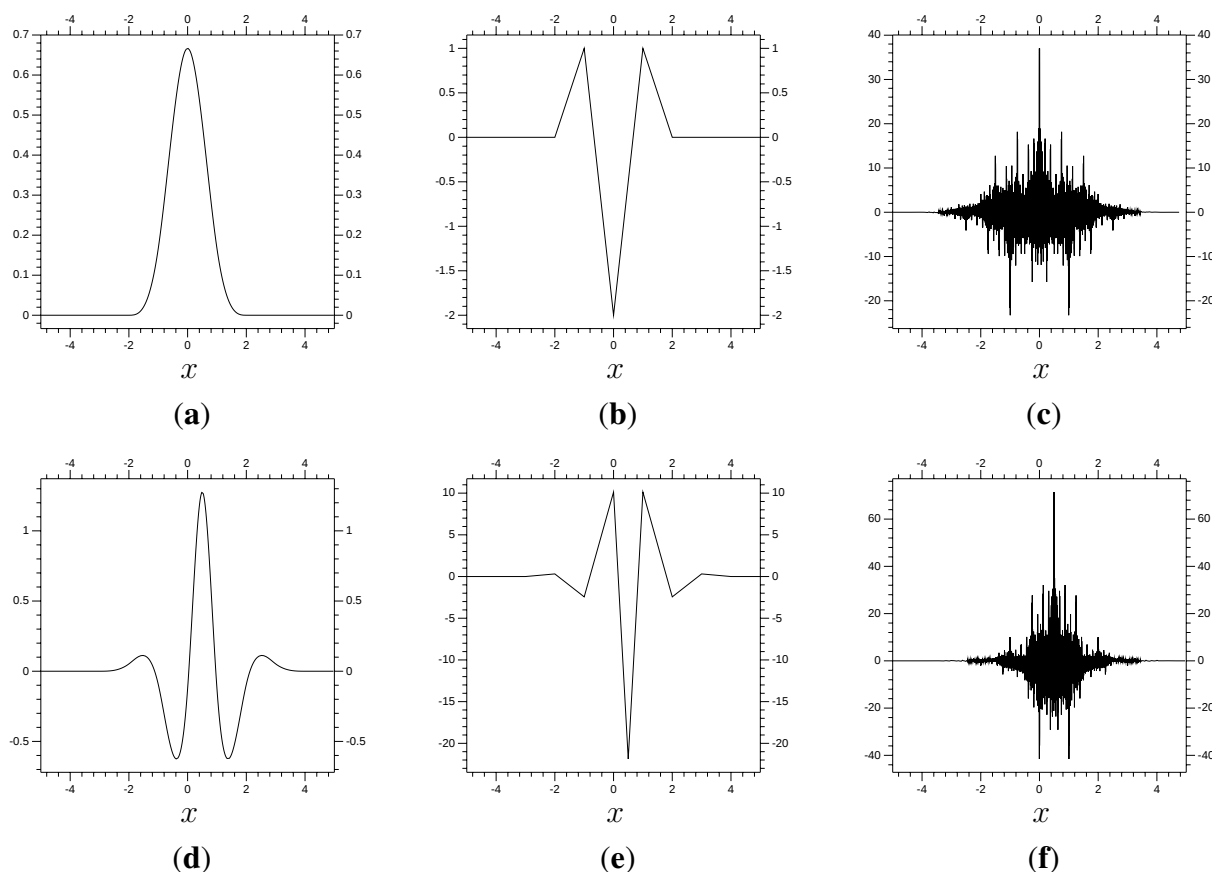
domains. Furthermore, they typically require limited interaction across the interfaces between domains. Our method is not restricted by the extent of interaction across the interfaces, only by whether those interactions are adequately modeled by the large-scale (coarse) resolution.

2. Wavelet Decomposition

Our notation is largely based on [2], a good introductory source for wavelets, which bases its approach on [1]. Further discussion on the many applications of wavelets can be found in [18]. We use the terminology, and some formatting, based on the theory of first generation wavelets, but the key steps for our method require only a multi-resolution decomposition. The concept could be extended to second generation wavelets, or to alternative decompositions. For instance, the discrete data based multi-resolution decomposition introduced by Harten in [13] would be ideal, in many ways.

Our numerical examples each use a biorthogonal spline wavelet/scaling function pair (biorthogonal wavelets were introduced in [6]). The first example uses a scaling function ϕ equivalent to the fourth order spline function, which is shown in Figure 1, along with its second derivative ϕ'' and its dual $\tilde{\phi}$. The figure also shows the corresponding wavelet ψ , along with its second derivative ψ'' and its dual $\tilde{\psi}$.

Figure 1. Top: The wavelet pair corresponding to the spline function B_3 , its second derivative and dual re-centered at $x = 0$. Bottom: The same for the corresponding wavelet. (a) $\phi(x)$; (b) $\frac{d^2}{dx^2} \phi(x)$; (c) $\tilde{\phi}(x)$; (d) $\psi(x)$; (e) $\frac{d^2}{dx^2} \psi(x)$; (f) $\tilde{\psi}(x)$.



A wavelet decomposition uses variants of the form

$$\phi_{j,k}(x) = 2^{-j/2} \phi(2^{-j}x - k), \quad \psi_{j,k}(x) = 2^{-j/2} \psi(2^{-j}x - k), \quad j, k \in \mathbb{Z} \quad (2.1)$$

Notice that $\|\phi_{j,k}\| = \|\phi\|$ and $\|\psi_{j,k}\| = \|\psi\|$ for all $j, k \in \mathbb{Z}$ (using the L^2 norm). Also, notice that the functions become more localized (*i.e.*, have their support reduced) as j becomes more negative, while k determines the location of $\phi_{j,k}$ or $\psi_{j,k}$. Notice also that $\phi_{0,k}$ and $\phi_{0,k+1}$ are equal except for a translation of one, and that $\phi_{-1,k}$ and $\phi_{-1,k+1}$ are equal except for a translation of $\frac{1}{2}$. As a result, more negative values of j do not just make the functions $\phi_{j,k}$ and $\psi_{j,k}$ more localized, but also reduce the space between them. One key property of the scaling function, ϕ , and wavelet, ψ , is that they can be written in terms of the finer scaling functions $\phi_{-1,k}$, so

$$\phi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} h_k \phi(2x - k), \quad \psi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} g_k \phi(2x - k) \quad (2.2)$$

for constants g_k and h_k .

These functions create subspaces of $L^2(\mathbb{R})$ of the form

$$V_j = \text{Span} \{ \phi_{j,k}, k \in \mathbb{Z} \}, \quad W_j = \text{Span} \{ \psi_{j,k}, k \in \mathbb{Z} \}, \quad \text{for all } j \in \mathbb{Z}, \text{ with } V_{-1} = V_0 \oplus W_0 \quad (2.3)$$

This definition and (2.2) mean that $V_0 \subseteq V_{-1}$ and, more generally, $V_j \subseteq V_{j-1}$ for all $j \in \mathbb{Z}$. We also get $V_j = V_{j+1} \oplus W_{j+1}$, which in turn makes

$$V_J = V_0 + W_0 + W_{-1} + \cdots + W_{J+1} \quad (J < 0) \quad (2.4)$$

As a result, any function $f \in V_J$ ($J < 0$) can be expressed using different, but equivalent, sets of basis functions from the spaces V_j and W_j . First, by the definition of V_J ,

$$f(x) = \sum_{k \in \mathbb{Z}} a_{J,k} \phi_{J,k} \quad (2.5)$$

However, assuming that $J < 0$, we can write it in terms of $V_{J+1} \oplus W_{J+1} = V_J$, so

$$f(x) = \sum_{k \in \mathbb{Z}} a_{J+1,k} \phi_{J+1,k} + \sum_{k \in \mathbb{Z}} b_{J+1,k} \psi_{J+1,k} \quad (2.6)$$

Continuing this pattern, we can get

$$f(x) = \sum_{k \in \mathbb{Z}} a_{J+2,k} \phi_{J+2,k} + \sum_{k \in \mathbb{Z}} b_{J+1,k} \psi_{J+1,k} + \sum_{k \in \mathbb{Z}} b_{J+2,k} \psi_{J+2,k} \quad (2.7)$$

which can be expanded to

$$f(x) = \sum_{k \in \mathbb{Z}} a_{0,k} \phi_{0,k} + \sum_{j=J+1}^0 \sum_{k \in \mathbb{Z}} b_{j,k} \psi_{j,k} \quad (2.8)$$

and further. Basically, there is a starting function of the form

$$\sum_{k \in \mathbb{Z}} a_{0,k} \phi_{0,k} \in V_0 \quad (2.9)$$

which is simply an approximation of f in the coarse resolution space V_0 . Then functions

$$\sum_{k \in \mathbb{Z}} b_{j,k} \psi_{j,k} \in W_j \quad (2.10)$$

are added to correct that approximation. So, any function $f \in V_J$ has a unique component in V_0 (equal to the projection of f onto V_0 when orthogonal wavelets are used), an approximation containing only the coarser, large-scale, features of f . The finer, smaller scale, features of f are found in the spaces W_0 to W_{J+1} .

3. Multi-Scale Method

We propose a new wavelet method for solving time-dependent partial differential equations, specifically multi-scale problems that require implicit time-discretization. The method exploits the form of the wavelet decomposition to divide the implicit system created by the time-discretization into multiple, smaller, systems that can be solved sequentially. One can, if certain requirements are taken into account, compute implicit time steps at multiple spatial resolutions. Most importantly, these resolutions are solved sequentially, and kept consistent so they need only be solved once per time step. For non-linear problems, where even a linearized scheme (such as that in Equation (7.2)) requires solving a new system every time step, this can greatly reduce computational expense. The key idea is to divide the system to be solved into two, smaller, systems that can be solved separately. First, we have a coarse resolution

$$V_j = V_0 \oplus W_0 \oplus \cdots \oplus W_{j+1} \quad (3.1)$$

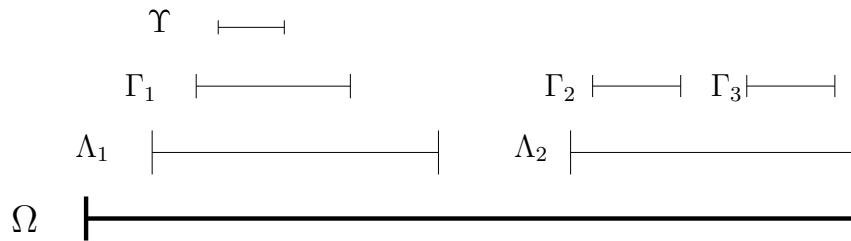
over the whole domain Ω , called the large-scale system. The next system has the finer resolution

$$V_{j+m} = V_j \oplus W_j \oplus \cdots \oplus W_{j+m+1} \quad (3.2)$$

only covering the subset $\Lambda \subset \Omega$, called the small-scale system (with $j, m < 0$). This arrangement allows the two systems to be solved sequentially. First, the large-scale system is solved, in V_j over Ω , and the resulting coefficients that are within (or near) Λ are converted so that they can be expressed in terms of the small-scale. Those coefficients are included in the calculations for the new small-scale system. Next, the large-scale (V_j) components of the newly calculated small-scale system are converted to the form used in the large-scale, and added to the earlier large-scale values. This finishes the time step. Note that the two systems, large and small-scale, are solved separately.

Depending on how the discretization is implemented, the two systems can be the same size (that is, have the same number of coefficients). In fact, this arrangement can be done with any number of systems, possibly nested, as shown in Figure 2. As long as their domains are either nested or disjoint, any number of systems can be used. That the different systems can be solved separately is the advantage here. A pair of systems sized 80×80 will be quicker to solve than a single system sized 160×160 .

Figure 2. A diagram of several nested small-scale systems of a domain Ω .



4. Implementation

In this section we describe the basic implementation of the method: The dividing of an implicit time step into two, separately calculated, systems. This concept should be compatible with many features of multi-scale methods, (*i.e.*, making the large- and small-scale systems use adaptive decompositions). However, we will not include any, to avoid additional complications. The numerical experiments will be kept as simple as possible, as well.

To explain our method in detail requires a certain level of notation. We shall look at a one-dimensional example, though the general concept behind the method can be expanded to any number of dimensions. For instance, our second numerical example uses a decomposition of a two-dimensional domain Ω via tensor product between a Fourier decomposition in the x direction and a wavelet decomposition in the y direction. This allows the small-scale domain Λ to be of the form $(0, 2\pi) \times (4.5, 7.0)$, *i.e.*, to cover the entire x domain and a subset of the y domain. If the decomposition was via a tensor product with wavelet decompositions in both directions, then we could use a subdomain Λ that is smaller, with finer resolution, in either direction, or both.

The actual functions we shall be dealing with are those of the form

$$f(x) = \sum_k a_k \phi_{0,k}(x) + \sum_j \sum_k b_{j,k} \psi_{j,k}(x) \quad (4.1)$$

Recall that wavelet decompositions are in the spaces V_j and W_j , $j \in \mathbb{Z}$, as defined in (2.3) and (2.4). In Section 2 we discussed how there are multiple ways of expressing a function via wavelet decomposition. Since the specific form of the decomposition is relevant to any wavelet-related method, a means of expressing these differences becomes helpful.

All functions in the space V_j and W_j can be characterized in terms of the coefficients of their wavelets and scaling functions. The wavelet (ψ) coefficients are written $b_{j,k}$, the scaling function (ϕ) coefficients are written $a_{j,k}$, with j relating to resolution and k relating to location. The coefficients $b_{j,k}$ are best organized into vectors of the form

$$\mathbf{b}_j = \begin{bmatrix} b_{j,0} & b_{j,1} & b_{j,2} & \cdots & b_{j,K_j} \end{bmatrix} \quad (4.2)$$

The same can be done with the sets of coefficients $a_{j,k}$, creating vectors $\mathbf{a}_j$ of the form

$$\mathbf{a}_j = \begin{bmatrix} a_{j,0} & a_{j,1} & a_{j,2} & \cdots & a_{j,K_j} \end{bmatrix} \quad (4.3)$$

Any decomposition of the form in (4.1) has an equivalent vector

$$\begin{bmatrix} \mathbf{b}_{j+1} & \mathbf{b}_{j+2} & \cdots & \mathbf{b}_1 & \mathbf{b}_0 & \mathbf{a}_0 \end{bmatrix} \quad (4.4)$$

Any function $f \in V_j$ can also be decomposed directly into V_j via

$$f(x) = \sum_k a_k \phi_{j,k}(x) \quad (4.5)$$

with a vector $\mathbf{a}_j$, which has the same number of components as the vector in (4.4). The setup discussed in Section 3 focused on two different levels of resolution, V_j and V_{j+m} , with $j, m < 0$. One helpful way of writing this is to define a vector $\mathbf{a}$, containing all the necessary coefficients for V_j , and a vector $\mathbf{b}$ containing all the necessary coefficients for the spaces W_j to W_{j+m+1} . This setup, using $j = -4$ and $m = -2$, would result in vectors of the form

$$\mathbf{a} = \begin{bmatrix} \mathbf{b}_{-3} & \mathbf{b}_{-2} & \mathbf{b}_{-1} & \mathbf{b}_0 & \mathbf{a}_0 \end{bmatrix}^T, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{-5} & \mathbf{b}_{-4} \end{bmatrix}^T$$

The result of this setup is that any function in V_j can be expressed in terms of a vector of the form $\begin{bmatrix} \mathbf{0} & \mathbf{a} \end{bmatrix}^T$, and any function in V_{j+m} can be expressed in terms of a vector $\begin{bmatrix} \mathbf{b} & \mathbf{a} \end{bmatrix}^T$.

Next, we have two different domains to use: Ω and $\Lambda \subset \Omega$. The vector $\mathbf{a}$ has all the coefficients from the V_j space over the entire domain Ω . Now we must consider vectors relating only to $\Lambda \subset \Omega$. Whether a coefficient $a_{j,k}$ can be considered relating to Λ depends on its scaling function $\phi_{j,k}$. We use biorthogonal scaling functions and wavelets that are symmetric, with finite support. As a result, we designate $\phi_{j,k}$ or $\psi_{j,k}$ to be within Λ if the center of the function's support is strictly within Λ . Other options would probably work, such as defining $\phi_{j,k}$ or $\psi_{j,k}$ to be in Λ if their supports intersect Λ . A subscript Λ , for $\mathbf{a}_\Lambda$, restricts the coefficients in $\mathbf{a}$ to those relating to the smaller domain Λ . This will be standard practice throughout this paper. If a subscript Λ is added to a vector or matrix, that means that the vector or matrix is restricted to the smaller domain Λ , with coefficients relating only to Λ .

Now we take a linear partial differential equation of the form

$$u_t(x, t) = Lu(x, t), \quad x \in \Omega, \quad t \in (0, \infty) \quad (4.6)$$

where L is a linear differential operator

$$L = \sum_{k=0}^K g_k(x) \frac{d^k}{dx^k} \quad \text{for some continuous functions } g_k \quad (4.7)$$

As L is linear, we discretize it using a matrix M . The idea is for M to calculate the effect of L restricted to V_{j+m} . So, if the function $u(x, t) \in V_{j+m}$ is expressed in terms of $\begin{bmatrix} \mathbf{b} & \mathbf{a} \end{bmatrix}$, then

$$(Lu(x, t))|_{V_{j+m}} = M \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} \quad (4.8)$$

This matrix M can be calculated via a Galerkin or collocation method, so long as it is accurate in approximating L . Our earliest tests of this method used a Galerkin method. In this paper, the examples will be calculated via a collocation method, discussed later in Subsection 7.2. Boundary conditions are a potential issue. To keep the equations in this section clean, we will be ignoring the boundary conditions, or assuming that they are satisfied via the wavelet decomposition and M . Our later examples use the method effectively on a problem with zero boundary conditions and on a problem with constant,

non-zero, boundary conditions. Our approach for satisfying zero boundary conditions is discussed in Subsection 7.2.

The method is meant for use with an implicit time-discretization. Here, and in the rest of the paper, we shall illustrate the implementation of the method using the second order Adams-Moulton scheme for the time-discretization. This particular scheme is ideal since it is simple and, most importantly, implicit. Combining that time-discretization with the matrix M gives us the time step

$$\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^n \\ \mathbf{a}^n \end{bmatrix} + \frac{h}{2} M \left(\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^n \\ \mathbf{a}^n \end{bmatrix} \right) \quad (4.9)$$

We also use the block decomposition

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \quad \text{thus} \quad M \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} A\mathbf{b} + B\mathbf{a} \\ C\mathbf{b} + D\mathbf{a} \end{bmatrix} \quad (4.10)$$

Assume that the vectors $\mathbf{b}$ and $\mathbf{a}$ have $n_{\mathbf{b}}$ and $n_{\mathbf{a}}$ components, respectively. The matrix A will have $n_{\mathbf{b}}$ rows and columns and D will have $n_{\mathbf{a}}$ rows and columns. The matrix B will have $n_{\mathbf{a}}$ rows and $n_{\mathbf{b}}$ columns, and C will have $n_{\mathbf{b}}$ rows and $n_{\mathbf{a}}$ columns. Note that while A and D are square, they do not have to be equal in size. In fact, our main example will usually give A four times as many rows and columns as D .

Now we separate the time step in Equation (4.9) into two systems. First, we have the large-scale system, with functions in the space V_j over Ω , characterized by vectors of the form $\mathbf{a}$. We also have the small-scale system with functions in V_{j+m} over Λ , characterized by vectors of the form $\begin{bmatrix} \mathbf{b}_{\Lambda} \\ \mathbf{a}_{\Lambda} \end{bmatrix}$. Recall that $\mathbf{a}_{\Lambda}$ has only the components of $\mathbf{a}$ corresponding to the functions $\phi_{j,k}$ in Λ . The large-scale system is at the limited resolution V_j , and so uses the matrix D to model the operator L from (4.7). The small-scale system is expressed in terms of a matrix M_{Λ} of the form

$$M_{\Lambda} = \begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \quad (4.11)$$

where A_{Λ} , B_{Λ} , C_{Λ} and D_{Λ} are composed of the elements from A , B , C and D that are related to Λ .

Example 4.1 Using the domains $\Omega = (0, 10)$ and $\Lambda = (3.75, 6.25)$ and the space V_{-3} , we shall take a look at getting D_{Λ} from D .

In this example, the vector $\mathbf{a}$ has 79 components: The functions $\phi_{-3,k}$ are at points separated by a distance of $\frac{1}{8}$, and so are at points $\frac{1}{8}, \frac{2}{8}, \dots, \frac{79}{8}$. The matrix D , which is a discretization for the differential operator L restricted to V_{-3} , is a square matrix. Multiplication (on the left) by this matrix takes the $a_{-3,k}$ coefficients from a function $f \in V_{-3}$ in the vector $\mathbf{a}$, creating the vector $D\mathbf{a}$, the V_{-3} approximation of Lf . We can write it as

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,78} & d_{1,79} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,78} & d_{2,79} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{78,1} & d_{78,2} & \cdots & d_{78,78} & d_{78,79} \\ d_{79,1} & d_{79,2} & \cdots & d_{79,78} & d_{79,79} \end{bmatrix}$$

The matrix D_Λ only operates on the coefficients of functions $\phi_{-3,k}$ restricted to Λ . As a result, the number of rows and columns of D_Λ have to be equal to the number of $a_{-3,k}$ coefficients in Λ . As a result, it is smaller:

$$D_\Lambda = \begin{bmatrix} d_{31,31} & d_{31,32} & \cdots & d_{31,48} & d_{31,49} \\ d_{32,31} & d_{32,32} & \cdots & d_{32,48} & d_{32,49} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{48,31} & d_{48,32} & \cdots & d_{48,48} & d_{48,49} \\ d_{49,31} & d_{49,32} & \cdots & d_{49,48} & d_{49,49} \end{bmatrix}$$

The first step is to use a time-discretization to find a temporary approximation of $\mathbf{a}^{n+1}$, which we shall call $\mathbf{a}^{T_m}$. Using the second-order Adams-Moulton method yields the large-scale system

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2} D (\mathbf{a}^n + \mathbf{a}^{T_m}) \quad (4.12)$$

This calculates the time step on Ω at the large-scale resolution V_j . What we want next is to calculate the time step on Λ at the small-scale resolution V_{j+m} . However, $\mathbf{a}^n$ and the newly calculated $\mathbf{a}^{T_m}$ have to be included. So, we use the components of $\mathbf{a}^n$ and $\mathbf{a}^{T_m}$ that are in Λ , which we call $\mathbf{a}_\Lambda^n$ and $\mathbf{a}_\Lambda^{T_m}$, respectively. These are used in the system

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} (\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n) \quad (4.13)$$

Note the presence of $\mathbf{a}_\Lambda^{C_r}$, a corrector term for the temporary approximation $\mathbf{a}^{T_m}$. The final step is to take $\mathbf{a}_\Lambda^{C_r}$, rewrite it in Ω as $\mathbf{a}^{C_r}$, and add it to $\mathbf{a}^{T_m}$ for

$$\mathbf{a}^{n+1} = \mathbf{a}^{T_m} + \mathbf{a}^{C_r} \quad (4.14)$$

This setup is consistent with the time-discretization. Since $\mathbf{a}_\Lambda^{n+1} = \mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{C_r}$, the restriction of $[\mathbf{b}^{n+1} \ \mathbf{a}^{n+1}]^T$ to Λ will be

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} + \mathbf{a}_\Lambda^{T_m} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda (\mathbf{a}_\Lambda^n + \mathbf{a}_\Lambda^{T_m}) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ (\mathbf{a}^n + \frac{h}{2} D (\mathbf{a}^n + \mathbf{a}^{T_m}))|_\Lambda \end{bmatrix} \quad (4.15)$$

Next, we will rewrite that expression using

$$\mathbf{a}^n + \frac{h}{2} D (\mathbf{a}^n + \mathbf{a}^{T_m}) \Big|_\Lambda = \mathbf{a}_\Lambda^n + \frac{h}{2} D_\Lambda (\mathbf{a}_\Lambda^n + \mathbf{a}_\Lambda^{T_m}) + \frac{h}{2} (D(\mathbf{a}^{n+1} + \mathbf{a}^n)_{\Lambda^C})_\Lambda \quad (4.16)$$

The last term in (4.16) is going to be smaller than the others, considering the restriction to Λ^C followed by a restriction to Λ . In the context of the small-scale system, the last term in (4.16) can be viewed as representing influences that are similar to boundary conditions: Influences consistent with the problem that come from outside the region being calculated (Λ , in this case). This term, being somewhat

cumbersome to write and read, will not be written in the next expressions. We will simply replace the first term in (4.16) with the second (they are, at least, approximately equal). This substitution yields

$$\begin{aligned} \begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} + \mathbf{a}_{\Lambda}^{T_m} \end{bmatrix} &= \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^n + \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_{\Lambda} (\mathbf{a}_{\Lambda}^n + \mathbf{a}_{\Lambda}^{T_m}) \\ \mathbf{0} \end{bmatrix} \\ &\quad + \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\Lambda}^n + \frac{h}{2} D_{\Lambda} (\mathbf{a}_{\Lambda}^n + \mathbf{a}_{\Lambda}^{T_m}) \end{bmatrix} \\ \begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{n+1} \end{bmatrix} &= \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_{\Lambda}^n \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^n + \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^n + \mathbf{a}_{\Lambda}^{T_m}) \\ &= \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_{\Lambda}^n \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^n + \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{n+1} + \mathbf{a}_{\Lambda}^n \end{bmatrix} \end{aligned} \quad (4.17)$$

which matches the second order Adams-Moulton discretization.

5. Error Sources

Now we take a closer look at the main sources of error. Recall, from (4.9), that the full resolution, full domain, time-discretization for a linear problem is

$$\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^n \\ \mathbf{a}^n \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \left(\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^n \\ \mathbf{a}^n \end{bmatrix} \right) \quad (5.1)$$

The multi-scale method makes use of the components of the vectors $\mathbf{a}$ and $\mathbf{b}$, as well as the elements of the matrices A to D , that relate to the sub-domain Λ . Now we will have to discuss the sub-domain $\Omega \cap \Lambda^C$, and related vectors/matrices. We will now subdivide the vectors as follows:

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_{\Lambda} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_{\Lambda} \end{bmatrix} \quad (5.2)$$

with $\mathbf{a}_1$ and $\mathbf{b}_1$ the components from $\Omega \cap \Lambda^C$. The related matrix block decomposition is

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_{\Lambda} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_{\Lambda} \end{bmatrix} \quad (5.3)$$

and the same for C and D . The idea is to have

$$A\mathbf{b} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_{\Lambda} \end{bmatrix} = \begin{bmatrix} A_1\mathbf{b}_1 + A_2\mathbf{b}_{\Lambda} \\ A_3\mathbf{b}_1 + A_{\Lambda}\mathbf{b}_{\Lambda} \end{bmatrix} \quad (5.4)$$

$$D\mathbf{a} = \begin{bmatrix} D_1 & D_2 \\ D_3 & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_{\Lambda} \end{bmatrix} = \begin{bmatrix} D_1\mathbf{a}_1 + D_2\mathbf{a}_{\Lambda} \\ D_3\mathbf{a}_1 + D_{\Lambda}\mathbf{a}_{\Lambda} \end{bmatrix} \quad (5.5)$$

and so on. Written this way, the time-discretization becomes

$$\begin{bmatrix} \mathbf{b}_1^{n+1} \\ \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_{\Lambda}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^n \\ \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_1^n \\ \mathbf{a}_{\Lambda}^n \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_1 & A_2 & B_1 & B_2 \\ A_3 & A_{\Lambda} & B_3 & B_{\Lambda} \\ C_1 & C_2 & D_1 & D_2 \\ C_3 & C_{\Lambda} & D_3 & D_{\Lambda} \end{bmatrix} \left(\begin{bmatrix} \mathbf{b}_1^{n+1} \\ \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_{\Lambda}^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1^n \\ \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_1^n \\ \mathbf{a}_{\Lambda}^n \end{bmatrix} \right) \quad (5.6)$$

Now compare that with the multi-scale method. The large-scale step is

$$\begin{bmatrix} \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^n \\ \mathbf{a}_\Lambda^{T-M} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} D_1 & D_2 \\ D_3 & D_\Lambda \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} + \begin{bmatrix} \mathbf{a}_1^n \\ \mathbf{a}_\Lambda^n \end{bmatrix} \right) \quad (5.7)$$

Next we look at the small-scale system,

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} (\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{C_r}) \quad (5.8)$$

Combining those two steps results in the full time stepping scheme

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{a}_1^n \\ \mathbf{a}_\Lambda^n \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & O & B_\Lambda \\ O & D_1 & D_2 \\ C_\Lambda & D_3 & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \\ \mathbf{a}_1^{n+1} + \mathbf{a}_1^n \\ \mathbf{a}_\Lambda^{n+1} + \mathbf{a}_\Lambda^n \end{bmatrix} - \begin{bmatrix} O \\ D_2 \\ O \end{bmatrix} \mathbf{a}_\Lambda^{C_r} \right) \quad (5.9)$$

So, the error inherent in the multi-scale system comes from three distinct sources:

1. The vectors $\mathbf{b}_1$ not being included at all. In order to keep the error below a specified tolerance, we would need to implement an adaptive procedure. This would involve calculating the values of the coefficients $b_{j,k}$ in $\mathbf{b}_1$ that are adjacent to the domain Λ . The domain Λ would be expanded if these values were found to have exceeded the specified threshold. Other coefficients $b_{j,k}$ in $\mathbf{b}_1$, further from Λ , would also be calculated. If these were found to be above the specified threshold, then the large-scale system would be given a finer resolution.
2. The matrices C_2 and B_3 from (5.6) being omitted from (5.9). This source of error is reduced by widening the vectors $\mathbf{a}_\Lambda$ at the expense of $\mathbf{a}_1$ (i.e., including extra large-scale coefficients in the calculation of the small-scale system). We call these “extra” terms, and they are explained in detail in Section 8. These terms reduce the error because the matrices A to D will be diagonally dominant (due to the fact that ϕ and ψ are well centered). Consider the block decomposition of

$$C = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_\Lambda \end{bmatrix}$$

Increasing the number of components in $\mathbf{a}_\Lambda$ increases the number of elements in C_Λ . It also reduces the number of elements in C_1 , C_2 , and C_3 . The matrix C_2 will have rows of elements removed, effectively restricting C_2 to the top right corner of C . Since the elements in C are smaller further from the diagonal, the result is a matrix C_2 that has both fewer elements and smaller valued elements. As a result, $\|C_2 \mathbf{b}_\Lambda\|$ becomes smaller as $\mathbf{a}_\Lambda$ is expanded.

The best way to control this source of error is an adaptive scheme that checks the largest values of $C_2 \mathbf{b}_\Lambda$ and $B_3 \mathbf{a}_1$. The number of “extra” terms would be increased if either calculation was above a given tolerance.

3. The last type of error is from the last vector term in (5.9). This source of error is best managed via the “extra” terms, as well (i.e., by increasing the number of components in $\mathbf{a}_\Lambda$). Including more “extra” terms will result in D_2 having fewer elements, as well as smaller elements (as with C_2 and

B_3 above). One other thing to notice is that $\mathbf{a}_\Lambda^{C_r}$ is a corrector term, and so, in general, is of smaller magnitude than $\mathbf{a}_\Lambda^n$. We have

$$-\frac{h}{2} D_2 \mathbf{a}_\Lambda^{C_r} = -\frac{h^2}{4} D_2 [C_\Lambda(\mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1}) + D_\Lambda \mathbf{a}_\Lambda^{C_r}]$$

so it is order h^2 , and a reasonable time step will keep its magnitude low.

Managing these sources of error should be sufficient to control the error stemming from the method itself, (*i.e.*, control the error stemming from limiting the small-scale resolution to Λ and from dividing the time step into two different systems). Other sources of error, those that are not a direct result of the multi-scale method itself, would be addressed separately. If a particular approach for keeping the time step small enough (or the resolution fine enough) is compatible with a wavelet decomposition and an implicit time-discretization, it should prove compatible with our multi-scale method.

6. A Test Problem

To illustrate the application of the method, and confirm that it can work, we test it on a one dimensional problem involving Burgers' equation. Burgers' equation is appropriate because it is not linear, its expression is fairly simple, and because it can be easily set up to require fine resolution in a small region and so benefit from the method we are testing. Note that we are deliberately keeping things very simple. We will use fixed Λ subdomains, and fixed resolutions. The method looks to be compatible with many types of adaptive schemes, and other expansions. However, in this paper, we want to ensure that the method can deliver consistent results, with no additional complications. We therefore use the simplest version of the method, without these additional features.

We solve for the function $u(x, t)$, $x \in [0, 10]$, $t \in [0, \infty)$, satisfying

$$u_t = -uu_x + \nu u_{xx}, \quad x \in \Omega = (0, 10), \quad t \in (0, \infty) \quad (6.1)$$

with boundary and initial conditions

$$u(0, t) = u(10, t) = 0, \quad t \in (0, \infty) \quad \text{and} \quad u(x, 0) = f(x), \quad x \in [0, 10] \quad (6.2)$$

The function $f(x)$ is plotted in Figure 3 (a). The exact value of $f(x)$ is

$$f(x) = \sum_{k=1}^9 f_k \phi_{k,0}(x), \quad [f_1, f_2, f_3, \dots, f_9] = [0, 1, 2, 1, 0, -1, -2, -1, 1, 0]$$

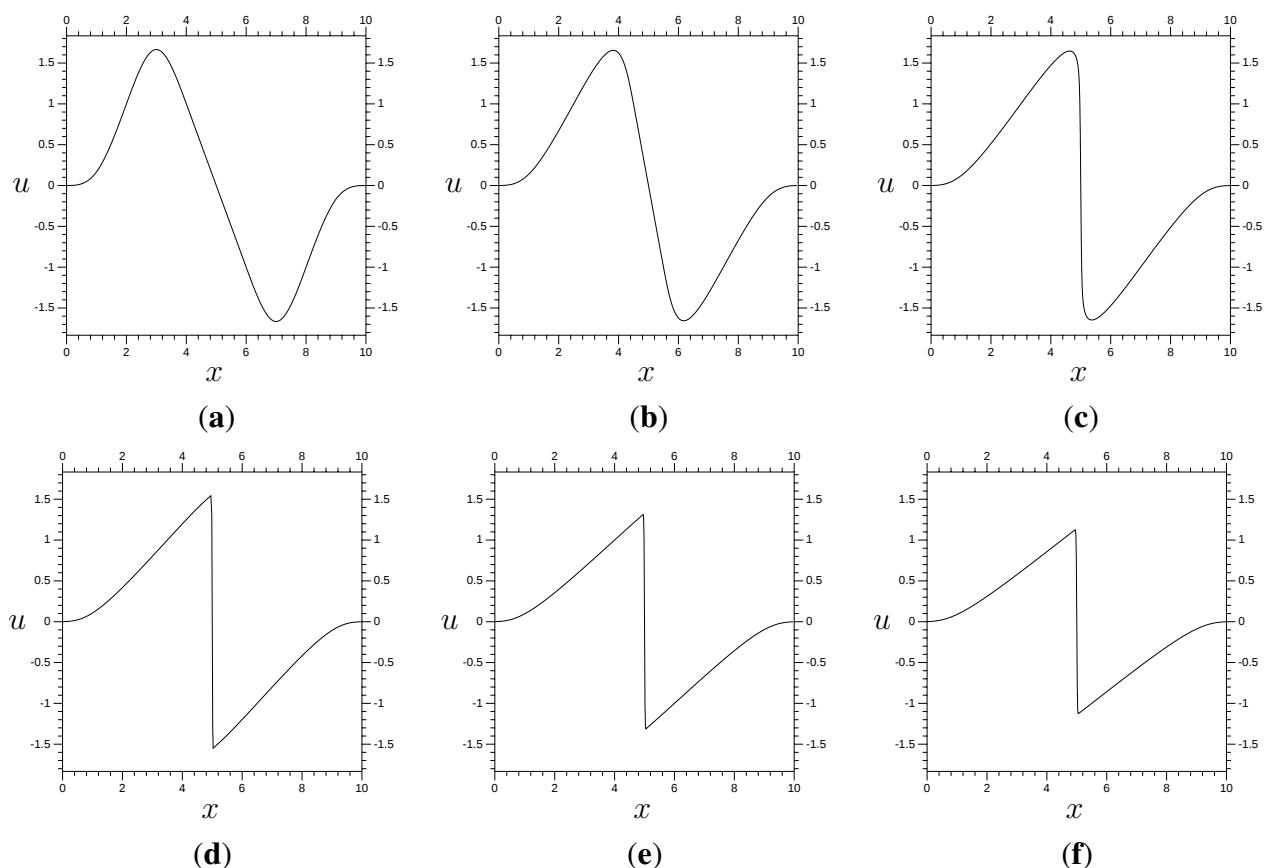
using ϕ from Figure 1. This choice of initial condition will require some explanation. First, we use an initial condition in V_0 to simplify the required programs and to avoid any error at $t = 0$. This means that all different models/resolutions start at zero error. The general shape chosen relates to the behavior of Burgers' equation, mainly the effect of the $-uu_x$ term. If we set the viscous parameter, ν , to zero then the solution becomes

$$u(x, t) = f(x - u(x, t)t)$$

meaning that the characteristics will start at points $(x, 0)$ and have slopes equal to $\frac{dx}{dt} = f(x)$. Of course, if $\nu > 0$ then this effect will be changed, but the basic nature, and directions, will be much the same.

Now consider what this means for $f(x)$ in Figure 3. As shown in this figure, the two peaks will approach each other as the time increases, colliding at $x = 5$. What happens when they meet depends on the magnitude of ν . If $\nu = 0$, then the characteristics will intersect, resulting in a singular solution. If $\nu > 0$ then the peaks will collide, but not cause an infinite slope. Instead, the translation will cease, and the solution will converge to zero under the influence of the viscosity (see Figure 3). So, instead of an infinite slope at $x = 5$, the solution will have a finite slope dependent on the magnitude of ν . Increase ν and the resulting slope at $x = 5$ will get smaller. Decrease ν and the slope at $x = 5$ will be greater. A very steep slope at $x = 5$ will make proper expression of the solution very difficult for a coarse resolution V_j space. The results shown here, with $\nu = 0.01$, require at least V_{-6} for reasonable results (using our wavelet/scaling function pair).

Figure 3. Results for Burgers' equation with V_{-6} resolution and time step $h = 0.01$.
(a) $t = 0.0$; (b) $t = 0.5$; (c) $t = 1.0$; (d) $t = 1.5$; (e) $t = 2.0$; (f) $t = 2.5$.



7. Calculation

7.1. Time-Discretization

As before, we shall use the second order Adams-Moulton method for our time-discretization:

$$\begin{aligned} u^{n+1} &= u^n + \frac{h}{2} (u_t^n + u_t^{n+1}) \\ &= u^n + \frac{h}{2} [\nu (u_{xx}^n + u_{xx}^{n+1}) - u^n u_x^n - u^{n+1} u_x^{n+1}] \end{aligned} \quad (7.1)$$

meaning we consider the functions $u^n(x) = u(x, nh)$ using the time step h . The main problem with this time-discretization is dealing with the non-linear term $u^{n+1}u_x^{n+1}$. To avoid additional complications arising from solving this non-linear system, we shall use the linearization

$$u^n u_x^n + u^{n+1} u_x^{n+1} = u^n u_x^{n+1} + u^{n+1} u_x^n + O(h^2) \quad (7.2)$$

Using this, we have the time-discretization

$$u^{n+1} = u^n + \frac{h}{2} \left[\nu \frac{\partial^2}{\partial x^2} (u^n + u^{n+1}) - \left(u^{n+1} \frac{\partial}{\partial x} u^n + u^n \frac{\partial}{\partial x} u^{n+1} \right) \right] + O(h^3) \quad (7.3)$$

with the non-linear term replaced by the linear operator $\left(\frac{\partial}{\partial x} u^n + u^n \frac{\partial}{\partial x} \right)$ applied to u^{n+1} . Furthermore, since the approximation (7.2) has the same order as the time-discretization, the loss of accuracy should be negligible, especially for a small time step h .

7.2. Physical Decomposition

The physical decomposition is in a space V_j , for $j \in \mathbb{Z}$, $j < 0$, using the functions ϕ and ψ in Figure 1. We use a collocation method, though the earliest experiments were done via a Galerkin method, and so both are compatible. Our collocation method involves matching the exact values of $u^n(x)$ and $u^{n+1}(x)$ from (7.3) at the points $x = 2^j k$, $k \in \mathbb{Z}$. Since we are using the domain $\Omega = (0, 10)$, with $u(0, t) = u(10, t) = 0$, that means $10 \cdot 2^{-j} - 1$ points to calculate at every time step. The number of coefficients $a_{j,k}$ and $b_{j,k}$ used will be the same as the number of collocation points, that is $10 \cdot 2^{-j} - 1$ for V_j . Our standard model will use V_{-6} , so that is 639 coefficients.

The boundary conditions are zero, with $u_{xx}(x, t) \rightarrow 0$ as $x \rightarrow 0$ and $x \rightarrow 10$, to maintain consistency with (6.1). We keep the decomposition consistent with these boundary conditions in the simplest way possible. To demonstrate, we will look to the $x = 0$ boundary. What we do is reshape every $\phi_{j,k}$ supported at $x = 0$ as follows:

$$\phi_{j,k}(x) = 2^{-j/2} \phi(2^{-j}x - k) - 2^{-j/2} \phi(2^{-j}x + k) \quad (7.4)$$

rather than the usual $2^{-j/2} \phi(2^{-j}x - k)$. Note that all that is being changed is that $\phi_{j,k}$ is, effectively, being subtracted by $\phi_{j,-k}$, the function with the same shape but from exactly the other side of $x = 0$. Our scaling function, ϕ , is that from Figure 1. It is symmetric around zero, so this makes $\phi_{j,k}$ anti-symmetric around $x = 0$, so

$$\phi_{j,k}(0) = 0, \quad \phi_{j,k}''(0) = 0$$

The same change is made to the $\phi_{j,k}$ that are supported at $x = 10$. They have their counterpart from exactly the other side of $x = 10$ subtracted from them, resulting in

$$\phi_{j,k}(10) = 0 \quad \phi_{j,k}''(10) = 0$$

The process for $\psi_{j,k}$ is much the same, with a small change because ψ is symmetric around $x = \frac{1}{2}$, meaning $\psi_{j,k}$ will have to be modified by $-\psi_{j,1-k}$ to keep $\psi_{j,k}(0) = 0$.

The derivatives will be approximated using matrices, of course, created out of some common components. First, the Fast Wavelet Transform and Inverse Fast Wavelet Transform are calculated via

$$M_{FWT} \mathbf{a}_j = \begin{bmatrix} \mathbf{b}_{j+1} \\ \vdots \\ \mathbf{b}_{m+1} \\ \mathbf{a}_m \end{bmatrix}, \quad M_{IFWT} \begin{bmatrix} \mathbf{b}_{j+1} \\ \vdots \\ \mathbf{b}_{m+1} \\ \mathbf{a}_m \end{bmatrix} = \mathbf{a}_j \quad (7.5)$$

where $m > j$ is the resolution of the scaling functions used in the wavelet decomposition.

Next, we have M_Σ^d , a matrix that when multiplied by $\mathbf{a}_j$ will output a vector of values of $\frac{d^d}{dx^d} u$ (where $\mathbf{a}_j$ contains the $a_{j,k}$ coefficients of u) at the collocation points. So, if

$$u(x) = \sum_k a_{j,k} \phi_{j,k}(x) \quad (7.6)$$

with the $a_{j,k}$ in $\mathbf{a}_j$, then $M_\Sigma^0 \mathbf{a}_j$ will be a vector composed of values $u(2^j m)$, $m \in \mathbb{Z}$.

Putting these together, we get

$$M_x = M_{FWT} (M_\Sigma)^{-1} D_\Sigma^1 M_{IFWT}, \quad M_{xx} = M_{FWT} (M_\Sigma)^{-1} D_\Sigma^2 M_{IFWT} \quad (7.7)$$

matrices that approximate the first and second x derivatives.

The collocation arrangement for the linearized uu_x term is based on the same matrices, and one more component:

$$\text{Diag}(\mathbf{v}) = \text{diagonal matrix with entries equal to the vector } \mathbf{v} \quad (7.8)$$

We approximate the nonlinear terms via the operator

$$N_L = -M_{FWT} (M_\Sigma)^{-1} [\text{Diag}(M_\Sigma M_{IFWT} M_x \mathbf{u}^n) M_\Sigma M_{IFWT} + \text{Diag}(M_\Sigma M_{IFWT} \mathbf{u}^n) M_\Sigma M_{IFWT} M_x] \quad (7.9)$$

7.3. Results

The results obtained using V_{-6} are in Figure 3. These are what we will usually call our “control results”, which we will use for comparison with those using our multi-scale method (Section 4), which will use V_{-6} as their finest resolution. The multi-scale results will attempt to duplicate the control results while taking less computational time (the control results required approximately one second per time step). The error resulting only from the multi-scale method will be the difference between the multi-scale results and the control results, since that will be the effect of ignoring the fine resolution coefficients outside of Λ , and the effect of dividing the time step calculations into two systems. So, this “error” term, the difference between the control results and the multi-scale results, will be our main focus in calculations. However, the accuracy of these V_{-6} control results may provide some useful perspective. For comparison, we solved the problem numerically using a physical decomposition in V_{-8} (so four times as many coefficients), and with a time step size of $h = 0.002$ (so five times as many time steps).

The calculation of a single time step with this model was in excess of seventy seconds. At $t = 2.0$, the difference between the V_{-8} results and the V_{-6} control results was, in the indicated L_p norms,

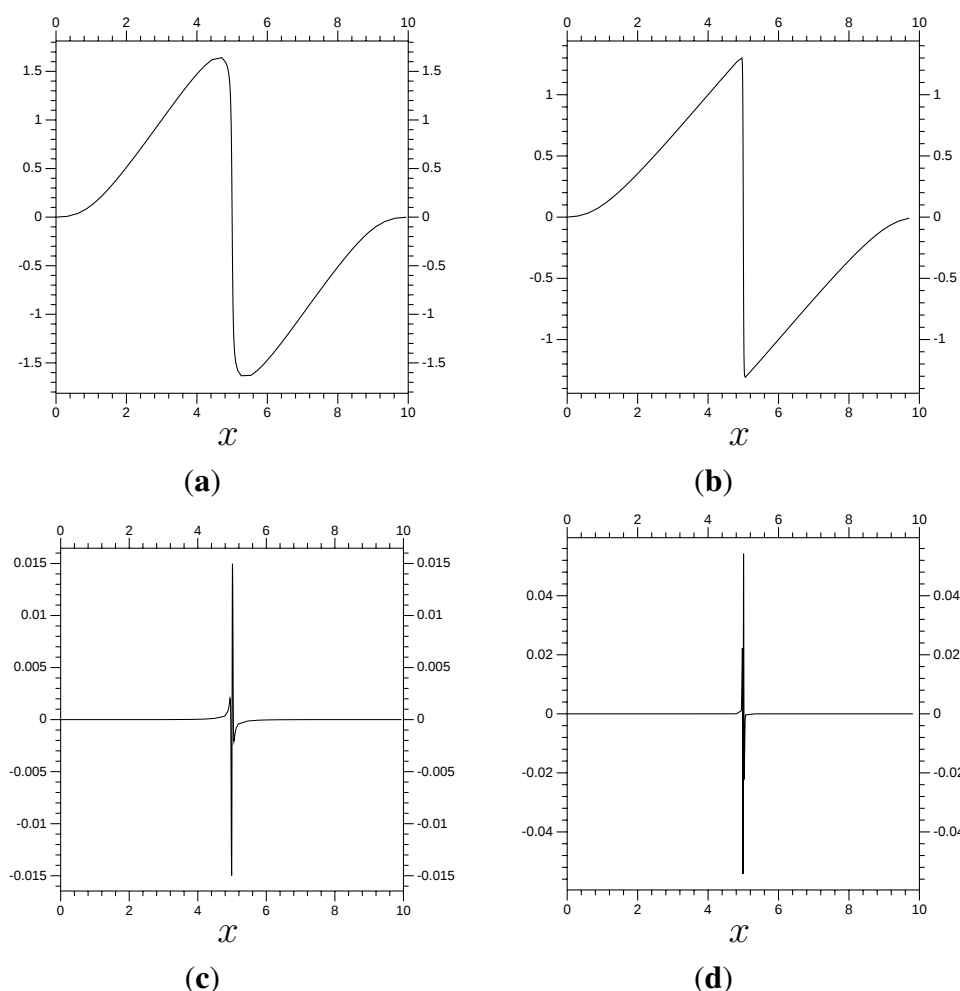
$$8.0 \times 10^{-3} \text{ using } L_2, \quad 2.1 \times 10^{-3} \text{ using } L_1, \quad 5.4 \times 10^{-2} \text{ using } L_\infty$$

At $t = 1.0$ it was

$$2.9 \times 10^{-3} \text{ using } L_2, \quad 1.2 \times 10^{-3} \text{ using } L_1, \quad 1.5 \times 10^{-2} \text{ using } L_\infty$$

Plots of the V_{-8} results, and their differences with the V_{-6} control results, can be found in Figure 4.

Figure 4. Results for Burgers' equation using V_{-8} , and time step size $h = 0.002$, (a) at $t = 1.0$ and (b) at $t = 2.0$. Next, the difference between the V_{-8} results and the V_{-6} results, (c) at $t = 1.0$ and (d) at $t = 2.0$.



Now we try to reduce the expense of calculating the V_{-6} results. The solution does not require fine resolution outside of the center. As a result, for smaller (more negative) values j , the coefficients $b_{j,k}$ away from $x = 5$ will be very small. So, one could use the method from Section 4, with a resolution of V_{-4} over all $\Omega = (0, 10)$ and have W_{-4} and W_{-5} only on $\Lambda = (4, 6)$. This arrangement would have 159 elements for the V_{-4} large-scale system, as well as $2(2^4 + 2^4 + 2^5) = 128$ V_{-4} , W_{-4} and W_{-5} terms for the small-scale system on $\Lambda = (4, 6)$. This makes for a total of 287 elements, solved in two, similarly

sized, systems. The control results have a full V_{-6} resolution on $\Omega = (0, 10)$, so 639 coefficients, with many contributing very little to the accuracy.

8. Further Improvements

The basic outline of the method is only the beginning, and it will need to be improved for the actual implementation. First, the basic method is for a linear problem, but the test problem is non-linear. Although we linearized it using (7.2), there are still some non-trivial difficulties arising from the linearized terms. Properly calculating the linearized terms in the two systems, and keeping those calculations consistent with each other, will be the first improvement. The second improvement is more general, suitable for any problem, and relates to the coefficients in $\mathbf{a}_\Lambda$, those that are common to both the large and small-scale systems.

8.1. Non-Linear Terms

The method described in Section 4 was, from the beginning, set up using a single matrix discretization of the operator L : The matrix M is created at V_{j+m} resolution and over all Ω . This is the best way to deal with linear operators. However, it is not a good option for dealing with any non-linear terms.

Recall that we use the linearization from (7.2),

$$u^{n+1} u_x^{n+1} + u^n u_x^n \approx u^{n+1} u_x^n + u^n u_x^{n+1} = \left(\frac{\partial}{\partial x} u^n + u^n \frac{\partial}{\partial x} \right) u^{n+1} \quad (8.1)$$

to avoid having to solve a system involving the non-linear term $u^{n+1} \frac{\partial}{\partial x} u^{n+1}$. Using this approximation involves creating a linear operator that calculates the multiplication by $(u^n \frac{\partial}{\partial x} + \frac{\partial}{\partial x} u^n)$, then applying that operator to the value u^{n+1} in the time step. The linearization requires creating a new linear operator at every time step, of the form from (7.9). To do so at V_{j+m} resolution over all Ω at every single time step would be time consuming. Instead, we create an operator at V_j resolution on Ω , call it T_Ω , and an operator at V_{j+m} resolution on Λ , S_Λ . These operators are created separately. The first one, T_Ω , is used in the large-scale system

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2} \left[D (\mathbf{a}^{T_m} + \mathbf{a}^n) + T_\Omega \mathbf{a}^{T_m} \right] \quad (8.2)$$

with T_Ω positive rather than negative due to the form used in (7.9). Note that we are keeping the linear terms, discretized as in (4.8) and block decomposed as in (4.10), hence the inclusion of D .

The second operator, S_Λ , created at the small-scale resolution over Λ , and so created independently of T_Ω , is used in the small-scale system

$$\begin{aligned} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} &= \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} (\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n) + S_\Lambda \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + S_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ (T_\Omega \mathbf{a}^{T_m})_\Lambda \end{bmatrix} \right) \end{aligned} \quad (8.3)$$

Note that O here represents a zero matrix and $\mathbf{0}$ is a zero vector. As usual, the last step is to compute $\mathbf{a}^{n+1} = \mathbf{a}^{T_m} + \mathbf{a}^{C_r}$.

One set of terms from (8.3),

$$S_{\Lambda} \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\Lambda}^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ (T_{\Omega} \mathbf{a}^{T_m})_{\Lambda} \end{bmatrix} \quad (8.4)$$

needs to be discussed in more detail. Notice that S_{Λ} is a matrix the same size as M_{Λ} (see (4.11)), so the vector on the left of (8.4) has components in the spaces W_j to W_{j+m+1} so it is not of the form $\begin{bmatrix} \mathbf{0} \\ \mathbf{a} \end{bmatrix}$. These components are not included in Equation (8.2), since that calculation is restricted to the large-scale resolution V_j , but these terms have to be included in the small-scale system. In this way, (8.4) is analogous to the term

$$\begin{bmatrix} B_{\Lambda} \\ O \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \quad (8.5)$$

found in Equation (4.13), applied to the linearized components of the problem. Both involve vectors used, and calculated, in the large-scale system, and their influence on the small-scale system. The linear version, (8.5), can be written as

$$\begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) - \begin{bmatrix} O \\ [D(\mathbf{a}^{T_m} + \mathbf{a}^n)]_{\Lambda} \end{bmatrix} \quad (8.6)$$

which is of the same form as (8.4), with $\begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} \end{bmatrix}$ in place of S_{Λ} and D in place of T_{Ω} . The first term on the left in (8.6) is equal to $M_{\Lambda} \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n \end{bmatrix}$, a term in the small-scale system. The second term in (8.6) is the vector $D(\mathbf{a}^{T_m} + \mathbf{a}^n)$ restricted to the sub-domain Λ . Since this has already been included in the calculations (in the large-scale system), it needs to be subtracted here in order to avoid being included twice. Equation (4.13) uses a simplified form of (8.6). If D_{Λ} is composed of elements of D , then (8.6) is equal to

$$\begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) - \begin{bmatrix} O \\ D_{\Lambda}(\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \end{bmatrix} = \begin{bmatrix} B_{\Lambda} \\ O \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \quad (8.7)$$

the same as the last term in (4.13). This process is accurate in the linear case when the matrices are created together (*i.e.*, since D and D_{Λ} are both from M). This process does not work for the non-linear case since the operators S_{Λ} and T_{Ω} are created separately, so the restriction of T_{Ω} to Λ is not equal to the large-scale related elements of S_{Λ} (*i.e.*, S_{Λ} restricted to V_j). As a result, (8.4) is necessary for non-linear problems.

Note that taking the matrix M from (4.8) and block decomposing it as per (4.10) is not the only way to calculating the matrix D and the matrices A_{Λ} to D_{Λ} . It is just the most accurate option. One could, for instance, calculate the matrix E to approximate the operator L at the large-scale resolution over Ω and, separately, calculate the matrices A_{Λ} to D_{Λ} to approximate L on Λ . This actually saves time, as smaller domains and coarser resolution reduce the size of the resulting systems substantially. This setup yields the large-scale system

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2} E (\mathbf{a}^n + \mathbf{a}^{T_m}) \quad (8.8)$$

Next, we get the small-scale system

$$\begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} + \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} + \begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} - E_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \right) \quad (8.9)$$

following the pattern from (8.6). If we, instead, assume that $D_{\Lambda} - E = O$, then the small-scale system becomes

$$\begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} + \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} + \begin{bmatrix} B_{\Lambda} \\ O \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \right) \quad (8.10)$$

However, using (8.9) results in errors around the boundaries of Λ , and when it is applied to linear problems (or components of problems) the errors can manifest as instability. This is particularly common with even derivatives (for example, viscosity terms). Using (8.10) reduces the error (and instability) at the boundary of Λ . The downside to (8.10) is persistent error in the center of Λ . The solution is to use both approaches simultaneously. We need (8.9) at the boundary and (8.10) in the center, and it is easy to do just that. We use a diagonal matrix, J , to convert smoothly between the two vectors

$$\begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} - E_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \quad \text{and} \quad \begin{bmatrix} B_{\Lambda} \\ O \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \quad (8.11)$$

What is needed is for J to be a diagonal matrix, with diagonal elements equal to one near the center, and zero at both ends, *i.e.*, a diagonal matrix J with diagonal elements

$$\text{diag}(J) = \left[0, \frac{1}{3}, \frac{2}{3}, 1, 1, \dots, 1, 1, \frac{2}{3}, \frac{1}{3}, 0 \right] \quad (8.12)$$

In place of either terms in (8.11), we use

$$\begin{bmatrix} I & O \\ O & J \end{bmatrix} \left(\begin{bmatrix} B_{\Lambda} \\ D_{\Lambda} - E_{\Lambda} \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \right) = \begin{bmatrix} B_{\Lambda} \\ J(D_{\Lambda} - E_{\Lambda}) \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \quad (8.13)$$

Consider the V_j components from (8.13) (as those that could be written in the form of a vector $\mathbf{a}$). The components nearest to the interface of Λ (in the current setup, those nearest to the top and bottom of the vector) will be equal to zero. Those in the center will be equal to $(D_{\Lambda} - E_{\Lambda}) (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n)$. There is also a gradual transition between the central and outer components. So, we get the small-scale system

$$\begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\Lambda}^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_{\Lambda} & B_{\Lambda} \\ C_{\Lambda} & D_{\Lambda} \end{bmatrix} \begin{bmatrix} \mathbf{b}_{\Lambda}^{n+1} + \mathbf{b}_{\Lambda}^n \\ \mathbf{a}_{\Lambda}^{C_r} \end{bmatrix} + \begin{bmatrix} B_{\Lambda} \\ J(D_{\Lambda} - E_{\Lambda}) \end{bmatrix} (\mathbf{a}_{\Lambda}^{T_m} + \mathbf{a}_{\Lambda}^n) \right) \quad (8.14)$$

which uses (8.9) in the center of Λ , for accuracy, and (8.10) near the interface of Λ , for stability.

This will be applied to nonlinear terms, using a diagonal matrix J_{NL} , of similar form to the matrix J in (8.12). The related term becomes

$$\begin{bmatrix} I & O \\ O & J_{NL} \end{bmatrix} \left(S_{\Lambda} \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{\Lambda}^{T_m} \end{bmatrix} - \begin{bmatrix} 0 \\ (T_{\Omega} \mathbf{a}^{T_m})|_{\Lambda} \end{bmatrix} \right) \quad (8.15)$$

(recall that the S_Λ matrix is the same size as M_Λ). The full small-scale system is

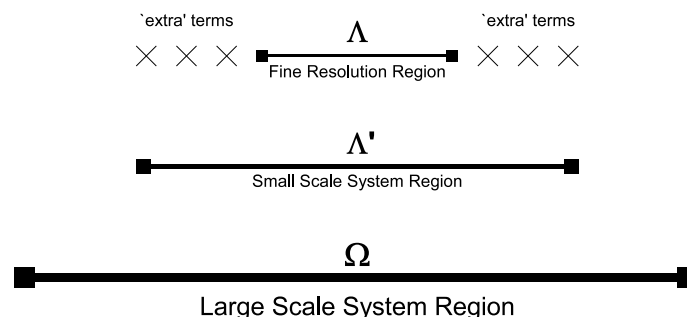
$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left\{ \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} (\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n) + S_\Lambda \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} I & O \\ O & J_{NL} \end{bmatrix} \left(S_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ (T_\Omega \mathbf{a}^{T_m})_\Lambda \end{bmatrix} \right) \right\} \quad (8.16)$$

As long as the matrices A , B , C and D are created from a single matrix M , there is no reason to use J on the linear terms from the problem.

8.2. “Extra” Terms

Our next modification to the systems is more complicated to implement than a matrix multiplication. Recall that our large-scale system is in the space V_j over all Ω , and the small-scale system is in the finer space V_{j+m} on the smaller domain Λ . By necessity, a lot of our focus is on the set of coefficients that they have in common, the set of coefficients of functions that are within both the large-scale system’s limited resolution and the small-scale system’s limited domain (so V_j functions within Λ). These coefficients are where most of the interactions between the two systems occur, where the coefficients/components in one system have an effect on those in the other system. Getting accurate interactions between the large and small-scale systems is key to getting accurate results with the method. This particular modification is about expanding that region of interaction, increasing the number of coefficients $a_{j,k}$ included in the small-scale system. These extra coefficients (and their associated functions $\phi_{j,k}$) are treated exactly the same as those in Λ . However, these extra terms (we shall frequently refer to them as simply “extra” coefficients $a_{j,k}$ or “extra” functions $\phi_{j,k}$) are not actually in Λ itself. We do not include any additional coefficients $b_{j,k}$ to $b_{j+m+1,k}$, and so we keep the small-scale resolution V_{j+m} restricted to the domain Λ . As a result, the increase in the size of the small-scale system (in terms of number of coefficients) is small, at least in terms of the original size of the system. This new feature creates a new domain Λ' , with $\Lambda \subset \Lambda' \subset \Omega$. The small-scale system calculates V_j resolution coefficients within Λ' , but keeps the V_{j+m} resolution within Λ itself. This is illustrated in Figure 5.

Figure 5. A simple diagram showing Ω , Λ , Λ' and the location of the “extra” terms.



Including the “extra” terms in the time step calculation of the small-scale system, (4.13), results in some changes to the vectors and matrices in the equation, but no changes to how the equation itself is written. The inclusion of more coefficients $a_{j,k}$ in the small-scale system results in larger vectors $\mathbf{a}$, and a

larger matrix D_Λ . The matrix C_Λ is larger as well, due to additional rows on top and bottom. The matrix B_Λ has extra columns on the left and right. The matrix A_Λ is unchanged.

Expanding the number of coefficients $a_{j,k}$ has a cost. First, there are those extra terms to solve in the time step calculation. Also, the matrix S_Λ will have to be calculated over the expanded domain, Λ' . This would require including all the coefficients $b_{j,k}$ within Λ' . Thankfully, in our current, linearized, scheme, those terms are not included in the actual computation of the values for the time step, which is the most time consuming phase. They only come into play when setting up the operators themselves.

9. Results

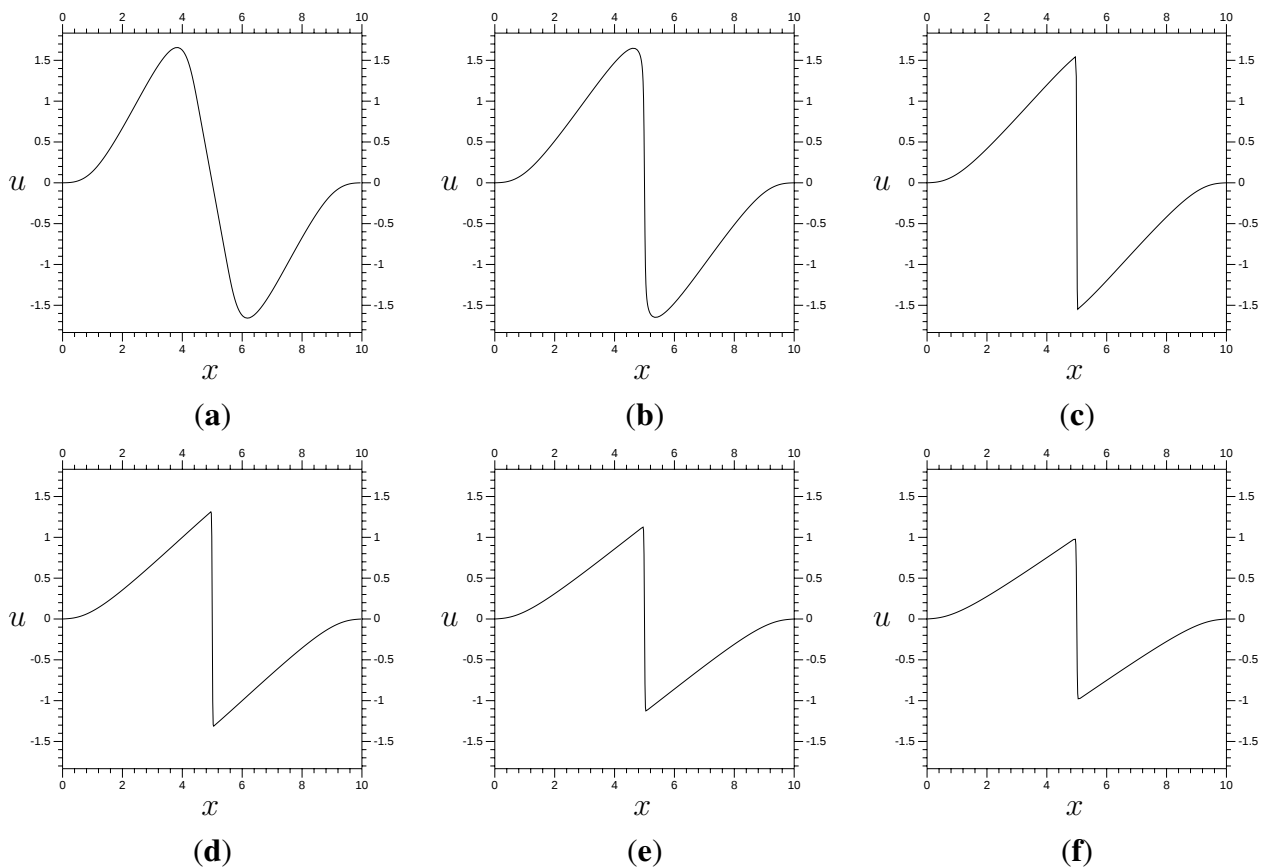
The most accurate (and stable) results use linear components derived from a full operator decomposition matrix M (see Section 4). Doing so results in the term $D_\Lambda - E_\Lambda$ from (8.9) being equal to a zero matrix, so there is no need for a matrix J (see Subsection 8.1) for the linear components of L . A matrix J_{NL} (again, see Subsection 8.1), is required for the non-linear derivative terms. Our first set of multi-scale results use V_{-4} on $\Omega = (0, 10)$ and V_{-6} on $\Lambda = (3.75, 6.25)$. They also use three “extra” terms and

$$\text{diag}(J_{NL}) = \left[0, 0, 0, 0, \frac{1}{2}, 1, 1, \dots, 1, 1, \frac{1}{2}, 0, 0, 0, 0 \right] \quad (9.1)$$

meaning that the non-linear effect on the “extra” terms in $\mathbf{a}_\Lambda^{T_m}$ is removed entirely. The results are in Figure 6, and the error (difference between the multi-scale, from Figure 6, and control results, from Figure 3) is in Figure 7. They show little difference from the full resolution V_{-6} results. Remember that these results use half as many terms, in total, and the method solves them in approximately half sized sections. As a result, the average time step for the V_{-6} system takes about 0.988 seconds (on a 2.5 GHz processor, standard for the remainder of the paper), while the average time step for the V_{-4}/V_{-6} multi-scale system takes about 0.0337 s. So, the multi-scale system gives results differing by 0.001, and takes about 3.4% as much time to compute.

Next, we need to confirm that the method can converge towards the results of the fine resolution system. To do so, we use a full domain V_{-6} and compare it to a scheme using V_{-5} with localized V_{-6} . An increase in the size of Λ and the number of “extra” terms should cause the resulting error to decrease. The multiplier J_{NL} is set to have a zero for each “extra” term, then one more zero followed by $\frac{1}{2}$, 1, and so on. The problem is the same, with $\nu = 0.01$, as is the size of the time step, $h = 0.01$. One interesting result is that the error in the interior of Λ becomes so small that it becomes insignificant next to the errors near $x = 0$ and $x = 10$. See Figure 8, and notice that the error on the boundary stays constant as Λ increases in size. The error near $x = 0$ and $x = 10$ is due to lack of resolution around the boundary. No matter how large we make Λ , or how close Λ gets to the boundary of Ω , that error stays constant (to within machine precision). We are mostly interested in the accuracy near the center of the domain, the modeling of the collision between the peaks. Also, the translation from the boundary to the center is trivial, meaning that the accuracy near the boundary has a minimal effect on the interior. So, we shall simply ignore $(0, 1.5)$ and $(8.5, 10)$ when calculating the error. In Figure 9 we have a set of plots of the errors (numerically approximated using a resolution of 2^6 elements per unit) of the different multi-scale systems, compared with the full domain V_{-6} system. These are calculated at $t = 2.0$, only on $(1.5, 8.5)$. Table 1 summarizes the Figure 9 results (notice that the “Terms” column relates to the horizontal axis

Figure 6. Results for Burgers' equation using the components from Section 8. These use V_{-4} with localized V_{-6} and three “extra” coefficients $a_{-3,k}$ around $\Lambda = (3.75, 6.25)$. It also uses $(J_{NL})_{i,i}$ values of 0, 0, 0, 0, $\frac{1}{2}$, 1, 1, etc. The time step size is $h = 0.01$. (a) $t = 0.5$; (b) $t = 1.0$; (c) $t = 1.5$; (d) $t = 2.0$; (e) $t = 2.5$; (f) $t = 3.0$.



for the plots on the left in Figure 9, and the “Time” column relates to those on the right). To put these results in perspective, the difference calculated in Subsection 7.3 between the V_{-6} control results and a much more accurate V_{-8} model has L_2 , L_1 and L_∞ differences of $8.0 \cdot 10^{-3}$, $2.1 \cdot 10^{-3}$ and $5.4 \cdot 10^{-2}$, respectively, on the subdomain $(1.5, 8.5)$. Basically, we have found solid evidence that the error resulting from our multi-scale method is trivial next to the error stemming from using a limited resolution of V_{-6} .

Figure 7. Results of a multi-scale system for Burgers' equation compared with a full resolution system. The error shown is the difference between the V_{-4}/V_{-6} results from Figure 6 and the full V_{-6} resolution results from Figure 3. (a) Error at $t = 1.0$; (b) Error at $t = 2.0$.

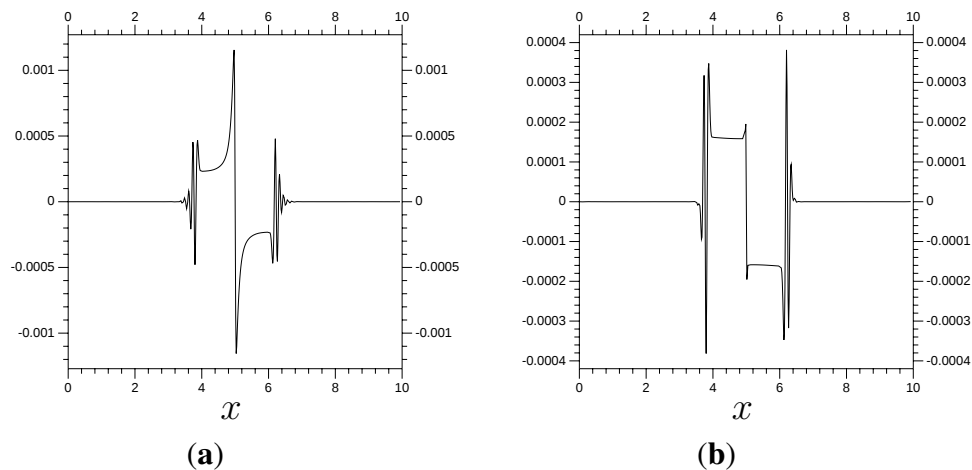


Figure 8. Error from results using V_{-5} with localized V_{-6} . The error is the difference between the multi-scale results and the V_{-6} control results, all calculated at $t = 2.0$. The multi-scale results differ in their Λ sub-domains. All use 4 “extra” terms per boundary of Λ . (a) $\Lambda = (3.75, 6.25)$; (b) $\Lambda = (3.50, 6.50)$; (c) $\Lambda = (2.50, 7.50)$; (d) $\Lambda = (1.50, 8.50)$.

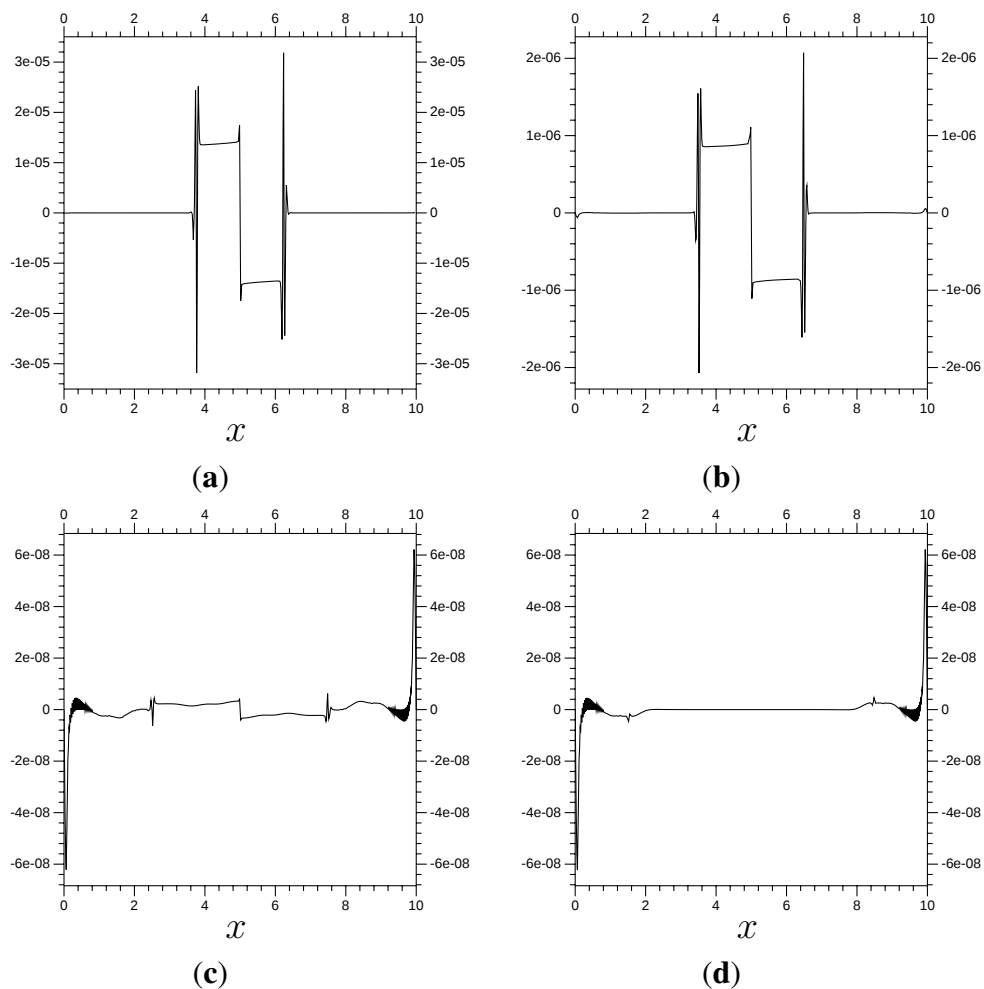


Figure 9. Error from multi-scale systems, compared with system size and computational time. The system size is the total number of coefficients $a_{j,k}$ and $b_{j,k}$ used in the systems (also seen in Table 1). The computational time is the average time taken to calculate a single time step, in seconds (all using the same computer). The vertical axis is the $\log_{10}$ of the error, the difference between the multi-scale results and the fine resolution control results on (1.5, 8.5). (a) $\log_{10}$ of the L^2 error; (b) $\log_{10}$ of the L^1 error; (c) $\log_{10}$ of the L^∞ error.

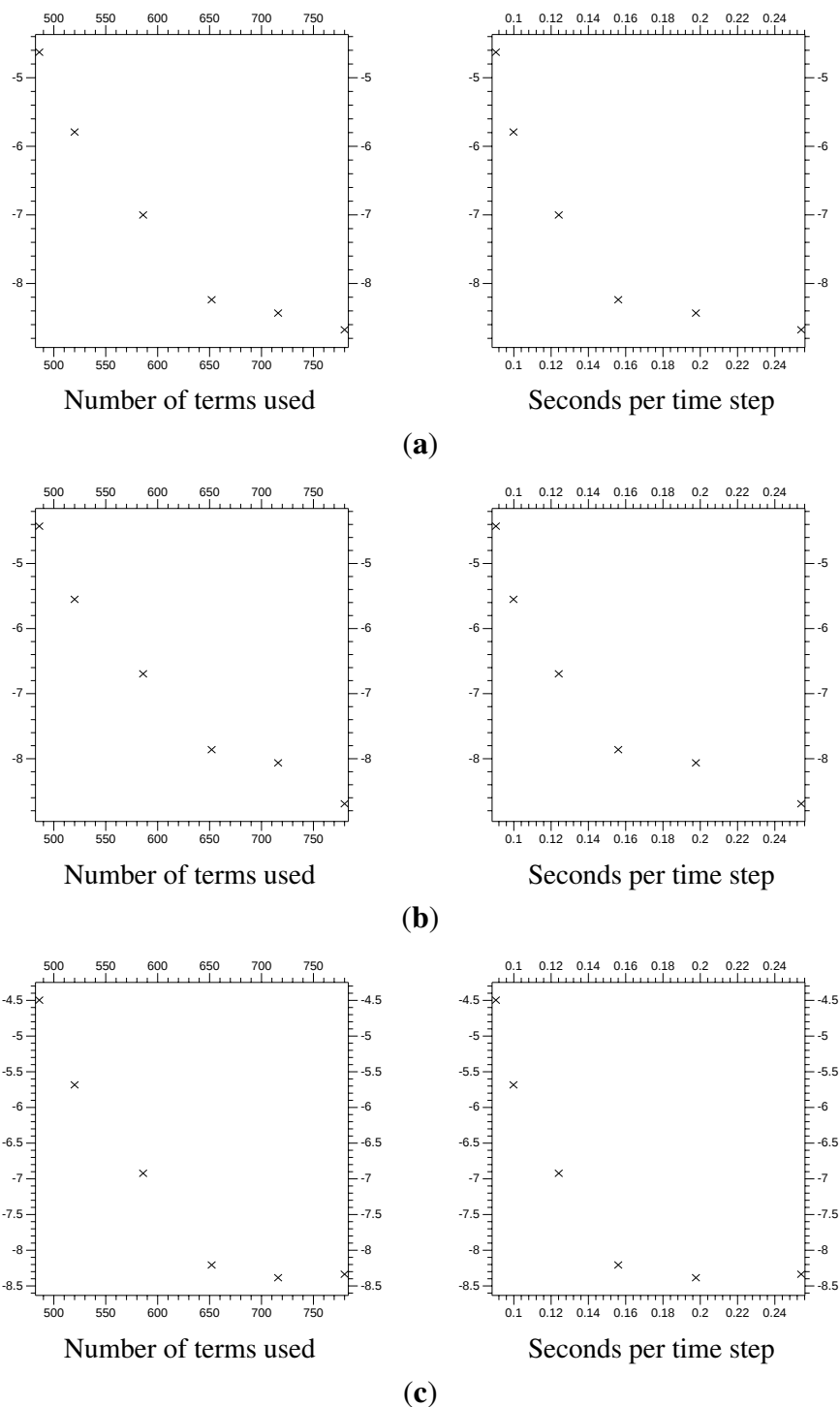


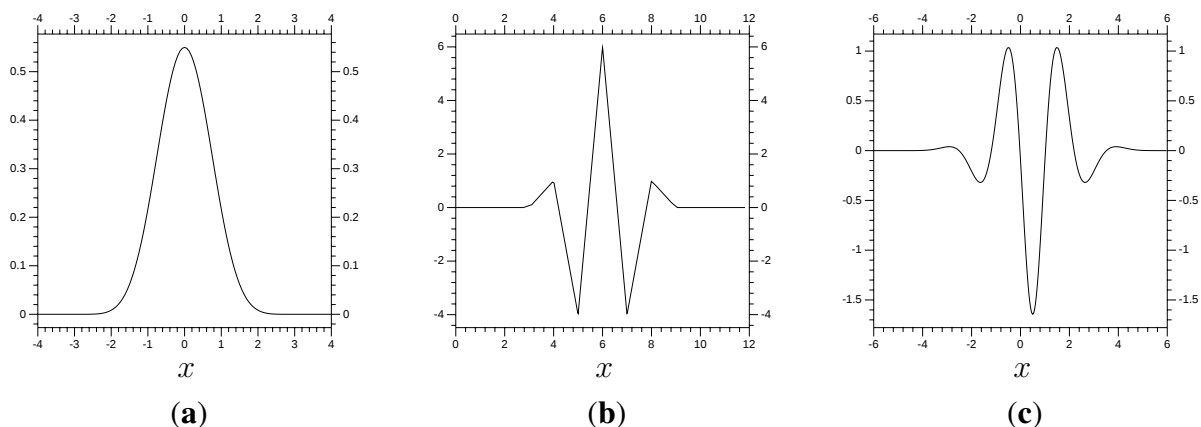
Table 1. Burgers’ equation error for multi-scale systems. By “Time” we mean the average computing time (in seconds) per time step. Note that the errors are only calculated in the region $[1.5, 8.5]$.

Λ	“extra”	Terms	Time	L_2 Error	L_1 Error	L_∞ Error
$[3.75, 6.25]$	4	128	0.0904	2.4×10^{-5}	4.6×10^{-5}	3.7×10^{-5}
$[3.50, 6.50]$	5	138	0.0998	1.6×10^{-6}	2.8×10^{-6}	2.1×10^{-6}
$[3.00, 7.00]$	6	156	0.1241	1.0×10^{-7}	2.1×10^{-7}	1.2×10^{-7}
$[2.50, 7.50]$	7	174	0.1560	5.8×10^{-9}	1.4×10^{-8}	6.2×10^{-9}
$[2.00, 8.00]$	7	190	0.1977	3.7×10^{-9}	8.6×10^{-9}	4.1×10^{-9}
$[1.50, 8.50]$	7	206	0.2541	2.1×10^{-9}	2.0×10^{-9}	4.6×10^{-9}
V_{-6} control results			0.9888			

10. A Further Test

Now we extend the method to a non-trivial problem, one that requires fine resolution in one of its two spatial dimensions. We shall use the same time stepping scheme, the same multi-scale method, and the same modifications (the “extra” terms and matrix J). However, the differential equation has a fourth spatial derivative, and so is best approached with a higher order scaling function and wavelet pair, seen in Figure 10.

Figure 10. The spline function $B_5 = \phi$, our scaling function. Next, $\frac{d^4}{dx^4} \phi$ and our wavelet, ψ . (a) $\phi(x)$; (b) $\frac{d^4}{dx^4} \phi(x)$; (c) $\psi(x)$.



10.1. The Rossby Wave Problem

The problem is time-dependent on an x, y , domain, with reference lengths L_x and L_y , respectively. There are periodic boundary conditions in the x -direction, and time-independent Dirichlet boundary conditions in the y -direction. First, we shall nondimensionalize the problem, converting the domain to

$x \in (0, 2\pi)$ and $y \in [0, 10]$. The primary effect of this is that the Laplacian operator $\left(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2}\right)$ will be re-scaled to

$$\Delta = \left(\frac{\partial^2}{\partial y^2} + \delta \frac{\partial^2}{\partial x^2}\right), \quad \text{with} \quad \delta = \left(\frac{L_y}{L_x}\right)^2 \quad (10.1)$$

For further details on the problem, see [19].

The problem is based on the nondimensionalized barotropic vorticity equation

$$\Delta \Psi_t + \Psi_x \Delta \Psi_y - \Psi_y \Delta \Psi_x + \beta \Psi_x - \nu \Delta^2 \Psi = 0 \quad (10.2)$$

(see ([20] [p. 637]) for details) with the (real) function $\Psi(x, y, t)$ the stream function on the domain (not a wavelet). The velocity component in the x -direction is equal to $-\Psi_y(x, y, t)$ and the y component is equal to $\Psi_x(x, y, t)$. The variable x represents longitude (east-west) and y the latitude (north-south). The coefficient β relates to planetary rotation and ν is the viscosity coefficient.

Next we decompose the stream function Ψ into two components,

$$\Psi(x, y, t) = \bar{\Psi}(y) + \epsilon P(x, y, t) \quad (10.3)$$

with ϵ small relative to Ψ and $\bar{\Psi}$. The function $\bar{\Psi}(y)$ is the stream function of the basic flow. We are assuming that the basic flow is a shear flow, with velocity only in the x -direction, meaning that $\bar{\Psi}$ is a function of y only. We set $\bar{\Psi}'(y) = -\bar{u}(y)$, a known function. The Rossby waves are given by $P(x, y, t)$, the stream function representing small perturbations of the basic flow, with the matching vorticity perturbation $Z = \Delta P$.

Substituting (10.3) into (10.2), we obtain

$$\epsilon \Delta P_t + \epsilon P_x (-\bar{u}''(y) + \epsilon \Delta P_y) - (-\bar{u}(y) + \epsilon P_y) \epsilon \Delta P_x + \epsilon \beta P_x - \nu (\bar{u}^{(3)}(y) + \epsilon \Delta^2 P) = 0 \quad (10.4)$$

which becomes

$$Z_t + \bar{u}(y) Z_x + (\beta - \bar{u}''(y)) P_x - \epsilon (P_y Z_x - P_x Z_y) - \nu \Delta Z + \frac{\nu}{\epsilon} \bar{u}^{(3)}(y) = 0 \quad (10.5)$$

(with $\bar{u}^{(3)}$ being the third derivative of $\bar{u}$). We shall ignore the last y dependent term, since $\bar{u}^{(3)}$ is small, and even when $\nu \neq 0$ we shall keep ν much smaller than ϵ , so (10.5) becomes

$$Z_t = -\bar{u}(y) Z_x - (\beta - \bar{u}''(y)) P_x + \nu \Delta Z + \epsilon (P_y Z_x - P_x Z_y) \quad (10.6)$$

Our examples will use $\bar{u}(y) = \tanh(y - 5)$, meaning the basic flow will be very nearly constant in the regions near $y = 10$ and $y = 0$, (at $\tanh(y) \approx 1$ and $\tanh(y) \approx -1$, respectively). Second, the basic flow velocity will be zero at $y = 5$. As we shall see, these properties of $\bar{u}$ have interesting implications for P .

The boundary conditions are $P(x, 10, t) = \cos(2x)$ for $t > 0$ and $P(x, 0, t) = 0$ (recall that the boundary conditions are periodic in x). We shall have those boundary conditions phased in gradually (over $t \in [0, 5]$ or so) to smooth out the discontinuity, starting at zero and smoothly increasing to $\cos(2x)$. Notice that the boundary conditions are all composed of $\cos(2x)$, so they can be written in terms of $e^{\pm 2ix}$. If we use the linear form of the problem (set $\epsilon = 0$), then P (and Z) can be written using only

functions $e^{\pm 2ix}$ in the x -direction (each multiplied by a function of y and t). The initial condition is simply $P(x, y, 0) = Z(x, y, 0) = 0$.

Now we go to the simplest case, set $\nu = 0$ along with ϵ . If we assume that the problem has a steady state, we can find that it will have the form

$$P(x, y) = \Phi_2(y) e^{i2x} + \Phi_{-2}(y) e^{-i2x}, \frac{d^2}{dy^2} \Phi_\kappa(y) + \left(-\delta\kappa^2 + \frac{\beta - \bar{u}''(y)}{\bar{u}(y)} \right) \Phi_\kappa(y) = 0 \quad (10.7)$$

In the area immediately around $y = 10$, $\bar{u}(y)$ is approximately one and $\bar{u}''(y)$ is approximately zero. The problem becomes

$$\left(\frac{d^2}{dy^2} + (\beta - \delta\kappa^2) \right) \Phi = 0 \quad (10.8)$$

with a solution of the form $e^{\pm il(y-10)}$, $l = \sqrt{\beta - \delta\kappa^2}$. As a result, recalling that the boundary condition at $y = 10$ is $P = \cos(2x)$, the steady state is

$$P(x, y) = \cos(2x + \rho(y - 10)) \quad (10.9)$$

approximately, near $y = 10$. The more general observation is that the problem in (10.7) has a singularity when $\bar{u}(y) = 0$, so at $y = 5$. The singularity at $y = 5$ means we require fine resolution at that location in order to accurately model the problem. The effect of the critical layer, the region around $y = 5$, on the problem is well known. The function P , which has the form of (10.9) near $y = 10$, will propagate downwards, curve slightly, then stop around $y = 5$ (see Figure 11B). This behavior is consistent with that found in the literature: Figure 11B is consistent with all the P (equivalent) plots in [19], and virtually identical to ([19] [Figure 2a]). The behavior of the perturbation velocity $Z = \Delta P$ is also fairly well known. Figure 11 contains plots of Z at different times. The general Z shape expected for the linear problem is the waves around $y = 5$, seen in Figure 11. Notice that the amplitude of Z near $y = 5$ is much higher than that anywhere else, and that the waves in the center get more tightly concentrated as t increases. As we shall see, this results in a greater requirement for fine resolution in the center as t increases, which is what we would expect considering the singularity at $y = 5$. The shape of our Z plots in Figure 11 matches well with ([19] [Figure 18a]) and ([21] [Figure 4]), even though our results use different domains and boundary conditions.

Apart from P and Z , there is one more quantity we shall use to check the accuracy of our results. We shall be interested in the momentum of the system, specifically the averaged x directional momentum flux of the region $y \in (2.5, 7.5)$. Information on momentum as it relates to fluid dynamics can be found in any text on the subject, such as ([20] [p. 88]). For our purposes, it suffices to say we calculate the momentum going in and out of the region via the integral

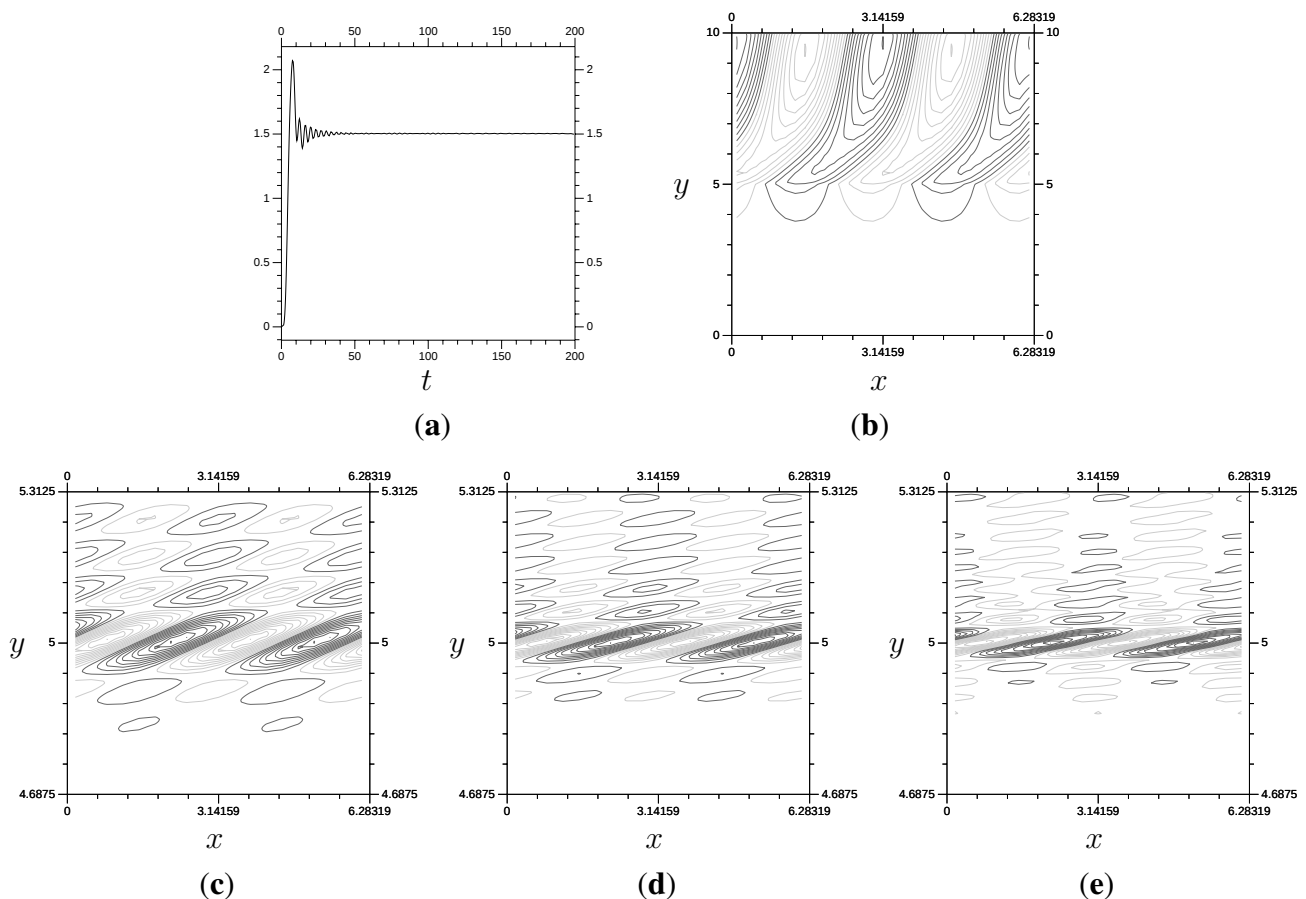
$$-\frac{1}{2\pi} \int_0^{2\pi} P_x(x, y, t) P_y(x, y, t) dx \quad (10.10)$$

at $y = 2.5$ and 7.5 . The change in the x -averaged momentum flux across the region $[2.5, 7.5]$ is equal to

$$F(t) = \frac{1}{2\pi} \int_0^{2\pi} P_x(x, 7.5, t) P_y(x, 7.5, t) dx - \frac{1}{2\pi} \int_0^{2\pi} P_x(x, 2.5, t) P_y(x, 2.5, t) dx \quad (10.11)$$

although the first integral is approximately zero since $P \approx 0$ for $y = 2.5$. In this particular system, there are some very clear expectations. First, the linear problem is expected to result in a positive flux at the

Figure 11. (Top) The flux (a), steady state for P (b). (Bottom) the progression of Z at $t = 50, 75, 100$ in the x - y plane, from left to right. These use $\delta = 0.2$, $\beta = 1$ and $\nu = \epsilon = 0$. Notice that Z increases in amplitude and decreases in wavelength as time increases. (a) The Flux; (b) P ; (c) Z , $t = 50$; (d) Z , $t = 75$; (e) Z , $t = 100$.



beginning, so momentum flowing into the critical layer. In ([22] [Figure 5, line 1]) and ([19] [Figure 3]), the flux for the linear problem can be seen reaching its maximum early, then lowering and stabilizing at a positive value. This is the expected behavior of the linear problem, see Figure 11a. When the non-linear interactions are included, the critical layer is expected to reflect some of the momentum. The flux, as seen in ([22] [Figure 5, lines 2,3]) and ([19] [Figure 3b]), is expected to reach its maximum value early, then reduce down to zero and begin oscillating around zero. Later, we shall test our numerically derived results against these expectations.

10.2. Linear Results

The key observation for the linear version of the problem (when $\epsilon = 0$) is the steady state expected by the theory, briefly discussed in Subsection 10.1. As t increases, the linear solution eventually converges to the steady state given in Figure 11b (as long as we are using our usual $\bar{u}(y)$). However, as shown in Figure 11, this results in Z requiring very fine resolution around $y = 5$. As a result, what we actually see in numerical computation is the perturbation stream function P reaching the steady state (or close to it), holding, then, after some time, becoming completely inaccurate. Generally, without viscosity, our

computed values of P are accurate up to approximately $t = 25 \cdot 2^{n-3}$ using resolution V_{-n} , as shown in Figures 12 and 13.

Figure 12. The results from a linear V_{-6} system. Notice that $F(t)$ (the x averaged momentum flux) remains stable until $t = 200$. These results use a “switch-on” function for the boundary condition values equal to $\min\{\frac{t}{5}, 1\}$, $\beta = 1$, $\delta = 0.2$, and no viscosity. (a) $F(t)$; (b) P , $t = 200$; (c) Z , $t = 200$.

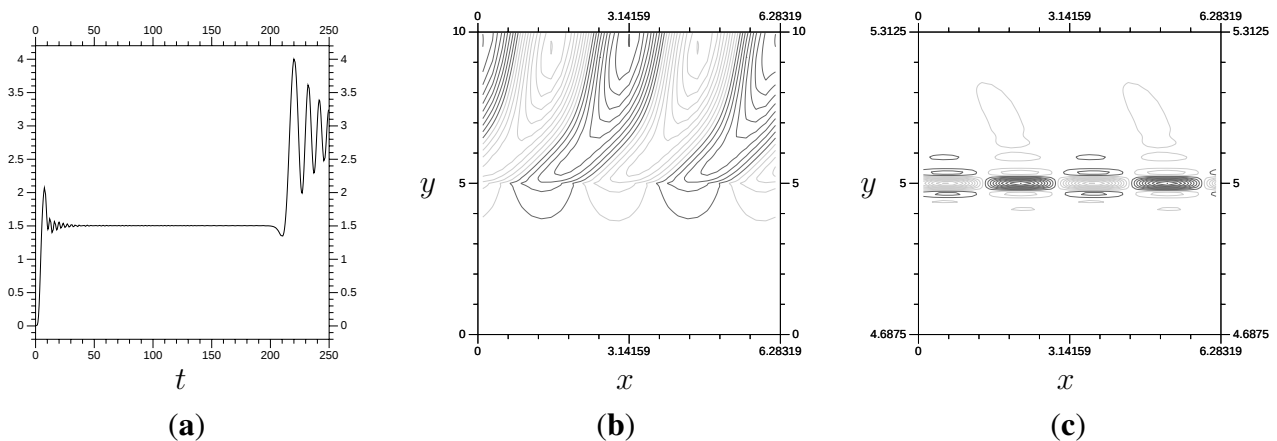
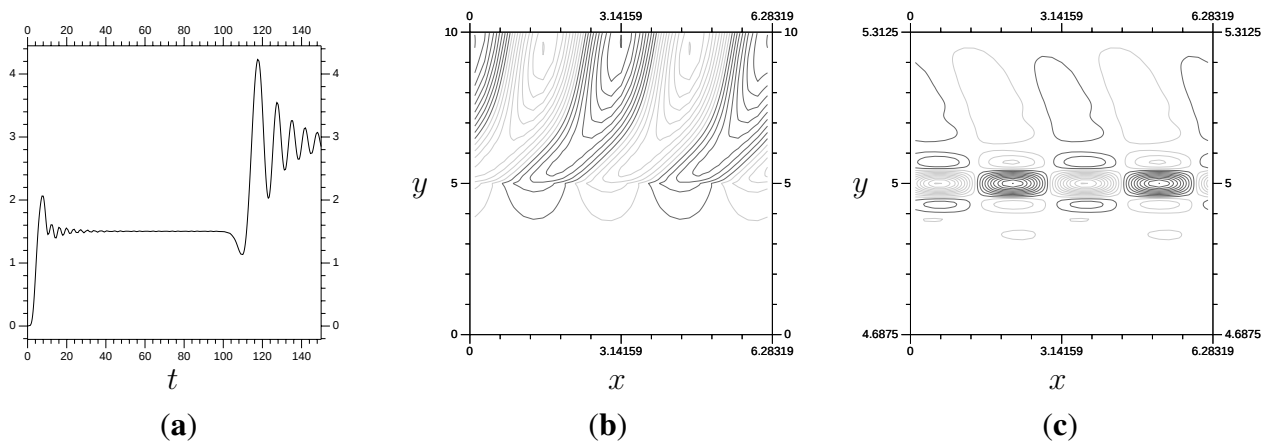
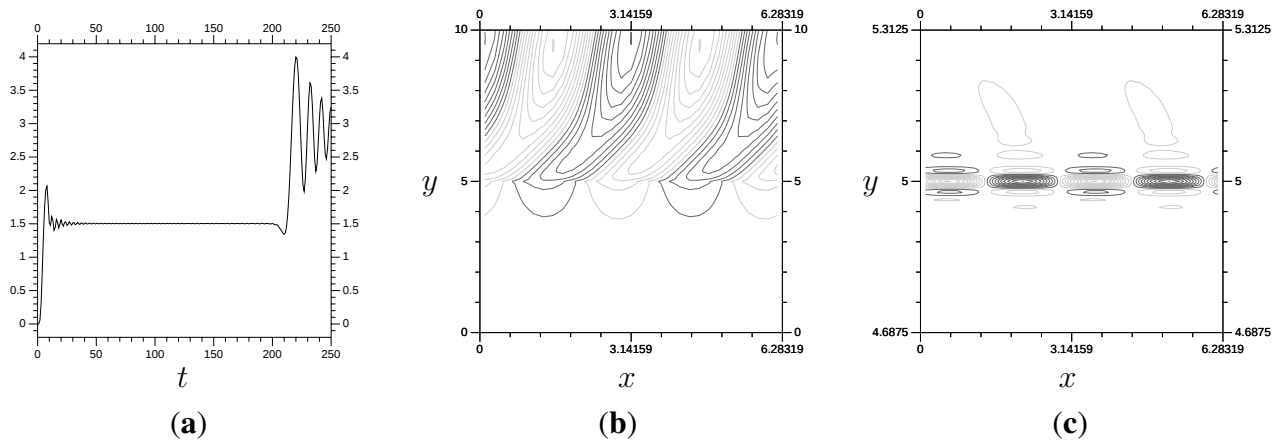


Figure 13. The results from a linear V_{-5} system. Notice that $F(t)$ remains stable until $t = 100$, where Z is following the same pattern seen in Figure 12c. These results are based around the same problem as those in Figure 12, just with a different resolution. (a) $F(t)$; (b) P , $t = 100$; (c) Z , $t = 100$.



As seen in the Z plots, the key information is around $y = 5$, so that is the location for the small-scale system. The real test is whether the multi-scale results will be accurate (stable, *etc.*) over the same time frame. The V_{-6} single system results are accurate to approximately $t = 200$ (Figure 12). In Figure 14, we have results using V_{-4} everywhere and V_{-6} on $\Lambda = (3.0, 6.5)$, showing an accurate solution of the problem up to $t = 200$, so the multi-scale method works for the linear problem. Note that the multi-scale arrangement has $159 + 165$ terms per Fourier mode, and the two systems are solved separately. The full V_{-6} system has 639 elements per Fourier mode, and the system has to be solved all at once.

Figure 14. Results from a V_{-4} with localized V_{-6} double system. The small-scale is over $\Lambda = (3.0, 6.5)$, and 3 “extra” terms are used over both interfaces of Λ . There is a little instability in the flux around $t = 200$, but this arrangement is very close to the full V_6 control system, found in Figure 12. (a) $F(t)$; (b) $P, t = 200$; (c) $Z, t = 200$.



10.3. Non-Linear Results

The actual benefits of the multi-scale system exist primarily for non-linear problems, so we shall switch to the non-linear version, with $\epsilon \neq 0$. The linearization scheme used on Burgers' equation (see (7.2)) will be used again.

There is one more, significant, issue: We shall need many more Fourier coefficients to properly express the x direction when $\epsilon \neq 0$. With the boundary conditions just multiples of $\cos(k_0 x)$, and with initial conditions of zero, the linear problem's solution can be written in terms of $e^{\pm k_0 x}$ in the x direction. If we have $\epsilon \neq 0$ with the same boundary and initial conditions, then the problem will require $e^{\pm k_0 n x}$, $n \in \mathbb{Z}$, in the x direction. We use $k_0 = 2$, so the even Fourier modes will be relevant. Obviously, we have to restrict the number of Fourier modes we calculate to a finite number, and use a pseudo Fourier scheme to approximate $P_x Z_y$ and $P_y Z_x$. The functions P and Z are real, so modeling the Fourier modes $\kappa = -2K, -2K + 1, \dots, 2K - 1, 2K$ (for $K \in \mathbb{N}$) requires K complex values and one real value to keep track of, which are effectively $2K + 1$ real values.

So, we have to create and solve a new linear system at every step, and the systems themselves will be significantly larger. As a result, we shall keep the y resolution down to a manageable V_{-4} , or 16 elements per interval of length one, so 159 total on $[0, 10]$. Combined with 11 elements for the x direction and that makes 1749 real coefficients in total. The boundary conditions are the same, and $\beta = 1$, $\delta = 0.2$ as before. We shall include some viscosity, $\nu = 0.0001$, to stabilize the system and make the V_{-4} resolution plausible. We use $\epsilon = 0.01$, and the size of the time step is $h = 0.05$.

Our multi-scale results use the method that was applied to Burgers' equation in Section 8. We use three “extra” terms and a J multiplier with

$$\text{Diagonal of } J = \left[0, 0, 0, 0, \frac{1}{2}, 1, 1, \dots, 1, 1, \frac{1}{2}, 0, 0, 0, 0 \right]$$

The linear components of the calculation are derived the same way as in Section 8 as well, from a system using the finest resolution over all Ω .

As discussed at the end of Subsection 10.1, the flux for the non-linear problem is expected to reach its maximum early, then reduce down to zero and begin oscillating around zero. As for P , it is expected to develop waves similar to the linear problem, at least initially. As t increases, we can expect the tip of the waves in P , the parts near $y = 5$, to break from the rest and create separate, somewhat circular, shapes. This behavior is shown nicely in ([19] [Figures 2b and 14b]). Other than the expectation of greater complexity, we shall say little of Z . We shall, however, use Z to check the accuracy of the multi-scale results.

When $\epsilon \neq 0$, the flux is expected to quickly reach the steady state of the linear system, then slowly reduce to zero and begin oscillating around zero. Take a look at Figure 15a–f. The value $\epsilon = 0.01$ is not high enough for the flux to reach zero during $t \in [0, 50]$, so instead we see a reduction towards zero for the V_{-4} model. We set V_{-4} as our top resolution, the results we want to duplicate efficiently with the multi-scale method. Our large-scale resolution is V_{-3} .

Now we discuss how this particular Rossby wave problem responds to the multi-scale method. The problem does seem to require a larger domain for the small-scale system, which cuts into the efficiency. The V_{-4} system took an average of about 9.55 s per time step. See Figure 16 for our multi-scale results with the smallest small-scale domain, $\Lambda = (1.5, 6.5)$. The $\Lambda = (3.5, 8.5)$ double system requires approximately 3.51 s per time step, a significant improvement on the V_{-4} system. However, the results are not quite as close to the V_{-4} system as we would like. Including more elements above $y = 8.5$ improves things. In fact, the results are visually indistinguishable to the control results in Figure 15. However, the $\Lambda = (3.5, 9.0)$ model requires 3.90 s per time step, and $\Lambda = (3.5, 9.5)$ requires 4.48 s per time step.

The multi-scale results were shown to be able to duplicate the V_{-4} results, using V_{-3} with a localized V_{-4} resolution. There was also a significant reduction in the effort, with the time step taking half as much time to compute. The resolution used was restricted to a coarse V_{-4} to keep computing requirements reasonable. The increase from V_{-3} to V_{-4} means that the small-scale system has approximately half of its coefficients be in V_{-3} , so coefficients that are also in the large-scale system. This is the least efficient form of the method, with only a single W_j space worth of additional resolution in the small-scale system, and it still proved beneficial with the non-linear Rossby wave problem. The V_{-4} control results took about 9.25 s per time step, while the, visually identical, multi-scale results with $\Lambda = (3.5, 9.5)$ took 4.48 s per time step. So, we have accurate results with less than half the computational expense.

Figure 15. Plots for the $\epsilon = 0.01$ problem using V_{-4} . (Top) Flux; (Middle) P ; (Bottom) Z .
(a) P , $t = 10$; (b) P , $t = 30$; (c) P , $t = 50$; (d) Z , $t = 10$; (e) Z , $t = 30$; (f) Z , $t = 50$.

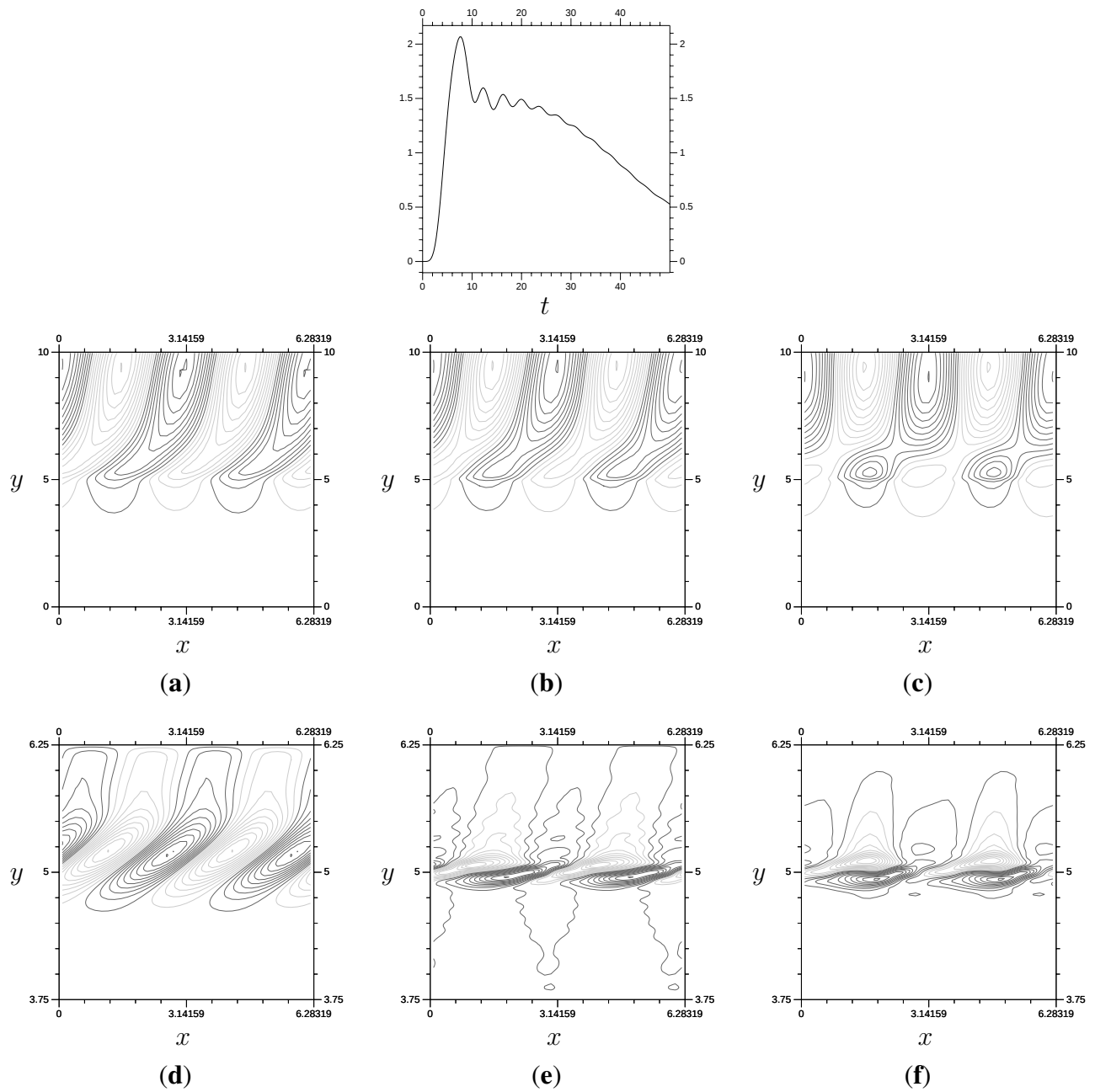
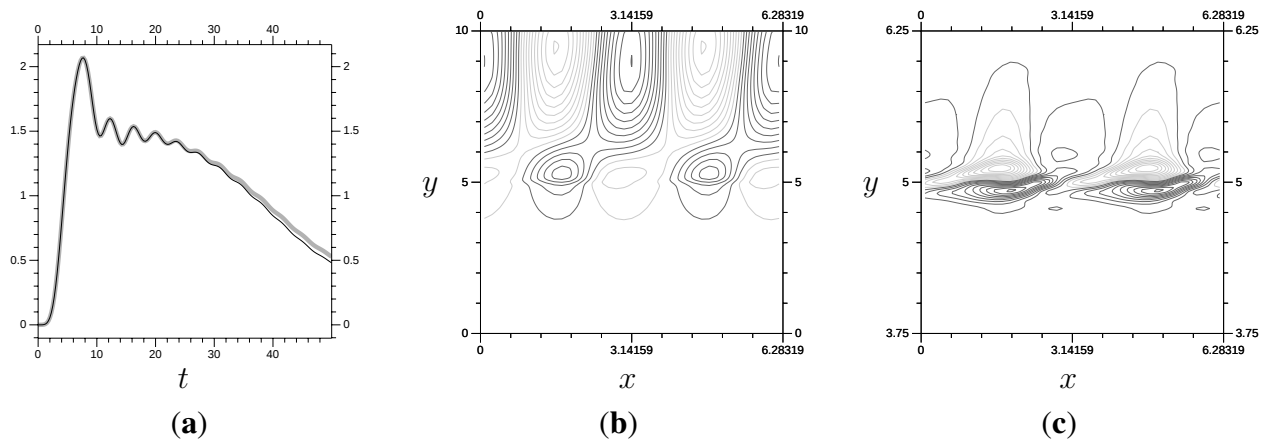


Figure 16. Plots for $V_{-3} : V_{-4}$ multi-scale models of the $\epsilon = 0.01$ Rossby wave problem. All use $\Lambda = (1.5, 6.5)$. The flux is plotted against that for the V_{-4} control results (the lighter, thicker line). The results using the larger $\Lambda = (3.5, 9.0)$ and $\Lambda = (3.5, 9.5)$ are omitted, as they are visually identical to the full V_{-4} control results from Figure 15. (a) $F(t)$; (b) P at $t = 50$; (c) Z at $t = 50$.



10.4. Convergence

The convergence of the Rossby wave problem is slightly restricted by the boundary conditions. Recall that the initial conditions are zero, while the $y = 10$ boundary condition is equal to $\cos(2x)$. Since Λ does not reach $y = 10$, the boundary conditions are modeled at the large-scale resolution. The lack of resolution for the boundary conditions results in error that cannot be removed without significant modifications to the method (including a second small-scale system at $y = 10$ or incorporating the non-zero boundary conditions into Λ). As in the convergence related discussion in Section 9, we look at the effect of different Λ sizes and “extra” terms on the accuracy of multi-scale results. The control results are a V_{-6} resolution single system. The test systems use V_{-4} or V_{-5} over Ω and V_{-6} on Λ . Plots of the $\log_{10}$ transformed error can be found in Figure 17. The summarized data is in Table 2.

Now we take a look at the convergence from the non-linear results in Section 10.3. The multi-scale results are calculated at V_{-3} with V_{-4} on differing small-scale domains Λ . They are compared with the full domain V_{-4} control results. The usual plots of the error against the number of $a_{j,k}$ and $b_{j,k}$ coefficients used, and the time required, for the multi-scale system can be found in Figure 18. The summarized data can be found in Table 3.

Figure 17. The error at $t = 200$ of the function Z from the linear problem. The vertical axis is the $\log_{10}$ of the error (the difference between the multi-scale system results and the V_{-6} control results). The horizontal axis is the total number of $a_{j,k}$ and $b_{j,k}$ coefficients used for the y directional decomposition. (a) $\log_{10}$ of the L^2 error for V_{-4} with V_{-6} on Λ ; (b) $\log_{10}$ of the L^2 error for V_{-5} with V_{-6} on Λ ; (c) $\log_{10}$ of the L^1 error for V_{-4} with V_{-6} on Λ ; (d) $\log_{10}$ of the L^1 error for V_{-5} with V_{-6} on Λ ; (e) $\log_{10}$ of the L^∞ error for V_{-4} with V_{-6} on Λ ; (f) $\log_{10}$ of the L^∞ error for V_{-5} with V_{-6} on Λ .

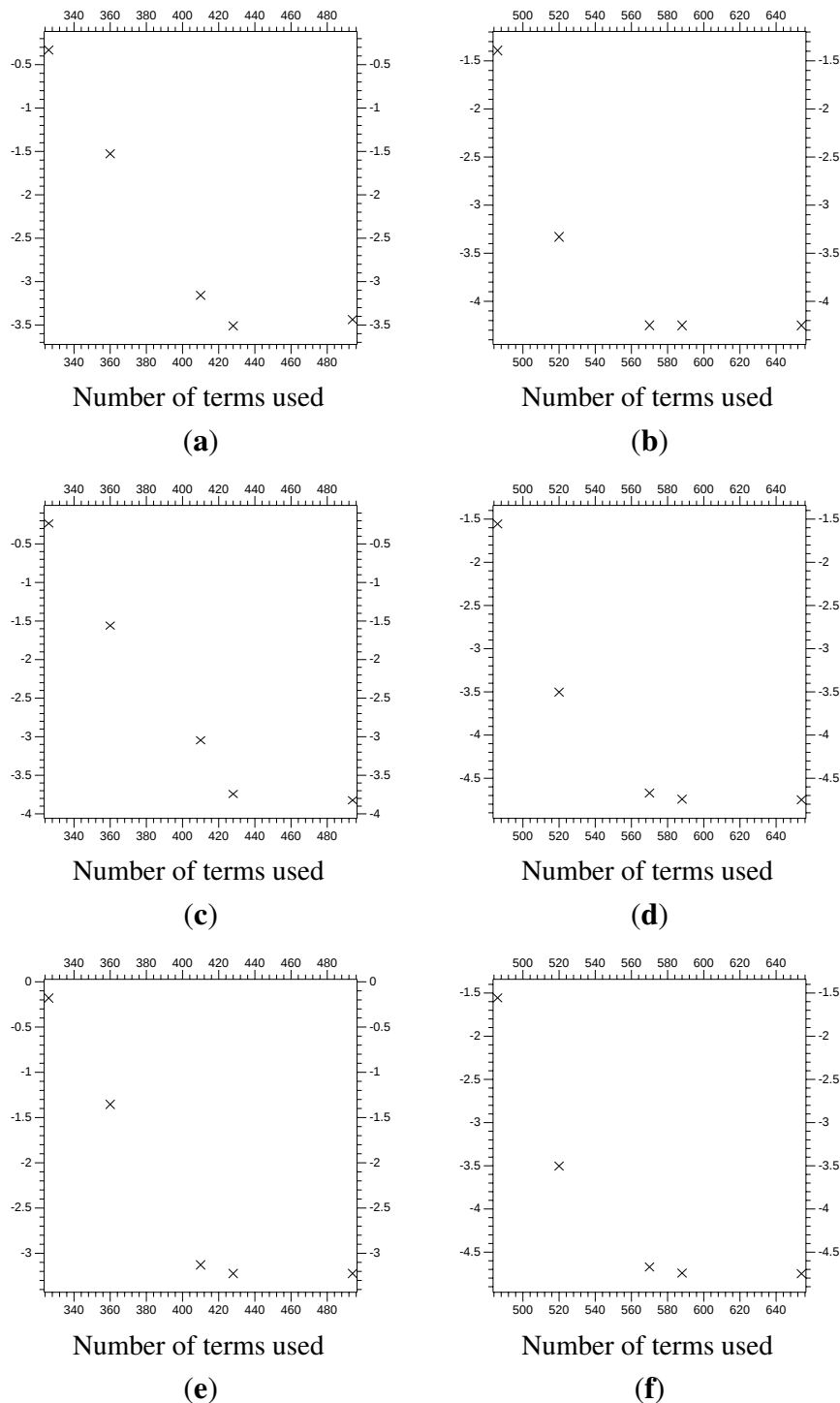


Table 2. Error from multi-scale linear Rossby wave results.

	Λ	“extra”	Terms	L^2 Error	L^1 Error	L^∞ Error
V_{-4}/V_{-6}	[3.75,6.25]	3	326	4.66×10^{-1}	5.82×10^{-1}	6.61×10^{-1}
V_{-4}/V_{-6}	[3.50,6.50]	4	360	2.97×10^{-2}	2.76×10^{-2}	4.42×10^{-2}
V_{-4}/V_{-6}	[3.25,7.00]	5	410	6.95×10^{-4}	7.99×10^{-4}	7.45×10^{-4}
V_{-4}/V_{-6}	[3.00,7.00]	6	428	3.09×10^{-4}	1.81×10^{-4}	5.98×10^{-4}
V_{-4}/V_{-6}	[2.50,7.50]	7	494	3.65×10^{-4}	1.50×10^{-4}	5.98×10^{-4}
V_{-5}/V_{-6}	[3.75,6.25]	3	486	4.05×10^{-2}	2.78×10^{-2}	7.19×10^{-2}
V_{-5}/V_{-6}	[3.50,6.50]	4	520	4.68×10^{-4}	3.14×10^{-4}	9.46×10^{-4}
V_{-5}/V_{-6}	[3.25,7.00]	5	570	5.64×10^{-5}	2.13×10^{-5}	1.65×10^{-4}
V_{-5}/V_{-6}	[3.00,7.00]	6	588	5.62×10^{-5}	1.81×10^{-5}	1.65×10^{-4}
V_{-5}/V_{-6}	[2.50,7.50]	7	654	5.62×10^{-5}	1.78×10^{-5}	1.65×10^{-4}

Figure 18. The error at $t = 50$ of the function Z from the non-linear ($\epsilon = 0.01$) problem. The vertical axis is the $\log_{10}$ of the error, the difference between the $V_{-3} : V_{-4}$ multi-scale results and the V_{-4} control results (see Figure 15). The horizontal axis is the total number of $a_{j,k}$ and $b_{j,k}$ coefficients used for the y directional decomposition. (a) $\log_{10}$ of the L^2 error; (b) $\log_{10}$ of the L^1 error; (c) $\log_{10}$ of the L^∞ error.

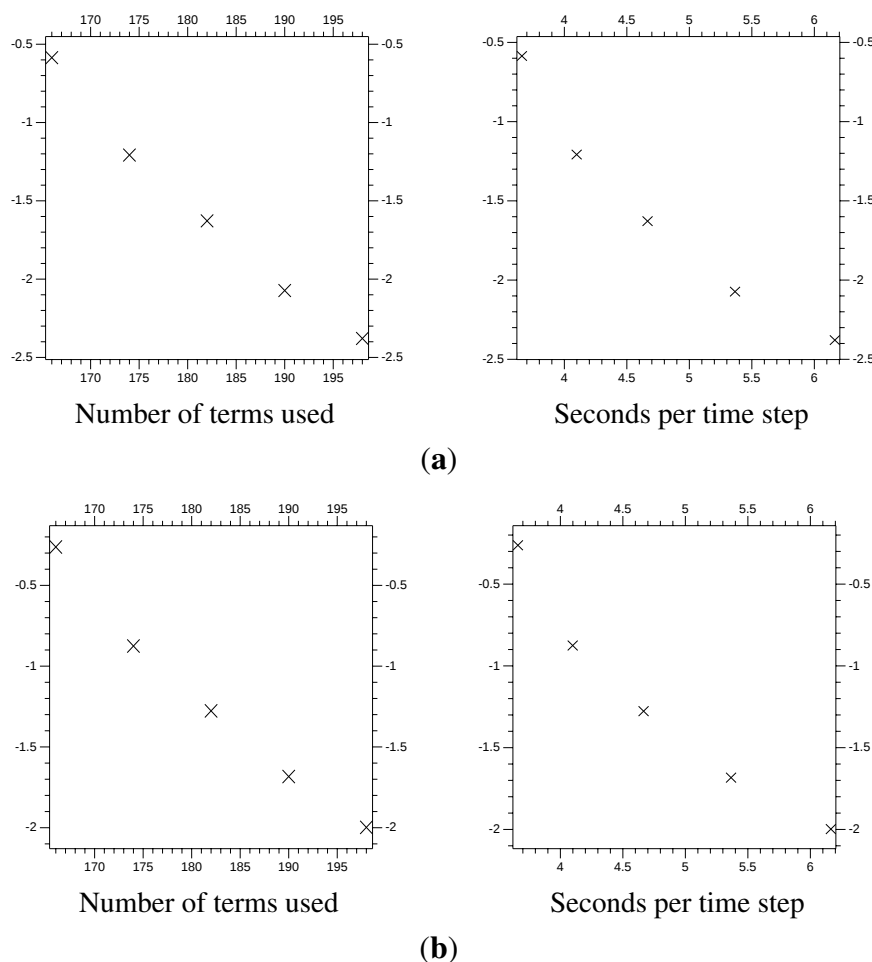


Figure 18. Cont.

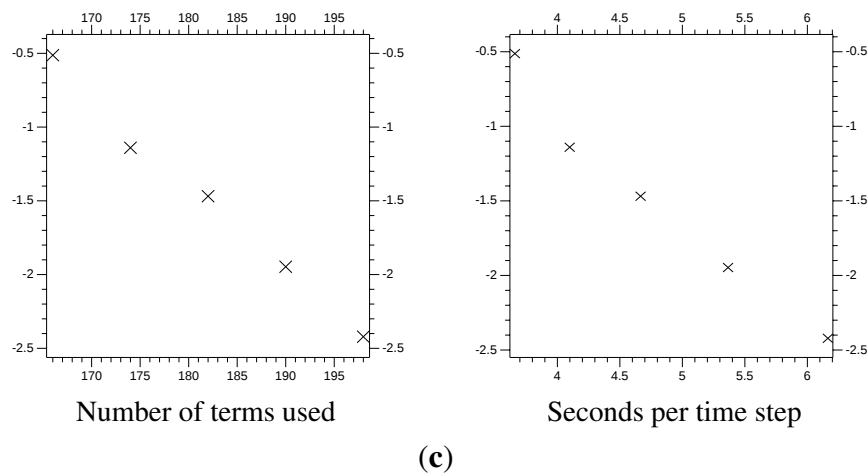


Table 3. Error from non-linear multi-scale Rossby wave results. All but the control use V_{-3} and V_{-4} .

Λ	“extra”	Terms	Time	L^2 Error	L^1 Error	L^∞ Error
[3.5, 8.5]	4	168	3.66	2.60×10^{-1}	5.46×10^{-1}	3.07×10^{-1}
[3.5, 9.0]	4	176	4.10	6.20×10^{-2}	1.33×10^{-1}	7.24×10^{-2}
[3.5, 9.5]	4	184	4.67	2.35×10^{-2}	5.29×10^{-2}	3.40×10^{-2}
[3.0, 9.5]	4	192	5.27	8.46×10^{-3}	2.07×10^{-2}	1.13×10^{-2}
[2.5, 9.5]	4	200	6.16	4.18×10^{-3}	1.01×10^{-2}	3.79×10^{-3}
V_{-4} Control			9.25			

11. Conclusions

After some wavelet based preliminaries, we described a new method for partial differential equations at multiple scales, in Section 3. The small-scale resolution is restricted to a sub-domain Λ of the full domain Ω . The method divides the implicit system created by the time-discretization scheme into two smaller systems: One at a coarse resolution over the entire domain Ω and the other at fine resolution on the subset $\Lambda \subset \Omega$. This dividing of the system will typically result in significant computational savings.

Next we set up a test problem based on Burgers’ equation. The initial condition, Figure 3a, results in two peaks that approach each other and collide. The resulting collision requires a fine resolution to be properly expressed. A resolution of V_{-6} is required for stable and accurate results once the peaks collide (at approximately $t = 1$). A resolution of V_{-4} causes instability (coefficients in the range of 10^{20}). However, since the collision is at $x = 5$, the fine resolution is only required in a domain in the center. The calculations using a single resolution of V_{-6} over the entire domain Ω take 0.988 s per time step. The resolution of V_{-6} would be restricted to a sub-domain Λ in the center of Ω , where the peaks collide. The method required a few improvements beyond the basic setup, outlined in Section 8.

We first tested the resolutions V_{-4} on Ω and V_{-6} on $\Lambda = (3.75, 6.25)$, yielding a close approximation to the V_{-6} control results. The largest L_∞ difference between them went briefly higher than 0.001, and

otherwise was below 0.0005 (see Figure 7). These differences were found to be an order of magnitude smaller than the error stemming from the limited resolution (V_{-6}) and time step ($h = 0.01$) common to both the multi-scale and control results (see Subsection 7.3 and Figure 4). Furthermore, the multi-scale results required 0.0337 s per time step, while the control results required approximately one second per time step. So, the multi-scale results were not only just stable and much less expensive to calculate, but also accurate.

The next step was to check for convergence towards the control results, computed using a resolution of V_{-6} on the full domain. For this test we used V_{-5} on Ω and V_{-6} on several different subsets Λ of Ω . We had to control for the error near the boundaries of Ω (far from the Λ sub-domains), which was, in fact, identical for all the different Λ sub-domains. Using the L_2 , L_1 and L_∞ norms, increasing the size of the small-scale system resulted in a smaller difference between the test systems and the control results. The eventual error was approaching machine precision.

Next, further confirmation was sought using a non-trivial problem. The Rossby wave problem is two dimensional, on $(x, y) \in (0, 2\pi) \times (0, 10)$, with small-scale interactions found towards the center of the domain in the y direction. First we tested the linear form of the problem, where fine resolution near $y = 5$ is needed to accurately represent the correct solution over time. Without viscosity, the numerical results would fail to maintain consistency beyond a certain value of t , with a finer resolution allowing this value of t to be larger. Using the linear problem with no viscosity, a wavelet decomposition with a resolution of V_{-4} will show the correct result over the time frame $t \in [0, 50]$, V_{-5} over $t \in [0, 100]$, and V_{-6} over $t \in [0, 200]$. The multi-scale method was used with V_{-6} on $y \in \Lambda = (3, 6.5)$ and V_{-4} everywhere else, as well as a total of 6 “extra” terms. The results were accurate on $t \in [0, 200]$, just like the full domain V_{-6} control results. So, the method worked for the linear version of the problem.

The non-linear Rossby wave problem also responded well to the multi-scale method. Similarly to the tests in Section 9, a set of V_{-3} with localized V_{-4} were calculated, using different sub-domains Λ and different numbers of “extra” terms (see Subsection 8.2). These were compared with full domain V_{-4} control results. The L_1 , L_2 and L_∞ norms of the differences between the multi-scale and control results were calculated. As in Section 9, the calculated differences between the test and control results showed a convergence to zero as the width of Λ was increased.

In Section 8 two modifications to the basic method are introduced, one necessary for accurate results with a non-linear problem, the other contributing to accuracy for any problem. There are several additional features beyond those in Section 8 that should be tested.

- First is the fact that both the large and small-scale systems should have adaptive decompositions. Due to how the systems are divided, there is no particular reason why the small-scale system in Λ could not follow any given adaptive scheme (that is compatible with an implicit time-discretization). Most of the changes in an adaptive scheme would be towards the finer resolution terms, which interact very little with the large-scale system. The real question is how to determine if a particularly tight concentration of fine resolution terms merits creating further small-scale systems.
- Giving the large-scale system an adaptive decomposition could be more complicated. Increasing the resolution of the large-scale system within Λ would probably be a bad idea. That would create duplicate terms (those that are calculated twice, once in each) to no net benefit. However, a few,

isolated, small-scale calculations outside of Λ would obviously give greater accuracy, at limited expense. Again, the real question is when to decide that a set of small-scale terms would be best given its own small-scale system, and be calculated separately from the large-scale system.

- Our two dimensional example involves a sub-domain Λ that covers a small subset of the domain in the y direction, but covers the entire interval in the x direction. Basically, the example has two spatial dimensions, but the multi-scale method is only used in one of those dimensions. It would be instructive to test a problem requiring localization in two or more spatial dimensions. A turbulence related problem would be appropriate for testing purposes, or any problem from fluid dynamics that results in small-scale vortices.
- An analysis of the effect of our method on the stability of the underlying time-discretization. Preliminary testing was done in [23], showing a minimal effect on stability, but more is needed.
- Using multiple small-scale systems, either nested or discrete, as shown in Figure 2. If a problem requires high resolution in two regions, call them Λ_1 and Λ_2 , it may be possible to calculate them separately, after the calculation of the large-scale system. The large-scale system could have, for example, 800 elements, with 400 small-scale elements and 200 large-scale in each of Λ_1 and Λ_2 . Solving all of these together would involve 1200 elements. Solving them broken up into three systems would involve an 800 element system and two of 400. This should involve substantial savings.
- Using a different time step size for the small-scale system. Instead of calculating $\mathbf{b}^{n+1}$ after each time step of the large-scale system, we could calculate $\mathbf{b}^{n+\frac{1}{2}}$ then $\mathbf{b}^{n+1}$ (or $\mathbf{b}^{n+\frac{1}{4}}$, $\mathbf{b}^{n+\frac{1}{2}}$, $\mathbf{b}^{n+\frac{3}{4}}$, then $\mathbf{b}^{n+1}$). Doing so requires the intermediate large-scale vectors $\mathbf{a}$, which are relatively easy to calculate via interpolation. However, further experimentation is necessary to find how well this works, and what modifications may be necessary.
- Making the size of Λ , the resolutions used, and the number of “extra” terms adaptive. This requires analyzing the sources of error stemming from the boundaries of Λ and the lack of “extra” terms.

Additionally, the key steps and components of this method could be combined with other methods. The method should be compatible with any multi-scale decomposition, and any procedures for solving the resulting systems. As long as the partial differential equation is time based, and the time-discretization is implicit, our multi-scale method should be useable. The nature of the partial differential equation itself will determine if our method would be useful.

So, in summary, this multi-scale implicit method shows potential for solving certain partial differential equations, namely those that produce stiff decompositions and require fine resolution in small regions. The preliminary tests went well, but further testing of the method, and extensions to the method, are required.

Acknowledgements

Thanks are due to the anonymous referees, who provided helpful, pointed, and insightful reviews of the initial submission.

References

1. Daubechies, I. *Ten Lectures on Wavelets*; SIAM: Philadelphia, PA, USA, 1992.
2. Blatter, C. *Wavelets, A Primer*; A K Peters Ltd.: Natick, MA, USA, 1998.
3. Meyer, Y. *Ondelettes et Fonctions Splines*; Séminaire EDP, Ecole Polytechnique: Paris, France, 1986.
4. Lemarié, P.G. Ondelettes à localisation exponentielle. *J. Math. Pures Appl.* **1988**, *67*, 227–236.
5. Daubechies, I. Orthonormal bases of compactly supported wavelets. *Commun. Pure Appl. Math.* **1988**, *41*, 909–996.
6. Cohen, A.; Daubechies, I.; Feauveau, J.-C. Biorthogonal bases of compactly supported wavelets. *Commun. Pure Appl. Math.* **1992**, *45*, 485–560.
7. Farge, M.; Schneider, K. Coherent Vortex Simulation (CVS), a semi-deterministic turbulence model using wavelets. *Flow Turbul. Combust.* **2001**, *66*, 393–426.
8. Alam, J.M.; Kevlahan, N.K.R.; Vasilyev, O.V. Simultaneous space-time adaptive wavelet solution of nonlinear parabolic differential equations. *J. Comput. Phys.* **2006**, *214*, 829–857.
9. Hesthaven, J.S.; Jameson, L.M. A wavelet optimized adaptive multi-domain method. *J. Comput. Phys.* **1998**, *145*, 280–296.
10. Léonard, S.; Terracol, M.; Sagaut, P. A wavelet-based adaptive mesh refinement criterion for large-eddy simulation. *J. Turbul.* **2006**, *7*, doi:10.1080/14685240601021608.
11. Beylkin, G.; Coifman, R.; Rokhlin, V. Fast wavelet transforms and numerical algorithms I. *Commun. Pure Appl. Math.* **1991**, *44*, 141–183.
12. Sweldens, W. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.* **1998**, *29*, 511–546.
13. Harten, A. *Multiresolution Representation and Numerical Algorithms: A Brief Review*; ICASE/LaRC Interdisciplinary Series in Science and Engineering; Springer: Berlin, Germany, 1997, Volume 4, pp. 289–322.
14. Vasilyev, O.V.; Kevlahan, N.K.-R. An adaptive multilevel wavelet collocation method for elliptic problems. *J. Comput. Phys.* **2005**, *206*, 412–431.
15. Vasilyev, O.V.; Paolucci, S. A dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain. *J. Comput. Phys.* **1996**, *125*, 498–512.
16. Müller, S.; Stiriba, Y. A multilevel finite volume method with multiscale-based grid adaptation for steady compressible flows. *J. Comput. Appl. Math.* **2009**, *227*, 223–233.
17. Mehra, M.; Kevlahan, N.K.-R. An adaptive multilevel wavelet solver for elliptic equations on an optimal spherical geodesic grid. *SIAM J. Sci. Comput.* **2008**, *30*, 3073–3086.
18. Tang, Y.Y.; Wickerhauser, V.; Yuen, P.C.; Li, C. *Wavelet Analysis and Its Applications*; Springer: New York, NY, USA, 2001.
19. Campbell, L.J. Wave-Mean-Flow interactions in a forced rossby wave packet critical layer. *Stud. Appl. Math.* **2004**, *112*, 39–85.
20. Kundu, P.K.; Cohen, I.M. *Fluid Mechanics*, 3rd ed.; Elsevier Academic Press: New York, NY, USA, 2004.

21. Campbell, L.J.; Maslowe, S.A. Forced rossby wave packets in baratropic shear flows with critical layers. *Dyn. Atmos. Oceans* **1998**, *28*, 9–37.
22. Béland, M. Numerical study of the nonlinear rossby wave critical level development in a barotropic zonal flow. *J. Atmos. Sci.* **1976**, *33*, 2066–2078.
23. McLaren, D.A. Sequential and Localized Implicit Wavelet-Based Solvers for Stiff Partial Differential Equations. Ph.D. Thesis, University of Ottawa, Ottawa, Canada, 2012.

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).