Calhoun: The NPS Institutional Archive

DSpace Repository

Faculty and Researchers | Faculty and Researchers' Publications

2012

An element-based spectrally-optimized approximate inverse preconditioner for the Euler equations

Carr, L.E. III; Borges, C.F.; Giraldo, F.X.

https://hdl.handle.net/10945/38323

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

AN ELEMENT-BASED SPECTRALLY-OPTIMIZED APPROXIMATE INVERSE PRECONDITIONER FOR THE EULER EQUATIONS

L.E. CARR III , C.F. BORGES , AND F.X. GIRALDO *

Abstract. We introduce a method for constructing an element-by-element sparse approximate inverse (SAI) preconditioner designed to be effective in a massively-parallel spectral element modeling environment involving non-symmetric systems. This new preconditioning approach is based on a spectral optimization of a low-resolution preconditioned system matrix (PSM). We show that the local preconditioning matrices obtained via this element-based, spectrum-optimized (EBSO) approach may be applied to arbitrarily high-resolution versions of the same system matrix without appreciable loss of preconditioner performance. We demonstrate the performance of the EBSO preconditioning approach using 2-D spectral element method (SEM) formulations for a simple linear conservation law and for the fully-compressible 2-D Euler equations with various boundary conditions. For the latter model running at sufficiently large Courant number, the EBSO preconditioner significantly reduces both iteration count and wall-clock time regardless of whether a generalized minimum residual (GMRES) or a stabilized biconjugate gradient (BICGSTAB) iterative scheme is employed. To assess the value added by this new preconditioning approach, we compare its performance against two other equally-parallel SAI preconditioning methods: low-order Chebyshev generalized least-squares polynomials and an element-based variant of the well-known Frobenius norm optimization preconditioner which we also develop herein. The EBSO preconditioner significantly out-performs both the Chebyshev polynomials and the element-based Frobenius-norm-optimized (EBFO) preconditioner regardless of whether the GMRES or BICGSTAB iterative scheme is employed. Moreover, when the EBSO preconditioner is combined with the Chebyshev polynomial method dramatic reductions in iterations per time-step can be achieved while still achieving a significant reduction in wall-clock time.

Key words. preconditioning; sparse approximate inverse; element-based; spectral elements; Galerkin methods; Euler Equations; nonhydrostatic atmospheric model

AMS subject classifications. 65M60, 65M70, 35L65, 86A10

1. Introduction.

1.1. Modeling Context and Preconditioner Requirements. A recent paper by Giraldo *et al.* [7] studies a number of semi-implicit (i.e., implicit-explicit, hereafter IMEX) spectral element (SE) formulations of the compressible Navier Stokes equations with a view towards application to operational nonhydrostatic atmospheric modeling over both regional and global domains [9]. The combination of the high-resolution associated with nonhydrostatic modeling and the large domain size associated with global modeling requires as many as $O(10^7)$ elements and $N_g = O(10^9)$ grid points (nodes). As a result, at each time-step in the IMEX time integration process there is a need to iteratively solve a very large, but sparse, linear system of the form

$$A\boldsymbol{q}^{n+1} = R(\boldsymbol{q}^n), \tag{1.1}$$

where $\boldsymbol{q}$ is a state vector, R is a right-hand side operator, and the matrix A is fixed, square, invertible, nonsymmetric and in general has a complex eigenspectrum.[1] As discussed in [7, 9], at a minimum the size of A is $N_g \times N_g$ if a *Schur* form is derived for A, and at worst the size of A is $4N_g \times 4N_g$ for 2-D modeling and $5N_g \times 5N_g$ for 3-D modeling if A is left in the non-Schur form.

The combination of very large (and increasing) system sizes and the hard wall-clock time constraints imposed on operational atmospheric prediction models (run every 6 hours with typically less than an hour allotted to the time integration phase) has not only necessitated parallel processing for some years, but also is rapidly heading into the realm of massively parallel processing. For example, the development work on the explicitly integrated Nonhydrostatic Unified Model of the Atmosphere (NUMA) reported in [9] is currently approaching $O(10^5)$ processors.

The motivation for pursuing the IMEX approach to solve such a challenging problem is that IMEX methods permit time-steps that can be as much as 100 times (or more) the maximum allowable explicit time-step. As a result, IMEX-based models can actually run faster than explicit models

*Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA 93943, USA

[1] The real spectra shown in Fig. 4.5 of [7] is a special case arising from the combination of using identical square elements and a Schur form for A. The Schur form eigenspectrum becomes complex when other element geometries are employed, and the non-Schur form always possesses a complex spectrum.

provided that the number of iterations needed to solve Eq. (1.1) is kept under control by a sufficiently effective preconditioner.

To summarize the preconditioned iterative approach to solving Eq. (1.1) at each time-step in the model integration, we first rewrite Eq. (1.1) using the standard symbology

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{1.2}$$

and then proceed to obtain its solution by instead solving either the equivalent left-preconditioned system

$$(KA)\boldsymbol{x} = K\boldsymbol{b} \tag{1.3}$$

or equivalent right-preconditioned system

$$(AK)\boldsymbol{y} = \boldsymbol{b} \tag{1.4a}$$
$$K\boldsymbol{y} = \boldsymbol{x} \tag{1.4b}$$

or sometimes a combination of the above two approaches.

In general, the large variety of preconditioning methods in existence can be divided into two classes that have been described as *explicit* and *implicit* [4, p. 970]. If explicit preconditioning is used, then K is a sparse matrix that is designed to approximate A^{-1} in some sense, and thus is often called a sparse approximate inverse (SAI). In this case, application of the preconditioner during the iterative solution process involves only matrix-vector products, which in general tends to be more amenable to parallel implementation. On the other hand, if implicit preconditioning is being used, then $K = M^{-1}$, where M is a sparse matrix that approximates A. In this case, Eq. (1.4a) becomes

$$A(M^{-1}\boldsymbol{y}) = \boldsymbol{b}, \tag{1.5}$$

where to calculate the parenthetical expression during the iterative process we instead solve the sparse system

$$M\tilde{\boldsymbol{y}} = \boldsymbol{y}. \tag{1.6}$$

Strategies to efficiently solve Eq. (1.6) involve various factoring methods such as incomplete LU (ILU) [2] and various multigrid methods [15], which can be made highly parallel, but compared to matrix-vector products represent a more challenging and complex problem.

If we define the residual vector associated with the ith approximate solution to Eq. (1.2) to be

$$\boldsymbol{r}_i = \boldsymbol{b} - A\boldsymbol{x}_i, \tag{1.7}$$

then an important distinction between Eq. (1.3) and (1.4) is that the stopping criterion for a left-preconditioned algorithm is based on $||K\boldsymbol{r}||$ as opposed to $||\boldsymbol{r}||$ for a right-preconditioned algorithm. Thus, if K is ill-conditioned, then the magnitude of $||\boldsymbol{x} - \boldsymbol{x}_i||$ obtained via a left-preconditioning algorithm may not be accurately reflected by the magnitude of $||K\boldsymbol{r}_i||$. Conversely, if the condition number of K is small, as is the case for problems solved in this paper, then either Eq. (1.3) or Eq. (1.4) can be used, and the choice of which to use simply depends on what works best for a particular problem [12, p. 272]. The results shown in Section 5 are for a *left* EBSO preconditioner (and comparison preconditioners) since we have found that left-preconditioning provides the best results for the modeling problems addressed herein.

Given the above modeling context, an effective preconditioned iterative approach to solving Eq. (1.1) must meet the following general requirements:

1. The iterative scheme must be able to handle a system matrix A that is non-symmetric with potentially a complex spectrum.
2. The preconditioner must have an application cost that is low in relation to the effectiveness of the preconditioner in reducing the number of iterations so that the wall-clock time for

the preconditioned model is significantly smaller than for the unpreconditioned model.[2] Ideally this net speed up should occur when running the model in serial mode (i.e., before parallelization is considered).

3. The preconditioner must provide no barrier to iterative method scalability in a massively parallel computing environment involving potentially millions of processors.
4. The preconditioner should preferably be based on the same SE-based parallelism employed when operating on a vector by the system matrix A to facilitate the simplest and maximally parallel implementation possible.[3]

1.2. Paper Format. Having laid out the modeling context of our preconditioning problem, the format of the remainder of the paper is as follows. In Section 2 we briefly survey the basic types of iterative methods and preconditioners currently available, and identify those suitable for our purposes. In particular we describe the preconditioners we will use both as baselines for assessing EBSO performance, and in one case for combining with the EBSO preconditioner to enhance its performance. In Section 3 we provide the mathematical basis for the particular global structure of the matrix K employed in the EBSO preconditioning approach, and in Section 4 we describe the process by which the local matrices of the EBSO and EBFO preconditioners are actually computed. In Section 5 we illustrate and analyze the results of applying the EBSO preconditioning approach to both a linear 2-D conservation law and a 2-D version of NUMA run in serial mode. Section 6 concludes the paper with a summary and outline of planned future work in this area.

2. Iterative Methods and Comparison Preconditioners.

2.1. Iterative Methods. Criterion 1 in the list in Section 1.1 above rules out the use of the highly efficient conjugate gradient (CG) method, for which convergence is guaranteed only if the matrix is symmetric positive definite (SPD). The results we show in this paper are based on both the generalized minimum residual (GMRES) scheme [13] based on the Arnoldi process [12, p. 165] and the transpose-free, stabilized biconjugate gradient (BICGSTAB) scheme [16] based on the Lanczos process [12, p. 234]. GMRES applies the system matrix once per iteration, whereas transpose-free BICGSTAB applies the system matrix twice. By contrast, if n is the number of iterations, then the number of dot-products performed by GMRES is $n^2/2$, as opposed to $6n$ for BICGSTAB. Thus, in general which method will perform better depends on the particular problem. Other things being equal, problems with a relatively large system size and relatively small number of iterations should tend to favor GMRES. By contrast, relatively small system size and a larger number of iterations would favor BICGSTAB, especially as the processor count increases due to the increasing communication penalty for each dot product.

2.2. Comparison Preconditioners. Before turning to the *explicit* preconditioners discussed below, we should make brief mention of why, at least in our current work, we are excluding consideration of *implicit* preconditioners. The justification is basically pragmatic. Per Criteria 3 and 4 above, the explicit preconditioners we consider are more naturally parallel and very simple to implement in a SE modeling environment. By contrast, efficient parallel implementation of implicit preconditioners requires considerably more effort and sophistication as discussed at length in [2, 12]. Although ILU methods in particular can be very powerful in serial computing, and have been shown to scale up to several hundred processors [2], whether they can be scalable in *massively* parallel computing environments remains to be seen. Since we are able to obtain considerable performance improvements with simple-to-implement explicit preconditioners, deferring consideration of implicit methods seems justified for the time being.

[2] Ironically, the need for a low application cost is contrasted by a loose constraint on the initial set-up or construction cost of a preconditioner to be used in our modeling problem. A potentially high construction cost is acceptable because the same preconditioner can be used not only in hundreds of time-steps in each model run, but also for potentially *years-worth* of runs in a particular version of an operational weather model (typically run four times a day).

[3] Preconditioners that may be simply implemented are particularly desirable in atmospheric prediction systems which already involve highly complex and massive code structures. Preconditioners requiring more complex parallel implementation schemes should be considered only if necessary to achieve adequate performance.

2.2.1. Polynomial Preconditioners. The polynomial preconditioner concept is an old idea that continues to find application [10, 11], and a concise summary of the pros and cons of the approach can be found in [5, p. 306]. Because all such preconditioners consist of a polynomial of the system matrix A and preconditioner application involves only matrix-vector products, they have the advantage of being very amenable to parallel implementation. In the case of a spectral element modeling environment, the advantage becomes considerable since any polynomial in A is itself element-based, which means it can be implemented using precisely the same highly parallel approach used to apply the system matrix. Moreover, polynomial preconditioners have a virtually non-existent storage requirement (i.e., a few scalars).

The principal disadvantages are that:

1. These preconditioners usually do not perform well on matrices with complex spectra.
2. Even for system matrices with real spectra the reduction in iterations is usually offset by the cost of application.
3. Prior knowledge of the spectrum of A is necessary.

With regard to disadvantages 1) and 2), we will later show that a polynomial preconditioner can perform exceedingly well when combined with the EBSO preconditioner (which creates a KA with a complex spectrum), significantly reducing *both* number of iterations and wall-clock time. As discussed in [12, p. 383, 387], disadvantage 3) means that for problems that are solved only once the polynomial preconditioner must be embedded in a Krylov space method such as GMRES and not used in the iterative process until the spectrum of A can be estimated by using the QR method for finding the extremal eigenvalues of the Hessenberg matrix generated as part of the Arnoldi process [14]. This requirement normally makes effective implementation of polynomial preconditioners a significant challenge. However, in an atmospheric IMEX model the structure of A is fixed, and thus we have the luxury of estimating the spectrum of A to whatever accuracy we need prior to commencing the time integration of the model. Having explained how we will overcome principal disadvantages, we now explain the particular polynomial preconditioner we will employ.

As mentioned earlier, all SAI preconditioning methods seek to create a sparse matrix K such that $K \approx A^{-1}$. For example, if the spectrum of A lies on the real interval $[\lambda_{min}, \lambda_{max}]$ where $\lambda_{min} > 0$ (as is the case for the Schur form matrices employed in [7]), then the spectrum of A^{-1} lies on the interval $[1/\lambda_{max}, 1/\lambda_{min}]$. In view of this fact, it is desirable that an explicit preconditioner K have a spectrum that is distributed over the interval $[1/\lambda_{max}, 1/\lambda_{min}]$ in a manner similar to A^{-1}. Least-squares polynomial preconditioners attempt to satisfy this criterion by specifying that K be a kth-order polynomial in A

$$K = s(A) = \sum_{i=0}^{k} c_i A^i, \tag{2.1}$$

so that the left preconditioned system in Eq. (1.3) becomes

$$s(A)A\boldsymbol{x} = s(A)\boldsymbol{b}, \tag{2.2}$$

and then requiring $s(A)$ to satisfy the optimization problem

$$s(\ ) = \min \|1 - \lambda s(\lambda)\|_w \qquad \|p(\lambda)\|_w \equiv \int_E p^2(\lambda) w(\lambda) d\lambda, \tag{2.3}$$

where $w(\lambda)$ is a weighting function that is nonnegative over region E on the complex plane, and E encloses the spectrum of A (*e.g.,* $E = [\lambda_{min}, \lambda_{max}]$ if the spectrum of A is real).

If we temporarily assume that $\lambda_{min} = 0$ and $\lambda_{max} = 1$ and require the weighting function to have the form

$$w(\lambda) = \lambda^{\mu-1}(1-\lambda)^{\nu}, \tag{2.4}$$

and further specify that $\mu = \frac{1}{2}$ and $\nu = -\frac{1}{2}$, then the polynomial $s(\lambda)$ can be assembled recursively from Chebyshev polynomials of the first kind as explained in Saad [12, p. 383-5]. These polynomials

	c_0	c_1	c_2	c_3	c_4	c_5
s_1	5	-2				
s_2	14	-14	4			
s_3	30	-54	36	-8		
s_4	55	-154	176	-88	16	
s_5	91	-364	624	-520	208	-32

TABLE I

Coefficient values for the first five Chebyshev-based polynomial preconditioners when the spectrum of the system matrix lies within the real interval $[0, 2]$.

can then be rescaled to apply to the interval $[0, \lambda_{max}]$ for arbitrary real $\lambda_{max} > 0$ by replacing λ with $4\lambda/\lambda_{max}$. For reasons explained in the next paragraph, we have chosen to let $\lambda_{max} = 2$, which results in the polynomial coefficients shown in Table I.

We then accurately estimate the extremal real eigenvalues λ_{min} and λ_{max} of a Schur form A using the Arnoldi process to create a Hessenberg matrix[4], whose eigenvalues are then determined via the QR algorithm. Once we have the extremal eigenvalues of A, we can then compute the mid-point value

$$\tilde{\lambda} = \frac{\lambda_{max} + \lambda_{min}}{2}, \tag{2.5}$$

which may then be used to define the scaling transformations

$$A = \tilde{\lambda}\tilde{A} \quad \boldsymbol{b} = \tilde{\lambda}\tilde{\boldsymbol{b}}, \tag{2.6}$$

where the scaled system matrix $\tilde{A}$ now has a spectrum that lies centered in the interval $[0, 2]$. Substituting Eq. (2.6) into Eq. (2.2) results in the equivalent preconditioned system

$$s(\tilde{A})\tilde{A}\boldsymbol{x} = s(\tilde{A})\tilde{\boldsymbol{b}},$$

for which the coefficients in Table I are applicable regardless of the values of λ_{min} and λ_{max} for any particular Schur form system matrix.

Before concluding this section, we emphasize that use of the above-described Chebyshev polynomial preconditioner is not strictly confined to matrices with real spectra. Chebyshev polynomial preconditioners can be constructed for non-symmetric system matrices with complex spectra as discussed in [12, p. 386]. In Section 5.2 we show that the preconditioners constructed using the coefficients in Table I are quite effective for system matrices KA, where K is the EBSO preconditioner, which exhibit complex spectra confined to a highly elliptical region with a semi-major axis that lies along the real interval $[0, 2]$.[5]

2.2.2. Frobenius Norm Optimization (FNO) Preconditioners. As discussed in [4, 5, 8], what we are terming a FNO preconditioner is the solution to linear least-squares optimization problems

$$K = \min_{K \in S} \|I - KA\|_F \qquad or \qquad K = \min_{K \in S} \|I - AK\|_F, \tag{2.7}$$

depending on whether a left or right preconditioner is to be constructed. The set S consists of all matrices of some user-specified sparsity pattern, which is usually based on the sparsity pattern of A.

Notice that whereas least-squares polynomial preconditioners based on Eq. (2.3) attempt to make the spectrum of the preconditioned system matrix (PSM) similar to I in a *spectral* sense, the FNO method instead attempts to make the PSM similar to I in an *entry-by-entry* sense. Despite

[4]The size of the matrix needed varies with Courant No. (i.e., time-step). Via some experimentation we have found that 300×300 is sufficient to ensure that λ_{min} and λ_{max} are accurate to 1 percent for Courant Nos. ≤ 32.

[5]In this case, Eq. (2.5) is replaced with $\tilde{\lambda} = (|\lambda_{min}| + |\lambda_{max}|)/2$, where $|\lambda_{min}|$ and $|\lambda_{max}|$ are the smallest and largest moduli of the spectrum of KA.

this indirect approach, the spectrum of an FNO-derived PSM (computable for smaller test systems) typically exhibits a tighter grouping around (1,0) on the complex plane than does the spectrum of A itself and accelerates iterative method convergence. Although Eq. (2.7) represents a global optimization problem, it can be recast into a set of N_g independent linear least-squares problems [5, p. 308] that can be solved by direct (*i.e.*, QR) or iterative means [12, p. 323]. Via such methods the entries of K are determined in a column-by-column fashion: a process which is amenable to parallelization, and thus makes FNO preconditioner construction tractable for very large systems.

Various strategies exist for parallel implementation of the FNO preconditioner when the sparsity pattern is based on the system matrix A. However, to exploit the highly parallel nature of the SE modeling environment, we show in Section 4.2 how an element-based FNO preconditioner (EBFO) can be constructed provided that we replace the sparsity pattern of A with another sparsity pattern that naturally arises during the so-called direct stiffness summation (DSS) process that is an inherent part of SE modeling and is discussed in detail in Section 3.2.

3. EBSO Preconditioner Development.

3.1. Overview of EBSO Preconditioner Concept. Recall from Section 2.2.1 that polynomial preconditioners are solutions to the optimization problem (2.3), which is based on a weighted L^2 norm. The key idea underlying the EBSO preconditioning approach is to replace (2.3) with the optimization problem

$$K = \min_{K \in S} \left\| \boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA) \right\|_{p,w}, \tag{3.1}$$

where:

* S is the set of matrices satisfying some user-specified sparsity pattern,
* $\boldsymbol{\sigma}$ is a vector-valued function whose components are the eigenvalues of the argument,
* subscript p denotes a modified form of a Euclidean p-norm,
* subscript w represents an imaginary-component weighting *factor* that is unrelated to the weighting *function* seen in Eq. (2.3).

The details of p-norm modification and the purpose of the factor w are explained in Section 4.1. For a large A associated with typical operational models, the above problem is intractable as it stands. However, we can transform Eq. (3.1) into a tractable non-linear least squares (NLLS) problem via the following set of observations and assumptions.

Firstly, as in all SE formulations, the *global* system matrix A is effectively assembled from *local* element matrices as indicated by the notation

$$A = \bigwedge_{e=1}^{N_e} A^e \tag{3.2}$$

where N_e is the number of elements and $\bigwedge$ denotes the usual DSS required by all continuous element-based Galerkin methods. Secondly, based on the observation that in certain types of SE problems many of the A^e matrices have related or even identical entries (e.g., when all elements are the same size and shape), we now assume that a useful global preconditioner K also might be assembled via

$$K = \bigwedge_{e=1}^{N_e} K^e \tag{3.3}$$

where large groups of the elemental matrices K^e's are *identical*. In fact, in the case of a two-dimensional model, we will see that a highly effective global K can be assembled from as few as three unique K^e matrices. By substituting Eqs. (3.2) and (3.3) into Eq. (3.1) we obtain the optimization problem

$$\min_{K \in S} \left\| \boldsymbol{\sigma}(I) - \boldsymbol{\sigma}\left(\left[\bigwedge_{e=1}^{N_e} K^e \right] \left[\bigwedge_{e=1}^{N_e} A^e \right] \right) \right\|_{p,w} \tag{3.4}$$

where the output of the optimization process is a few small matrices that determine K via Eq. (3.3). Computationally speaking, Eq. (3.4) is a significant improvement over Eq. (3.1), since the unknowns we are seeking are the optimal entries for a few small (but potentially full) matrices rather than all the non-zero entries of a global K.

Despite being an improvement over Eq. (3.1), solving Eq. (3.4) is still intractable for large A since it requires us to repeatedly compute the complete spectrum of the global matrix KA inside some NLLS algorithm. The solution to this problem is our observation that:

- if we reduce N_e to a tractable size (e.g., $N_e = 25$) so that a K^e matrix set can be created for a *low-resolution* version of a SE problem
- then we can utilize the same K^e matrix set for a *high-resolution* version of our SE problem (e.g., a 1000-fold increase in N_e) without appreciable loss of preconditioner performance.

3.2. EBSO Preconditioner Structure Formulation.

3.2.1. 1-D Analysis. The rationale for the chosen structure of the EBSO preconditioner is most clearly demonstrated within the context of using the SE method combined with implicit time-integration to solve a simple 2-D linear conservation law. For reasons that will become clear, we begin with the 1-D linear conservation law

$$\frac{\partial q}{\partial t} + c\frac{\partial q}{\partial x} = 0 \quad x \in \Omega \tag{3.5}$$

where for our purposes here we can set the basic state advective speed c to unity without any loss of generality. As in all SE formulations, we assume that the domain Ω is composed of elements

$$\Omega = \bigcup_{e=1}^{N_e} \Omega^{(e)} \tag{3.6}$$

and that the physical coordinate (x) and a standard reference coordinate (ξ) are related by the invertible mapping

$$x = f_e(\xi) \quad \xi \in [-1, +1]. \tag{3.7}$$

If we focus on the eth element and recall that $c = 1$, then Eq. (3.5) becomes

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0 \quad x \in \Omega^{(e)}. \tag{3.8}$$

Using an Nth order Lagrange polynomial basis

$$\{\phi_j(x)\}_{j=1}^{N+1}$$

we can express $q(x,t)$ as

$$q(x,t) = \sum_{j=1}^{N+1} q_j(t)\phi_j(x) + \epsilon_q(x,t) \tag{3.9}$$

where ϵ_q is a residual function arising from approximating $q(x,t)$ using a finite set of basis functions. Substituting Eq. (3.9) into (3.8) results in

$$\sum_{j=1}^{N+1} \frac{dq_j}{dt}\phi_j + \sum_{j=1}^{N+1} q_j \frac{d\phi_j}{dx} = -\frac{\partial \epsilon_q}{\partial t} - \frac{\partial \epsilon_q}{\partial x} \equiv \epsilon(x,t). \tag{3.10}$$

Next, we multiply Eq. (3.10) by any one of the basis functions (*i.e.*, test function) and integrate over the element domain

$$\sum_{j=1}^{N+1} \left(\int_{\Omega^{(e)}} \phi_i \phi_j dx \right) \frac{dq_j}{dt} + \sum_{j=1}^{N+1} \left(\int_{\Omega^{(e)}} \phi_i \phi_j' dx \right) q_j = \int_{\Omega^{(e)}} \phi_i \epsilon dx \quad i = 1, \ldots, N \tag{3.11}$$

where ϕ' denotes the derivative of ϕ with respect to x. If we now require the residual function ϵ to be orthogonal to the space spanned by the Lagrange polynomial set, then using matrix notation Eq. (3.11) becomes the normal system

$$M_1^e \frac{d\boldsymbol{q}}{dt} + D_1^e \boldsymbol{q} = \boldsymbol{0} \tag{3.12}$$

where the mass matrix (M_1^e), differentiation matrix (D_1^e), and state vector $\boldsymbol{q}$ are given by

$$M_1^e = \int_{\Omega^{(e)}} \phi_i \phi_j dx \tag{3.13a}$$

$$D_1^e = \int_{\Omega^{(e)}} \phi_i \phi_j' dx \tag{3.13b}$$

$$\boldsymbol{q} = \left[q_1 \ \cdots \ q_{N+1} \right]^T .$$

The subscript "1" on the left-hand sides of Eq. (3.13) denotes that the matrices are associated with a 1-D SE formulation, and are needed to distinguish these matrices from their 2-D counterparts that appear later. Integral formulas equivalent to Eq. (3.13), but expressed with respect to the element reference coordinate, are

$$M_1^e = \int_{-1}^{+1} \phi_i(\xi) \phi_j(\xi) J^e(\xi) d\xi \tag{3.14a}$$

$$D_1^e = \int_{-1}^{+1} \phi_i(\xi) \phi_j'(\xi) d\xi \tag{3.14b}$$

where $J^e(\xi)$ represents the Jacobian

$$J^e(\xi) = \frac{d}{d\xi} f^e(\xi)$$

and ϕ' now represents the derivative of ϕ with respect to ξ.

In general, the Jacobian is a non-linear function of ξ. However, if there is an affine relationship between the physical and reference coordinates system variables, then the Jacobian is a constant for each element and is simply the ratio of the physical and reference coordinate domains of the element

$$J^e = \frac{\Delta x^e}{2} \qquad \Delta x^e = \texttt{length}\left(\Omega^{(e)}\right).$$

In such a case, Eqs. (3.14) can be written

$$M_1^e = J^e \overline{M}_1 \tag{3.15a}$$

$$\overline{M}_1 = \int_{-1}^{+1} \phi_i \phi_j d\xi \tag{3.15b}$$

$$D_1^e = \overline{D}_1 \tag{3.16a}$$

$$\overline{D}_1 = \int_{-1}^{+1} \phi_i \phi_j' d\xi \tag{3.16b}$$

where $\overline{M}_1$ and $\overline{D}_1$ are the mass and differentiation matrices for the 1-D reference element, respectively. From Eq. (3.15) it is clear that for a constant Jacobian the mass matrix of each element is just the reference mass matrix scaled by the Jacobian for that element. From Eq. (3.16a) it is clear that every element differentiation matrix is identical to the reference element differentiation matrix since the Jacobian is absent in Eq. (3.14b). These properties will have important implications later on.

Whereas Eq. (3.12) applies to a single element, the analogous form that is defined globally on the domain Ω is

$$M\frac{d\boldsymbol{q}}{dt} + D\boldsymbol{q} = \boldsymbol{0}$$

where the global mass and differentiation matrices are obtained via the DSS operations

$$M = \bigwedge_{e=1}^{N_e} M_1^e = \bigwedge_{e=1}^{N_e} J^e \overline{M}_1 \qquad D = \bigwedge_{e=1}^{N_e} D_1^e = \bigwedge_{e=1}^{N_e} \overline{D}_1.$$

3.2.2. 2-D Analysis. With the above 1-D analysis in mind, we now consider the 2-D linearized conservation law analogous to Eq. (3.5), which is

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0 \quad (x,y) \in \Omega. \tag{3.17}$$

In general, the relationships between the global and elemental domains and the associated physical and reference coordinate systems are given by Eq. (3.6) and

$$x = f_e(\xi,\eta) \qquad y = g_e(\xi,\eta) \qquad \xi,\eta \in [-1,+1] \tag{3.18}$$

$$J_e(\xi,\eta) = \left| \frac{\partial f_e}{\partial \xi}\frac{\partial g_e}{\partial \eta} - \frac{\partial g_e}{\partial \xi}\frac{\partial f_e}{\partial \eta} \right|. \tag{3.19}$$

However, if all elements are rectangular and the physical and reference coordinate systems are again related in an affine manner, then Eqs. (3.18) and (3.19) simplify to

$$x = f_e(\xi) \qquad y = g_e(\eta)$$

$$J_e(\xi,\eta) = \left| \frac{\partial f_e}{\partial \xi}\frac{\partial g_e}{\partial \eta} \right| = \frac{\Delta x^e}{2}\frac{\Delta y^e}{2} = J_x^e J_y^e$$

where the product of the two Jacobians is simply the ratio of the element area in the physical and reference coordinate systems.

If we again focus on the eth element, then Eq. (3.17) becomes

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} = 0 \quad (x,y) \in \Omega^{(e)}. \tag{3.20}$$

We now define a N_2-member multivariate basis

$$\{\psi_k(x,y)\}_{k=1}^{N_2}$$

and expand $q(x,y,t)$ in terms of these 2-D basis functions

$$q(x,y,t) = \sum_{k=1}^{N_2} q_k(t)\psi_k(x,y) + \epsilon_q(x,y,t) \tag{3.21}$$

where again e_q is a residual function. Substituting Eq. (3.21) into (3.20) results in

$$\sum_{k=1}^{N_2} \frac{dq_k}{dt}\psi_k + \sum_{k=1}^{N_2} q_k \frac{\partial \psi_k}{\partial x} + \sum_{k=1}^{N_2} q_k \frac{\partial \psi_k}{\partial y} = -(\epsilon_q)_t - (\epsilon_q)_x - (\epsilon_q)_y \equiv \epsilon(x,y,t). \tag{3.22}$$

As in the 1-D example, we now multiply Eq. (3.22) by any one of the 2-D basis functions and integrate over the element domain, and require the residual to be orthogonal to all the basis functions. The resulting equation set, expressed in matrix notation, is

$$M_2^e \frac{d\boldsymbol{q}}{dt} + D_2^{e,x}\boldsymbol{q} + D_2^{e,y}\boldsymbol{q} = \boldsymbol{0} \tag{3.23}$$

where the 2-D element mass matrix and differentiation matrices are

$$M_2^e = \int_{\Omega^e} \psi_l \psi_k dx dy \tag{3.24a}$$

$$D_2^{e,x} = \int_{\Omega^{(e)}} \psi_l (\psi_k)_x dx dy \tag{3.24b}$$

$$D_2^{e,y} = \int_{\Omega^{(e)}} \psi_l (\psi_k)_y dx dy \tag{3.24c}$$

with $k, l = 1, \dots N_2$.

In order to make use of our earlier 1-D analysis, we rewrite the above matrices for the 2-D conservation law in terms of the matrices for the 1-D conservation law by

- requiring the number of 2-D basis functions N_2 to be $(N+1)^2$,
- writing the 1-D basis functions in terms of the vector $\boldsymbol{\phi}(x) = [\phi_1(x) \ \cdots \ \phi_{N+1}(x)]^T$
- equating each 2-D basis function with the appropriate entry of the outer product of $\boldsymbol{\phi}$ with itself via a column-wise bijective rule for associating the index k with the indices i and j:

$$\begin{bmatrix} \psi_{k=1} & \cdots & \psi_{N(N+1)+1} \\ \downarrow & \cdots & \downarrow \\ \psi_{N+1} & \cdots & \psi_{(N+1)^2} \end{bmatrix} = \begin{bmatrix} \phi_{i=1}(x)\phi_{j=1}(y) & \cdots & \phi_1(x)\phi_{N+1}(y) \\ \vdots & \ddots & \vdots \\ \phi_{N+1}(x)\phi_1(y) & \cdots & \phi_{N+1}(x)\phi_{N+1}(y) \end{bmatrix}.$$

It will be convenient to represent the above basis-function-association rule using the concise notations

$$\psi_k = (\phi_i \phi_j)_k = (\phi_i)_k (\phi_j)_k \tag{3.25}$$

where

- the value of the outer index k determines the values of the inner indices i and j
- the index i implies that ϕ depends on x
- the index j implies that ϕ depends on y.

Substituting Eq. (3.25) into (3.24) and exploiting the fact that the double integrals are separable gives:

$$M_2^e = \int_{\Omega^{(e)}} (\phi_i)_k (\phi_i)_l dx \int_{\Omega^{(e)}} (\phi_j)_k (\phi_j)_l dy \tag{3.26a}$$

$$D_2^{e,x} = \int_{\Omega^{(e)}} (\phi_i)_k (\phi_i')_l dx \int_{\Omega^{(e)}} (\phi_j)_k (\phi_j)_l dy \tag{3.26b}$$

$$D_2^{e,y} = \int_{\Omega^{(e)}} (\phi_i)_k (\phi_i)_l dx \int_{\Omega^{(e)}} (\phi_j)_k (\phi_j')_l dy \tag{3.26c}$$

where $k, l = 1, \dots, (N+1)^2$.

Comparing the right hand sides of Eq. (3.26) with Eq. (3.13) reveals that all entries in the 2-D element mass and differentiation matrices are *Kronecker* products of the 1-D mass and differentiation matrices, which we denote by

$$M_2^e = M_1^e \otimes M_1^e \quad D_2^{e,x} = D_1^e \otimes M_1^e \quad D_2^{e,y} = M_1^e \otimes D_1^e. \tag{3.27}$$

By making use of Eq. (3.15a) and (3.16a), we can re-express Eq. (3.27) in terms of the 1-D reference mass and differentiation matrices

$$M_2^e = J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1 \right) \tag{3.28a}$$

$$D_2^{e,x} = J_y^e \left(\overline{D}_1 \otimes \overline{M}_1 \right) \tag{3.28b}$$

$$D_2^{e,y} = J_x^e \left(\overline{M}_1 \otimes \overline{D}_1 \right). \tag{3.28c}$$

Note that whereas Eq. (3.23) applies to a single element, the analogous form that is defined globally on the domain Ω is

$$M\frac{d\boldsymbol{q}}{dt} + (D_x + D_y)\,\boldsymbol{q} = \boldsymbol{0} \tag{3.29}$$

where the global mass and differentiation matrices above are obtained via the assembly operations

$$M = \bigwedge_{e=1}^{N_e} M_2^e \tag{3.30a}$$

$$D^x = \bigwedge_{e=1}^{N_e} D_2^{e,x} \tag{3.30b}$$

$$D^y = \bigwedge_{e=1}^{N_e} D_2^{e,y} \tag{3.30c}$$

or equivalently, after substituting Eq. (3.28) into Eq. (3.30)

$$M = \bigwedge_{e=1}^{N_e} J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1\right) \qquad D^x = \bigwedge_{e=1}^{N_e} J_y^e \left(\overline{D}_1 \otimes \overline{M}_1\right) \qquad D^y = \bigwedge_{e=1}^{N_e} J_x^e \left(\overline{M}_1 \otimes \overline{D}_1\right). \tag{3.31}$$

The last step toward identifying the desired structure of the EBSO preconditioner is to discretize Eq. (3.29) in time using a simple two-level implicit time differencing scheme:

$$M\frac{\boldsymbol{q}^{m+1} - \boldsymbol{q}^m}{\Delta t} + (D_x + D_y)\left(\alpha\boldsymbol{q}^m + \beta\boldsymbol{q}^{m+1}\right) = \boldsymbol{0} \tag{3.32}$$

where $\alpha + \beta = 1$. Solving Eq. (3.32) for the unknown state vector gives us

$$\left[M + \beta\Delta t\,(D^x + D^y)\right]\boldsymbol{q}^{m+1} = \left[M - \alpha\Delta t\,(D^x + D^y)\right]\boldsymbol{q}^m \tag{3.33}$$

which has the form of the standard $A\boldsymbol{x} = \boldsymbol{b}$ matrix equation with

$$A = \left[M + \beta\Delta t\,(D^x + D^y)\right] \tag{3.34a}$$

$$\boldsymbol{b} = \left[M - \alpha\Delta t\,(D^x + D^y)\right]\boldsymbol{q}^m. \tag{3.34b}$$

Combining Eq. (3.34a) with (3.31), we can express the global system matrix A as

$$A = \bigwedge_{e=1}^{N_e} \left[J_x^e J_y^e \left(\overline{M}_1 \otimes \overline{M}_1\right) + \beta\Delta t J_y^e \left(\overline{D}_1 \otimes \overline{M}_1\right) + \beta\Delta t J_x^e \left(\overline{M}_1 \otimes \overline{D}_1\right)\right] \tag{3.35}$$

where we see that the *global* system matrix A is an assemblage of Kronecker products of *local* 1-D reference mass and differentiation matrices that have been scaled by the Jacobians relating the physical and reference variable coordinate systems. Notice in particular that for a given grid structure (which determines the Jacobians) and a fixed time-step Δt the structure of the system matrix A is fixed, and thus does not change throughout the course of the model time integration phase. Furthermore, the Schur form system matrix employed in both the 2-D and 3-D versions of NUMA has the exact same structure since it is a second order operator involving two applications of the DSS process. The point here is that the preconditioning strategy we develop for the 2-D problem can be directly applied to 3-D.

Based on the structure of the system matrix A that we see in Eq. (3.35), we will now assume that a global preconditioner K constructed according to the rule

$$K = \bigwedge_{e=1}^{N_e} J_x^e J_y^e \left(\overline{K}_1 \otimes \overline{K}_1\right), \tag{3.36}$$

will have the potential to be an effective preconditioner for accelerating the iterative solution to Eq. (3.33). We emphasize here that $\overline{K}_1$ is in principle a *single* small and local preconditioner matrix analogous to $\overline{M}_1$ in Eq. (3.31), and thus is distinctly different from the existing *implicit* element-by-element (EBE) preconditioning approach in which the local preconditioner matrix is necessarily different for every element [1, 12, 17].

4. EBSO and EBFO Preconditioner Construction and Application.

4.1. EBSO Preconditioner. In this initial investigation into the EBSO preconditioning approach we will restrict our focus to cases in which all domain elements are of equal size. Under this restriction the Jacobian factors in Eq. (3.36) are the same for all elements, and scale all entries (and thus all eigenvalues) of the global preconditioner K to the same degree. As a result, we can incorporate the Jacobian factors into the reference matrices in Eq. (3.36) and assemble a functionally equivalent global preconditioner according to the rule

$$K = \bigwedge_{e=1}^{N_e} (K_1 \otimes K_1) \,. \tag{4.1}$$

We now turn our attention in the next section to the scheme for computing and applying the set of K_1 matrices.

Here we assume that some combination of SE spatial discretization and IMEX time differencing generates the need to iteratively solve the left preconditioned matrix equation

$$KA\boldsymbol{q}^{m+1} = K\boldsymbol{b} \tag{4.2}$$

where the global system matrix A and global preconditioner K are assembled according to Eqs. (3.35) and (4.1), respectively. Based on the structure of K as specified by Eq. (4.1), the spectral optimization problem (3.1) now takes the form:

$$K_1 = \min_{K \in S} \|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} \tag{4.3}$$

where, were it not for the presence of boundary conditions, a single local K_1 matrix could conceivably be used to construct the global preconditioner K. However, since we must contend with the effect of boundary conditions on the global system matrix A, we must include a way to let the structure of the preconditioner K vary in response to boundary conditions, while at the same time keeping the number of independent variables to be computed to a manageable number. Via some experimentation we have found that a feasible strategy is to limit the number of unique local K matrices to three using the element-based assignment rule

$$K_1 = \begin{cases} K_I & \texttt{if element } e \texttt{ is in the domain } \textit{interior} \\ K_S & \texttt{if element } e \texttt{ includes a domain } \textit{side} \\ K_C & \texttt{if element } e \texttt{ includes a domain } \textit{corner} \end{cases} . \tag{4.4}$$

We have found that this strategy gives sufficient degrees of freedom for handling not only the same boundary condition on all sides of the domain, but also various combinations such as no-flux on the top and bottom and periodic on the left and right sides.[6] As a result of assignment rule (4.4), Eq. (4.3) is transformed into

$$\{K_I, K_S, K_C\} = \min_{K \in S} \|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} \,. \tag{4.5}$$

The norm on the right-hand side of Eq. (4.5) is obtained via the formula

$$\|\boldsymbol{\sigma}(I) - \boldsymbol{\sigma}(KA)\|_{p,w} = \left[\sum_{k=1}^{N_g} \left(\, |a_k - 1|^p + w \, |b_k|^p \, \right) \right]^{1/p} , \tag{4.6}$$

where $N_g = \text{size}(KA)$, and $a_k = \text{Re}(\lambda_k)$ and $b_k = \text{Im}(\lambda_k)$, where λ_k is an eigenvalue of KA. This formula is basically a p-norm that has been modified so that the imaginary component of each eigenvalue can be weighted independently (in practice, more *heavily*) of the real component via the

[6] We also investigated using different matrices for each side and each corner, but the resulting number of unknowns was so large that NLLS algorithm would not reliably converge to a sufficiently optimal local minimum.

factor w. Notice that if we choose $p = 2$ and $w = 1$, then the kth term of the sum in Eq. (4.6) is the square of the distance of the kth eigenvalue from (1,0) on the complex plane.

We have experimented with various NLLS algorithms to iteratively solve Eq. (4.5). The best results occur when we use a Gauss-Newton (GN) algorithm coupled with a golden section line search that terminates when the relative error estimate for the function minimum is $< 10^{-3}$. We also employ as first guesses for K_I, K_S, K_C the identity matrix scaled so that eigenvalues of KA start out very close to the origin. Through experimentation, we have found that letting $p = 8$ has the dual benefit of improving the convergence rate of the GN algorithm[7] and improving the convergence rate of GMRES and BICGSTAB by more tightly clustering the eigenvalues of KA about (1,0) on the complex plane. Again through experimentation, we have found that an adequate NLLS algorithm stopping criterion is a change in relative residual of less than 10^{-3}. To ensure that the NLLS algorithm has found a robust local minimum, we restart the algorithm several times with the values of K_I, K_S, K_C matrices perturbed by 5 percent. For an SE interpolation order $N = 5$, the GN algorithm running on a current desktop PC takes about an hour to compute the $3 \times (N+1)^2 = 108$ unknowns that comprise the EBSO preconditioner matrices K_I, K_S, K_C.

Once the matrices K_I, K_S, K_C have been computed, the global EBSO preconditioner can be assembled via

$$K = \bigwedge_{e=1}^{N_e} (K_i \otimes K_i), \qquad i = I, S, C \tag{4.7}$$

where index i is set to I, S, or C depending where element e is located in the model domain per (4.4). For examples of the global sparsity pattern that results from (4.7), we refer the reader to Figs. 5.1b and 5.2b.

Finally, since the matrices K_I, K_S, K_C are fixed, their Kronecker products are also fixed and need not be recomputed at every iteration. Thus, before running the preconditioned model we compute and store the three $(N+1)^2$ by $(N+1)^2$ matrices

$$(K_I \otimes K_I) \qquad (K_S \otimes K_S) \qquad (K_C \otimes K_C,)$$

to maximize the implementation efficiency of the EBSO preconditioner.

4.2. EBFO Preconditioner. To create a EBFO preconditioner that is basically comparable to the EBSO preconditioner, we begin with a knowledge of the global sparsity pattern resulting from the DSS process as illustrated in Figs. 5.1b and 5.2b for $N_e = 25$ and $N_e = 100$, respectively. We then perform the following steps:

1. Construct a global FNO-based preconditioner K using the column-by-column process described in Section 2.2.2 using a low-resolution system (*e.g.*, $N_e = 100$).
2. Disassemble the global FNO-based K by reversing the DSS process to obtain N_e individual elements matrices of size $(N+1)^2$ by $(N+1)^2$.
3. Average all the interior element matrices to create a single K_I.
4. Average the element matrices of each side to create four distinct K_S matrices.
5. Average the element matrices of each corner to create four distinct K_C matrices.

To assemble a global EBFO preconditioner we use

$$K = \bigwedge_{e=1}^{N_e} K_i, \qquad i = I, S, C$$

in which case no Kronecker products are necessary since the K_i matrices here are already size $(N+1)^2$ by $(N+1)^2$.

We emphasize that steps 3-5 are not required for a functional EBFO preconditioner, but are included both to drastically reduce the number of local matrices that must be stored, as well as to

[7] We note in passing that the GN algorithm requires the objective function to be differentiable and thus, for example, would preclude the use of $p = 1$.

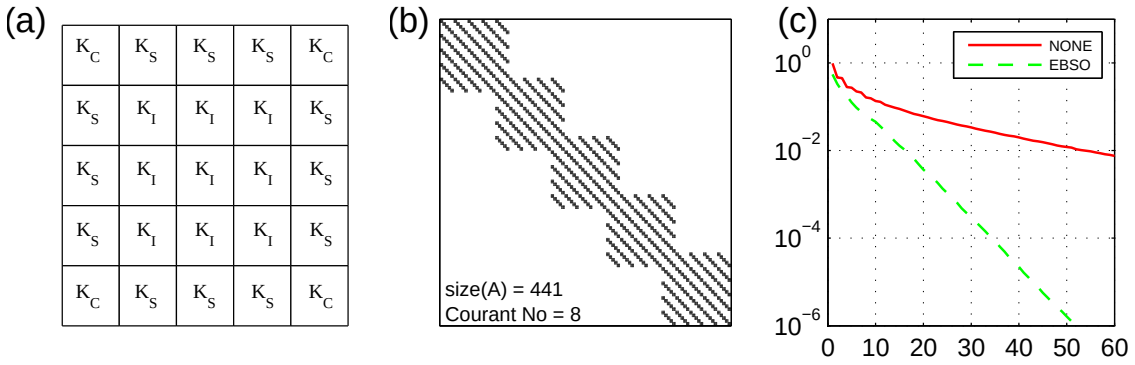

FIGURE 5.1. *(a) Distribution of local EBSO preconditioner matrices within a model domain of 5-by5 elements ($N_e = 5 \times 5$). (b) Sparsity pattern of global matrices A and K. (c) Relative residual versus number of iterations for the unpreconditioned (NONE) and preconditioned GMRES (EBSO).*

make the EBFO and EBSO preconditioners basically comparable in form. We have verified that the averaging process in steps 3-5 does not impair the effectiveness of the EBFO preconditioner. GMRES convergence graphs resulting from using an EBFO preconditioner based on steps 1-2 and EBFO preconditioner using steps 1-5 are visually indistinguishable.

5. Results. In Section 5.1 we show how the EBSO preconditioner improves the convergence of GMRES for the simple linear dynamical model on which the structure of the preconditioner was based. We also show why the imaginary component weighting factor w is necessary. In Section 5.2 we show and discuss the results of using the preconditioner to accelerate both GMRES-based and BICGSTAB-based solutions to the fully non-linear and non-hydrostatic Euler Equations.

5.1. Preconditioning the Linear 2-D Conservation Law Model. Here we show how the EBSO preconditioner facilitates the solution of Eq. (4.2) where A and $\boldsymbol{b}$ are as given by Eqs. (3.34) and the previous state vector $\boldsymbol{q}_m$ is a random vector with normally distributed components supplied by MATLAB's randn( ). We employ trapezoidal implicit time differencing ($\alpha = \beta = 1/2$) with the time-step Δt selected so as to produce a Courant Number of 8, which is large enough to result in an unacceptably slow GMRES convergence rate. We also use 4th-order spatial discretization and doubly periodic boundary conditions on a model domain of $(x, y) \in [0, 10]^2$ meters^2.

To construct a local K_I, K_S, K_C matrix set, we set the number of elements to $N_e = 5 \times 5$ (*i.e.*, in a 5-by-5 grid) and specify that the local preconditioner matrices be assigned to the domain elements according to the pattern shown in Fig. 5.1a. We then perform the optimization problem represented by Eq. (4.5) with $p = 8$ and $w = 1$ to obtain local matrices K_I, K_S, K_C. The sparsity patterns for both global matrices A and K appear in Fig. 5.1b. In Fig. 5.1c we provide a comparison of the GMRES convergence rates exhibited for the unpreconditioned (NONE) and preconditioned (EBSO) problems for a single right-side $\boldsymbol{q}_m$ vector. The significantly more rapid and geometric GMRES convergence provided by the EBSO preconditioner is readily apparent.

In Fig. 5.2 we summarize the results of applying the *same* EBSO preconditioner used in Fig. 5.1 to a higher resolution version of the conservation law model in which we have increased $N_e = 10 \times 10$, but have reduced Δt sufficiently to maintain the same Courant Number. By comparing Fig. 5.2a with Fig. 5.1a, the reader can see how the same set of local K_I, K_S, K_C matrices are assigned to the larger number of domain elements in the higher resolution model. The representative examples of the GMRES convergence rates (for one of the randomly selected right-side $\boldsymbol{q}_m$ vectors) that we provide in Fig. 5.2c provide a quick visual indication that the EBSO preconditioner created using a lower resolution version of the model still provides a considerable improvement in the GMRES convergence rate for the higher resolution model. In Table I we provide a statistical summary of how the EBSO preconditioner described above performs in response to changes in Courant No. and number of elements (N_e). By comparing the parenthetical numbers along rows we can see that the reduction in the number of GMRES iterations is relatively insensitive to significant changes in the

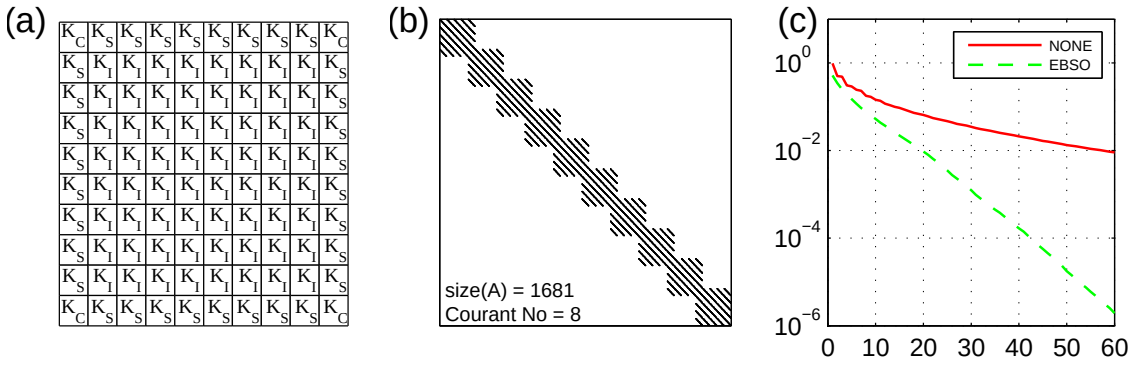

FIGURE 5.2. *As in Fig. 5.1, except for a model domain with* $N_e = 10 \times 10$.

	Courant Number		
N_e and size(A)	8	16	24
5^2 441	57.4 16.3 (3.52)	80.0 22.7 (3.53)	90.3 27.0 (3.35)
10^2 1681	57.8 19.6 (2.95)	82.2 22.3 (3.01)	93.8 32.6 (2.88)
15^2 3721	57.8 20.9 (2.76)	81.7 30.5 (2.68)	93.4 36.6 (2.55)

TABLE I

Average number of unpreconditioned (upper), preconditioned (lower) GMRES iterations and their quotient (in parentheses) for the linear conservation law as a function of number of elements (N_e) and Courant Number. Results are based on a sample size of 20, a stopping relative residual of 10^{-2}*, and the EBSO preconditioner shown in Fig. 5.1.*

Courant Number. However, by comparing the parenthetical numbers along columns we can discern a significant and unsatisfactory reduction in the effectiveness of the EBSO preconditioner in response to relatively modest increases in the size of the system matrix.

To discern the source of the problem just identified, we show in Fig. 5.3 the spectra of the system matrices A and KA corresponding to the Courant No. 8 column of Table I. Notice that the effect of K is to transform the vertically-distributed spectrum of the matrix A in Fig. 5.3a, which is associated with the slow unpreconditioned GMRES convergence rate (Fig. 5.1c; NONE) into the disk-shaped spectrum of KA in Fig. 5.3a, which is confined well within the unit circle, and which is associated with the significantly faster EBSO-preconditioned GMRES convergence rate seen in Fig. 5.1c.

However, notice in Figs. 5.3b and c that as model resolution is increased the spectra of KA develop an increasingly dense, *vertically-oriented* line of eigenvalues that extends outside the unit circle. Conjecturing that this vertical spreading of the spectrum of KA is the source of the degradation of GMRES performance with increasing matrix size, we decided to repeat the process of computing the K_I, K_S, and K_C matrices using a model resolution of $N_e = 5 \times 5$, but this time with the imaginary component weighting factor increased to $w = 10^3$ in Eq. (4.5). This change results in the horizontally-oriented, quasi-elliptical spectrum distribution for the matrix KA shown in Fig. 5.4a.

Now notice that as we increase the number of elements to $N_e = 10 \times 10$ (Fig. 5.4b) and $N_e = 15 \times 15$ (Fig. 5.4c), while again using the same set of local preconditioner matrices K_I, K_S, K_C, the spectrum expands only slightly in the imaginary direction. In Figs. 5.4d-f we show that a similar behavior occurs when the same preconditioner is applied to various model resolutions with the time-step increased to give a Courant No. of 16. Notice that although the spectra extend outside the unit circle in the direction of the positive real axis in Fig. 5.4d-f, nevertheless the shape of the spectra retains the same sort of elliptical character as seen in Fig. 5.4a-c. We note in passing that in Fig.

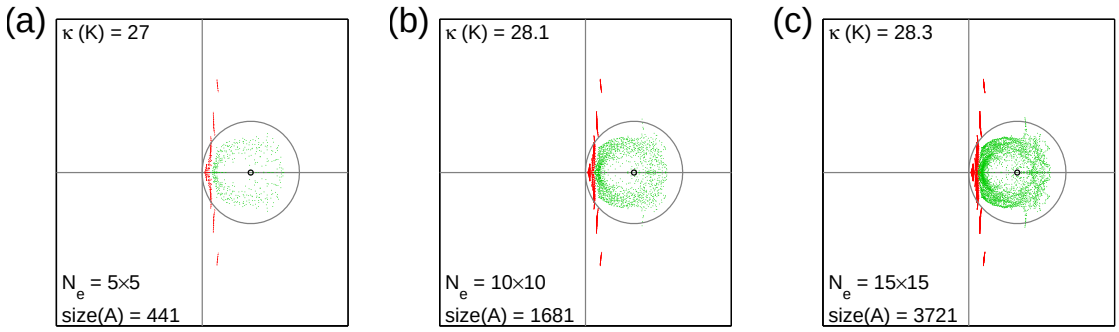

FIGURE 5.3. *Spectra of the global system matrices A (red, vertically-oriented pattern) and KA (green, disk-shaped pattern) resulting from Courant No. of 8 and number of elements shown in the lower left of each panel. The 2-norm condition number (κ) of matrix K appears in the upper left of each panel to verify that the global EBSO left-preconditioner K remains well-conditioned with increasing size.*

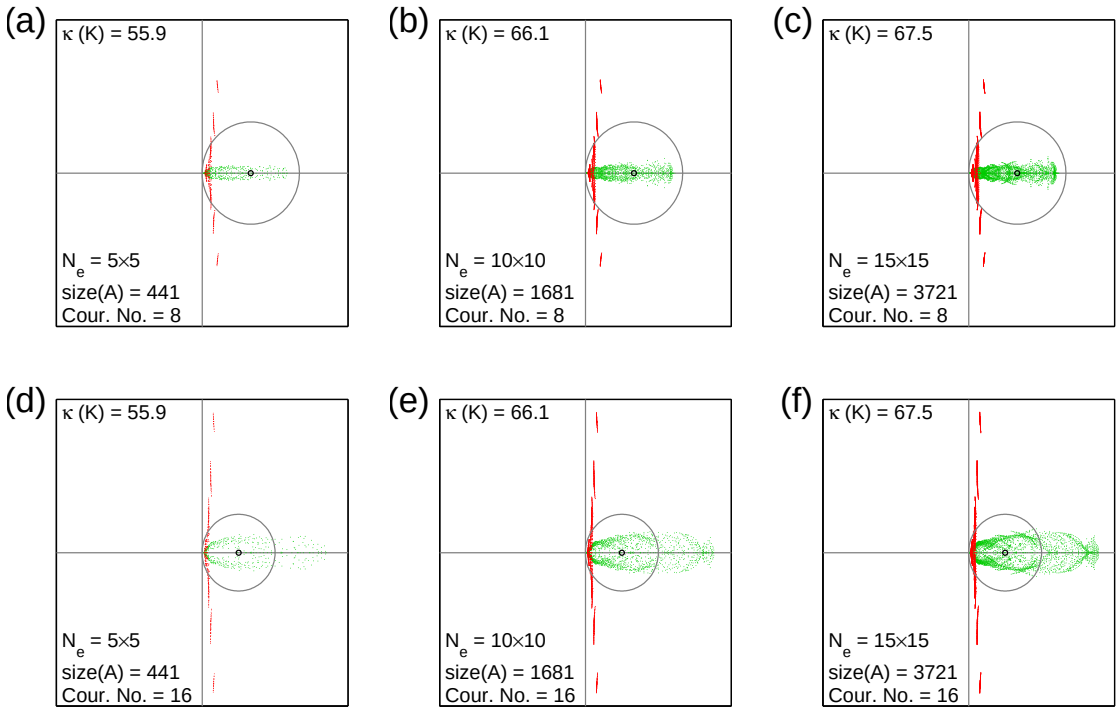

FIGURE 5.4. *Comparison of the spectra of the global system matrices A (red, vertically-oriented pattern) and KA (green, ellipse-shaped pattern) as the number of elements is increased from 5-by-5 to 15-by-15 for a Courant No. of 8 (Panels a-c) and Courant No. of 16 (Panels d-f). See Fig. 5.3 for the meaning of κ(K).*

5.4a-c, for the which the Courant No. is fixed at 8, the distribution of the spectrum of A is visually virtually unchanged in Fig. 5.4a compared to Fig. 5.4c. The insensitivity of the interval over which the spectrum is distributed to increases in system size (model resolution) probably plays a key factor in the insensitivity of the EBSO preconditioner to changes in system size.

In Table II we summarize the performance of the above EBSO preconditioner created via a weighted NLLS optimization process. By comparing the parenthetical numbers along columns we can see that there is now no significant decrease in preconditioner performance as the number of elements is increased. By comparing the parenthetical numbers along rows we can see that there is now actually a tendency for preconditioner performance to improve as the Courant No. is increased.

	Courant Number		
N_e and size(A)	8	16	24
5^2 441	57.5 16.5 (3.48)	79.3 21.4 (3.71)	89.8 24.0 (3.73)
10^2 1681	58.2 17.0 (3.42)	83.0 22.1 (3.75)	94.0 25.1 (3.75)
15^2 3721	58.2 17.3 (3.37)	81.8 22.1 (3.70)	93.1 24.9 (3.74)

TABLE II

As in Table I, except for the EBSO preconditioner described by the text associated with Fig. 5.4.

5.2. Preconditioning the Non-Linear 2-D Euler Equations. Here we show how the EBSO preconditioner facilitates the solution of a particular formulation of the 2-D Euler equations:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \qquad \frac{\partial \theta}{\partial t} + \boldsymbol{u} \cdot \nabla \theta = 0$$
$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \frac{1}{\rho} \nabla P + g\boldsymbol{k} = 0 \qquad P = P_A \left(\frac{\rho R \theta}{P_A} \right)^{\gamma} \tag{5.1}$$

where ρ is the density, θ is the potential temperature, $\boldsymbol{u} = (u, w)^T$ is the velocity field, P is the pressure, P_A is the surface pressure, R is the ideal gas constant, and $\gamma = 1.4$. Giraldo *et al.* [7] present a detailed development of a SE model employing IMEX time integration and utilizing a Schur complement form of Eq. (5.1) so that the perturbation state variables may be obtained in a sequential fashion beginning with the pressure.[8]

The three dynamical test cases (i.e., inertia gravity wave, density current, and mountain wave) employed in [7] to evaluate the performance of the model did not result in a large number of GMRES iterations, and thus did not necessitate preconditioning. Therefore, we will employ a fourth standard test case here that does result in an excessive number of unpreconditioned GMRES iterations; namely, a rising thermal bubble [6]. The problem domain is $(x, z) \in [0, 1000]^2$ meters^2 with reflecting boundary conditions (no flux) on all four sides. The basic state is a motionless air mass that has a potential temperature of 300^oK and that is in hydrostatic balance. The initially motionless bubble has a perturbation potential temperature θ' given by:

$$\theta' = \begin{cases} 0 & \texttt{if } r > r_c \\ \frac{\theta_c}{2} \left[1 + \cos\left(\frac{\pi r}{r_c} \right) \right] & \texttt{if } r \le r_c \end{cases} \qquad r = \sqrt{(x - x_c)^2 + (z - z_c)^2} \tag{5.2}$$

where $\theta_c = 0.5^o$C, $r_c = 250$m, $(x_c, z_c) = (500,350)$. For all the results that follow we employ 5th-order spatial interpolation. We also use the same 10^{-2} relative residual as the stopping criterion for GMRES, and the same 2nd-order backward time differencing scheme (BDF2) and serial computing environment as used in [7].

Figure 5.5 provides a visual comparison of the evolution of the bubble out to 650 seconds for a short time-step reference run (Fig. 5.5a), a large time-step unpreconditioned run (Fig. 5.5b) and large time-step EBSO preconditioned runs (Fig. 5.5c). Notice that the warmest potential temperature perturbation found anywhere (T_{max}) at $t = 650s$ in the two large time-step runs is the same to 6 decimal places, which indicates that the EBSO preconditioner introduces no significant error to the GMRES computations. That T_{max} in the two large time-step runs varies from the short time-step reference run by 1 percent is consistent with the 10^{-2} relative residual stopping criterion for GMRES, and confirms that the BDF2 time-differencing scheme can handle the very long time-step (≈ 100 times the maximum explicit time-step) associated with a model run at Courant No. = 28.

In order to construct the EBSO preconditioner employed in the rising bubble problem, we again used $N_e = 5 \times 5$, $p = 8$, and $w = 10^3$ based on the results in Section 5.1. However, this time we set the Courant Number at 18 to obtain a similar unpreconditioned GMRES convergence rate as in

[8] For more details, see the portion of [7] dealing with Eq. set (5.1), which they term SE2NC.

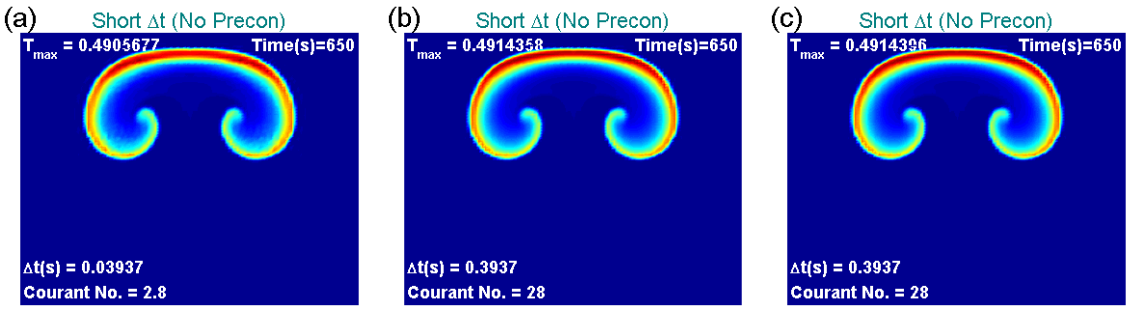

FIGURE 5.5. *Perturbation potential temperature fields at 650s from rising thermal bubble problem using a small Δt (panel a), a large Δt and no preconditioner (panel b), and the same large Δt and the EBSO preconditioner. For all runs $N_e = 40^2$, and the T_{max} number in the upper left of each figure gives the warmest temperature at any node in the domain. The Δt and associated Courant No. appear in the lower left of each panel.*

Section 5.1. Using the same Courant No., we also constructed an EBFO preconditioner using the procedure described in Section 4.2, but using a higher resolution model with $N_e = 10 \times 10$ since the preconditioner is easy to compute for the larger system, and to improve its competitiveness with EBSO. For a model size of $N_e = 10 \times 10$, the global sparsity pattern of both preconditioners are shown in Fig. 5.6a, and that of the system matrix A is illustrated in Fig. 5.6b.

Notice that in contrast to the linear conservation law problem, here the bandwidth of the Schur form system matrix (Fig. 5.6b) is precisely *twice* that of the global preconditioner (Fig. 5.6a), regardless of the size of the problem. This difference exists because the Schur form involves a product of discretized DSS-based operators (i.e., divergence $\times$ gradient). Despite the narrower bandwidth, the preconditioner K has 1.22 times as many non-zero entries as the matrix A, which is explained by the fact that the EBSO preconditioner employs the same relatively dense sparsity pattern that would arise from computing a mass matrix using *exact* quadrature, whereas the Schur form sparsity pattern is relatively less dense due to the use of *inexact* (collocated) quadrature. Even though the preconditioner K is slightly less sparse than the system matrix A, timing tests have shown that the cost in CPU time for applying K is approximately half the cost of applying A at each time-step. This difference is explained by the fact that whereas K employs the DSS operator only once, the Schur form A must use the DSS process twice, and must include within each DSS operation multiple additional loops associated with the collocated quadrature that occurs at each time-step.

The spectra of the PSM's KA when K is the EBSO preconditioner and EBFO preconditioner are shown in Fig. 5.6c and e, respectively. Notice that whereas the spectrum of the system matrix A is real and spread over the interval $[1, 1082]$ (see Fig. 5.6b lower left), the spectra of the global KA matrices are complex and confined within the unit circle. The KA spectrum due to the EBSO preconditioner (Fig. 5.6c) is centered within the circle as in Fig. 5.4a, whereas the KA spectrum due to the EBFO preconditioner (Fig. 5.6e) has a dense cluster of eigenvalues centered on unity and a line of eigenvalues trailing off toward the origin. Notice that based on the condition numbers shown in 5.6c and e, both preconditioners are well-conditioned, as required to justify the use of a left-preconditioner as discussed in Section 1.1.

The impact of the EBSO and EBFO preconditioners on the GMRES convergence rate for one representative time-step early in the evolution of the rising bubble problem for $N_e = 10 \times 10$ is shown in Fig. 5.6d and f, respectively. The relative residual graphs for EBSO-preconditioned GMRES and EBFO-preconditioned GMRES[9] both reach 10^{-2} at about 25 iterations, but thereafter the EBSO graph descends more quickly. A relative residual graph for a more traditional FNO preconditioner constructed using the sparsity pattern of A (Fig. 5.6b) has been included in Fig. 5.6f (FNO) to show that the EBFO preconditioner is competitive out to 60 iterations, and actually reaches a residual of 10^{-2} faster, despite the fact that the EBFO preconditioner K has half the bandwidth of the FNO

[9]The curve labeled EBFO represents the EBFO preconditioner with and without the averaging process discussed in Section 4.2. The two curves are visually indistinguishable.

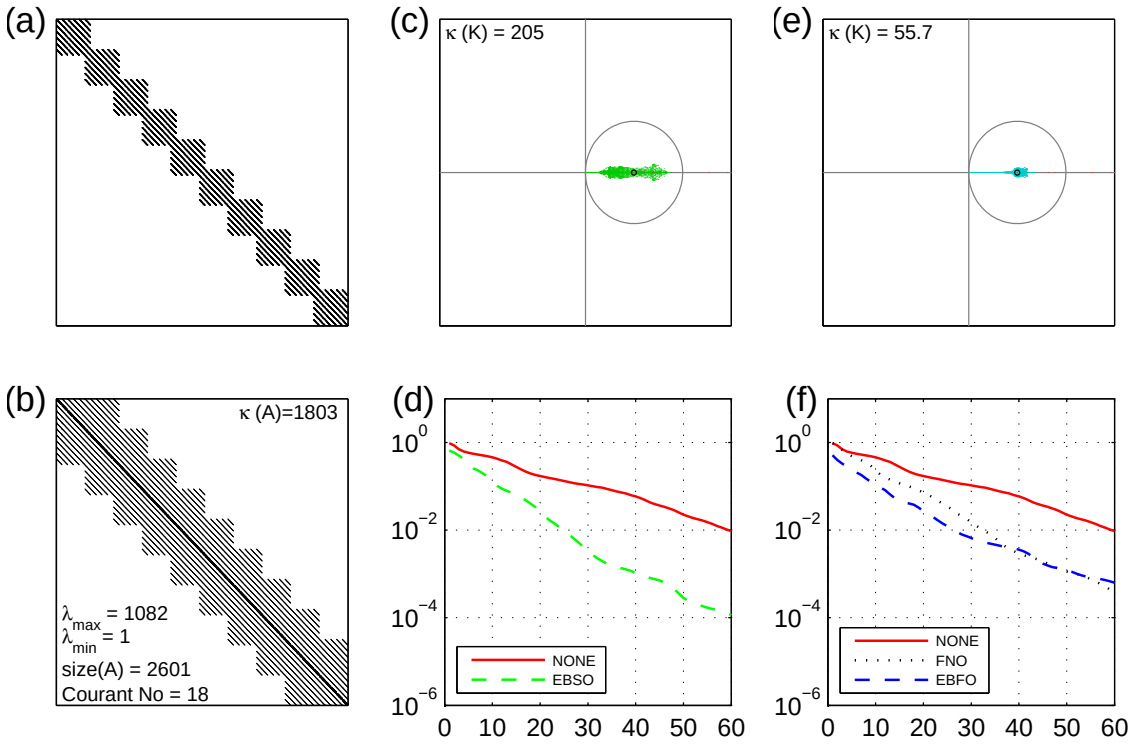

FIGURE 5.6. *(a) Sparsity pattern for both the EBSO and EBFO preconditioners for the rising bubble problem model using $N_e = 10 \times 10$ and 5th order spatial interpolation. (b) As in (a), except for the Schur form global system matrix A. (c) The spectrum of preconditioned system matrix KA when K is the EBSO preconditioner with the 2-norm condition number $\kappa(K)$ shown. (d) Representative examples of GMRES convergence trends without a preconditioner (NONE) and with the EBSO preconditioner (EBSO). (e) As in (c), except for an EBFO preconditioner (EBFO) based on the sparsity pattern shown in (a). (f) As in (d), except for an EBFO preconditioner (EBFO) and a traditional FNO preconditioner (FNO) based the system matrix sparsity pattern in (b).*

preconditioner.

The results of using both GMRES and BICGSTAB to run a comparatively low-resolution model ($N_e = 25 \times 25$) of the the rising bubble problem for 100 seconds of model time using various combinations of Courant No. and preconditioning are summarized in Fig. 5.7. By comparing the upper panels it is evident that BICGSTAB requires substantially fewer iterations than GMRES at all Courant Numbers, and regardless of which iterative method is considered all the preconditioners achieve a substantial reduction in the number of iterations. If we focus on the larger Courant numbers (where IMEX models tend to run), then we can see that the iteration-reducing power of the EBSO preconditioner (inherently first order) falls in between the 1st and 2nd order Chebyshev polynomial preconditioner if either GMRES or BIGSTAB is employed. The EBFO preconditioner performs the poorest, except for Courant No. 32 where it is better then CHEB1.

By comparing the lower panels of Fig. 5.7, we can immediately see that for low values of Courant No. all the preconditioners slow the model down, regardless of the iteration method used, because the number of iterations per time-step is simply too small to justify preconditioning. As Courant No. increases, we see a somewhat more complex relationship between GMRES and BICGSTAB with regard to wall-clock time owing to their relative efficiency in applying the system matrix and performing dot-products. Wall-clock time for unpreconditioned GMRES falls and then rises with increasing Courant No. due to a non-linear growth in the number of dot-products per iteration. By contrast, unpreconditioned BICGSTAB wall-clock time steadily decreases with increasing Courant No. due to only a linear growth in the number of dot-products per iterations. Preconditioning GMRES with CHEB1 and CHEB2 avoids the slow-down with increasing Courant No., but preconditioning BICGSTAB with CHEB1 and CHEB2 moderately degrades wall-clock time performance at all Courant Num-

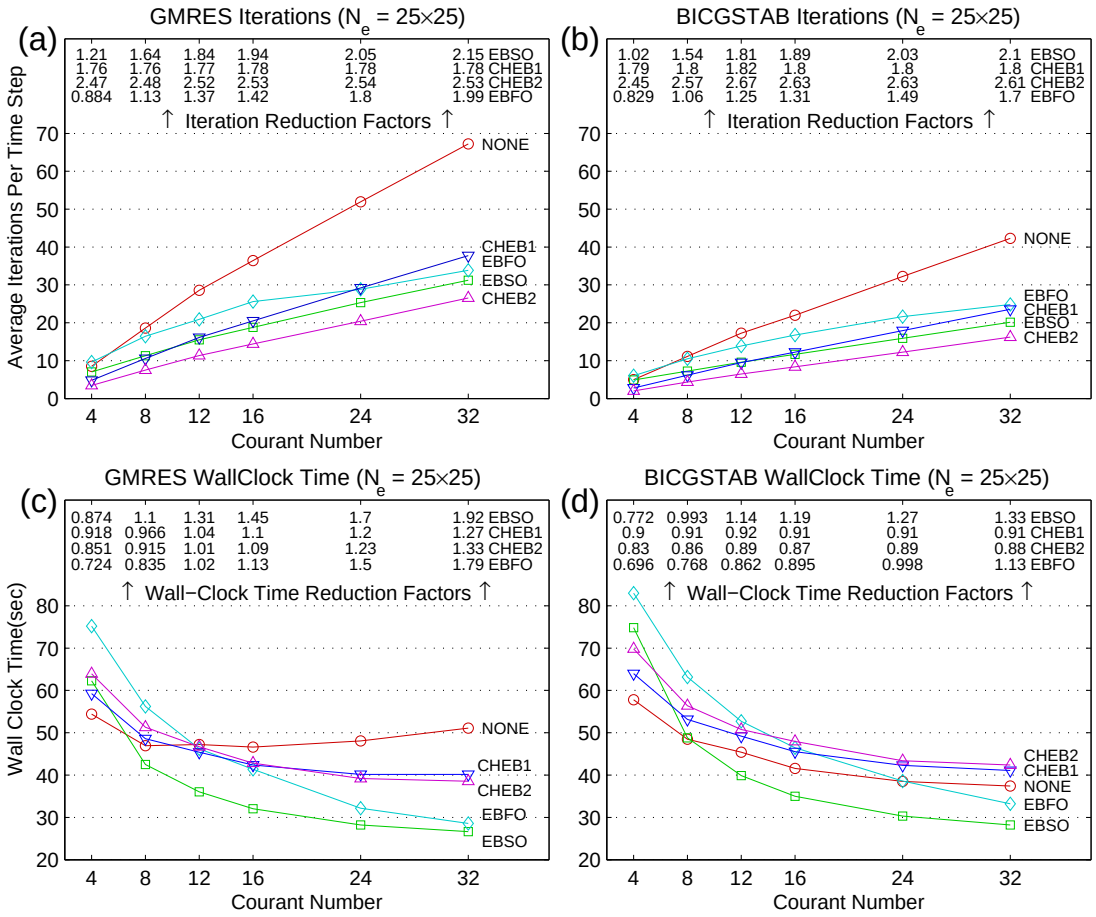

FIGURE 5.7. *A comparison of unpreconditioned (NONE) and preconditioned GMRES and BICGSTAB iterations per time-step (upper panels) and wall-clock time (lower panels) for a rising bubble problem integrated for 100s at Courant Numbers ranging from 4 to 32 using 5th order spatial interpolation and* $N_e = 25 \times 252$ *resulting in size(A) = 15,876. Preconditioners included are 1st and 2nd order Chebyshev polynomials (CHEB1, CHEB2), the element-based spectrally-optimized preconditioner (EBSO), and the element-based Frobenius-norm-optimized preconditioner (EBFO). Tabular data at the top of upper panels give the ratio of unpreconditioned to preconditioned iterations for the preconditioners listed. Tabular data at the top of lower panels give the ratio of unpreconditioned to preconditioned wall-clock time for the preconditioners listed.*

bers. Due to its low application cost, the EBSO preconditioner results in a significant reduction in wall-clock time for Courant Nos. greater than 8 using either iterative scheme, and the amount of model acceleration rises to 1.92 for GMRES and 1.33 for BICGSTAB at Courant No. 32. Notice also that EBSO-preconditioned GMRES runs slightly faster than EBSO-preconditioned BICGSTAB at Courant No. 32, due to the combined effect of the significant reduction in preconditioned-GMRES dot-products and the fact that BICGSTAB must use the system matrix twice per iteration. Thus, the model acceleration factor when comparing unpreconditioned BICGSTAB (fastest unpreconditioned algorithm) and EBSO-preconditioned GMRES (fastest preconditioned method) is actually 1.41. Interestingly the EBFO preconditioner, which performs the poorest for lower Courant Nos., actually becomes competitive with EBSO above Courant No. 24.

To see if the results shown in Fig. 5.7 will scale with increasing model resolution, we repeated all the model runs at double the resolution (Fig. 5.8, see caption concerning EBFO). It is immediately clear that the iteration results in the upper panels of Fig. 5.8 are quantitatively very similar to the results in Fig. 5.7. The fact that the unpreconditioned iterations are insensitive to changes in problem size indicates that fixing the Courant No. creates a resolution-independent scalable iterative problem *at each time-step*. The entries in the tables of ratios of unpreconditioned to preconditioned

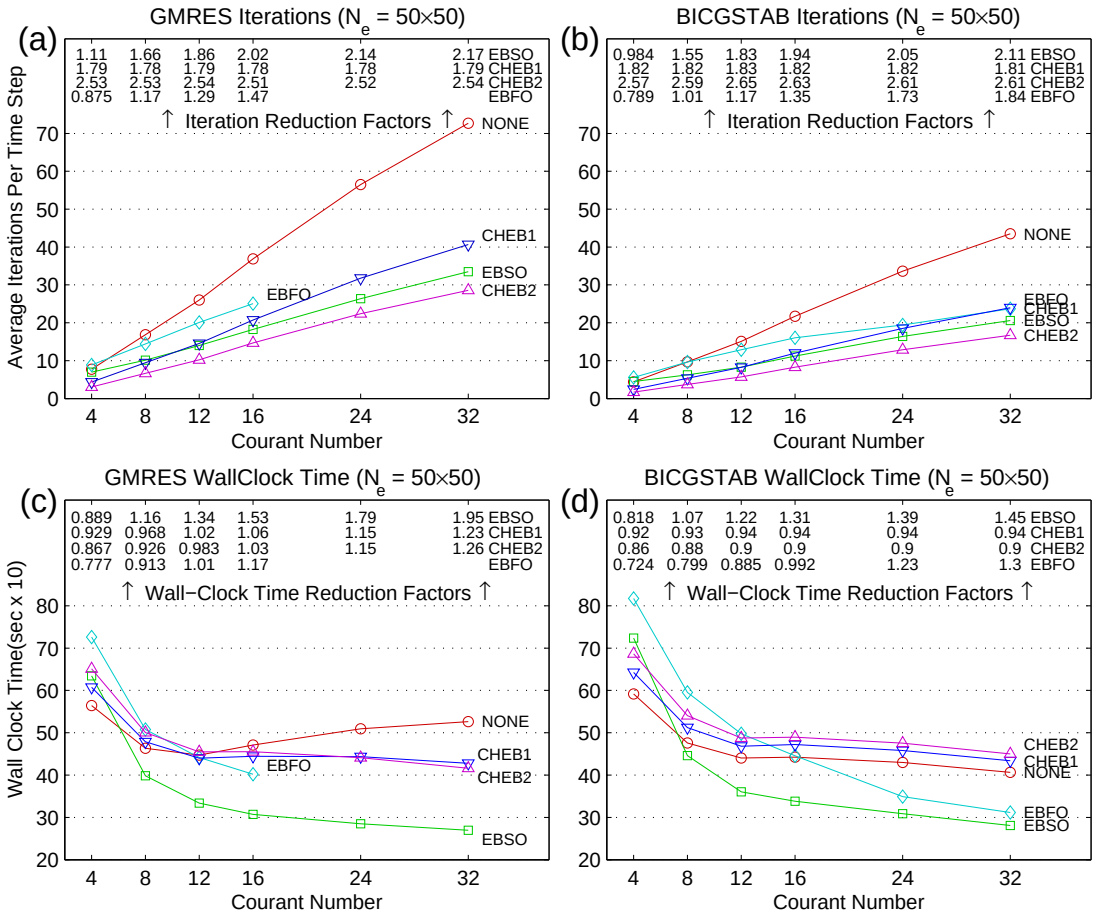

FIGURE 5.8. *As in Fig. 5.7, except for* $N_e = 50 \times 50$ *resulting in size(A) = 63,001. The premature termination of the graphs for EBFO-preconditioned GMRES reflect the fact that the model would not run to completion unless the relative residual stopping criterion was reduced from* 10^{-2} *to* 10^{-3}*. Thus, EBFO is a less robust preconditioner when used with GMRES.*

iterations provided at the top of Figs. 5.7 and 5.8 indicate that the performance of the SE-based preconditioners is independent of model scale.

Although, the wall-clock time results in Fig. 5.8c-d exhibit basically the same *relative* performance patterns as in Fig. 5.7c-d, notice that the quantitative values in Fig. 5.8c-d are approximately *ten times larger* than those in 5.7c-d. This behavior is explained as follows. Fixing the Courant No. means that the time-step must be proportionally reduced as resolution is increased, which in turn means that that the number of time-steps must be increased by the same proportion. As a result, the total number of iterations doubles with every doubling of model resolution, despite the fact that the number of iterations per time-step can be held constant.

Doubling model resolution again using $N_e = 100 \times 100$ resulting in size$(A) = 251{,}001$ produced results (not shown) that were again qualitatively similar to Figs. 5.7 and 5.8, other than the fact that wall-clock times again increased by a factor of about ten. However, we note an interesting trend with regard to EBSO wall-clock time reduction at Courant No. 32. Recall that we previously noted that the ratio of unpreconditioned BICGSTAB to EBSO-preconditioned GMRES wall-clock time in Fig. 5.7c-d was 1.41 for a problem with $N_e = 25 \times 25$. This ratio increases to 1.51 for a higher resolution problem with $N_e = 50 \times 50$, and further increases to 1.60 for a still higher resolution problem with $N_e = 100 \times 100$.

A final very promising result is shown in Fig. 5.9, which: i) repeats the GMRES iteration and wall-clock time curves from Fig. 5.7a and c (with EBFO omitted), and ii) adds the curves for

preconditioners EBSO1 and EBSO2 which are 1st and 2nd order Chebyshev polynomials in which $s(A)$ is replaced by $s(KA)$, where K is the EBSO preconditioner. At Courant No. 32, the EBSO1 and EBSO2 preconditioners achieve up to a 5-fold reduction in GMRES iterations (Fig. 5.9a) while retaining most of the wall-clock time reduction power of the EBSO preconditioner (Fig. 5.9b). The EBSO2 preconditioner reduces the number of iterations so greatly that the number of dot-products in GMRES per time-step is actually at or below the number in BICGSTAB (12 iterations is the break even point).

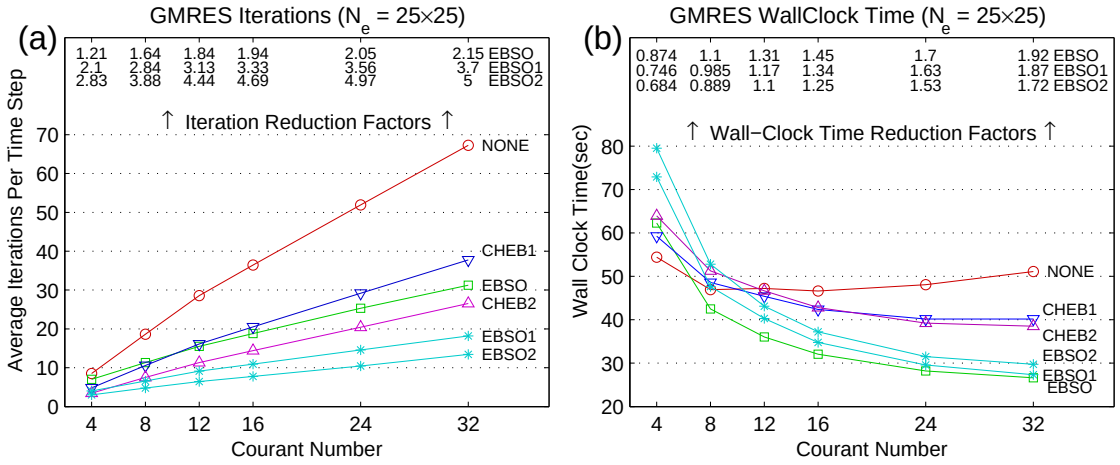

FIGURE 5.9. *As in Fig. 5.7a and c, except that the EBFO preconditioner plots have been omitted, and plots for the EBSO1 and EBSO2 preconditioners have been added.*

6. Summary and Concluding Remarks. We have provided an initial development and demonstration of a preconditioning method for accelerating the iterative solution of the system $A\boldsymbol{x} = \boldsymbol{b}_i$ that arises at the ith time-step in any spectral element method (SE) fluid dynamics model that employs IMEX time integration. By means of this preconditioning method the large and sparse global approximate inverse preconditioner K in the equivalent left-preconditioned system

$$(KA)\boldsymbol{x} = K\boldsymbol{b}_i \tag{6.1}$$

is effectively assembled from a set of a few small and full local matrices whose entries result from an optimization problem that seeks to make the matrix KA the best approximation of I in an eigenvalue-by-eigenvalue sense (i.e., spectrally) rather than in an entry-by-entry sense as in the Frobenius norm method. Thus, we have assigned the preconditioning method the descriptor "element-based, spectrum-optimized" (EBSO).

Using both a 2-D linear conservation law and the 2-D fully compressible, non-linear Euler equations, we have demonstrated that after the local matrices of the EBSO preconditioner are created using a low resolution version of the applicable SE model, the preconditioner then may be applied to arbitrarily high-resolution versions of the same model without appreciable loss of preconditioner performance. In a test case in which the Euler equations are used to model a rising thermal bubble, the number of iterations per time-step is cut in half for Courant numbers of 16 and greater regardless of whether GMRES and BICGSTAB is used, and the model wall-clock time is significantly reduced using either iterative method. The wall-clock time reducing power of the EBSO preconditioner is significantly better than either conventional low-order Chebyshev polynomials or an element-based Frobenius norm optimized (EBFO) preconditioner specifically developed herein for the purpose of a fair head-to-head comparison with the EBSO preconditioner. For large Courant No., the wall-clock time reduction factor for the EBSO preconditioner was found to be as large as 2.1 for GMRES and 1.45 for BICGSTAB. Importantly, when the EBSO and Chebyshev polynomial methods are combined up to a 5-fold reduction in iterations is achieved while still achieving significant reductions in wall-clock times.

All results presented herein have been accomplished using relatively small problem sizes to facilitate completion of the extremely large number of model runs required to do all the comparative analysis in a serial SE computing environment. However, the performance of the EBSO preconditioner and comparison preconditioners was found to be independent of problem size, and the relative performance of the preconditioners remained the same as problem size increased. Moreover, since the EBSO preconditioner and comparison preconditioners presented herein are all explicit and use exactly the same spectral element approach to accomplishing matrix-vector products, they are all equally amenable to implementation in a massively parallel SE computing environment. Thus, they are all highly parallelizable and their relative performances should remain the same, which means the EBSO (and EBSO1 and EBSO2) preconditioner should continue to produce substantially better reductions in iterations per time-step and model wall-clock time. Although the EBSO preconditioner does have a relatively high construction cost, the eventual target application is operational weather prediction models that run for years without modification, and involve hundreds to thousands of time-steps during each run of the model, and thus making high construction cost of little consequence.

Our future work will include incorporating the EBSO and comparison preconditioners into the IMEX version of the 3-D NUMA model that is currently under development for use within a massively parallel computing environment. We will also look into extending the EBSO methodology to include variable domain elements sizes and geometries.

Acknowledgments. The authors gratefully acknowledge the support of the Computational Mathematics program of the Air Force Office of Scientific Research. F.X.G also gratefully acknowledges the support of the Office of Naval Research through program element PE-0602435N. We also thank J.F. Kelly and two anonymous reviewers for providing thorough reviews of the draft manuscript and offering many helpful suggestions.

REFERENCES

[1] C.E. Augarde, A. Ramage, and J. Stauchacher, *An element-based displacement preconditioner for linear elasticity problems*, Computers and Structures, 84 (2006) pp. 2306-2315.
[2] M. BENZI, *Preconditioning Techniques for Large Linear Systems: A Survey*, J. Comput. Phys., 182 (2002), 418-477.
[3] M. BENZI, J.C. Haws, and M. Tuma, *Preconditioning Highly Indefinite and Nonsymmetric Matrices*, SIAM J. Sci. Comput., 22 (2000) pp. 1333-1353.
[4] M. BENZI and M. Tuma, *A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems*, SIAM J. Sci. Comput., 19 (1998) pp. 968-994.
[5] M. BENZI and M. Tuma, *A Comparative Study of Sparse Approximate Inverse Preconditioners*, Appl. Numer. Math., 30 (1999) pp. 305-340.
[6] F.X. GIRALDO and M. Restelli, *A Study of Spectral Element and Discontinuous Galerkin Methods for the Navier-Stokes Equations in Nonhydrostatic Mesoscale Atmospheric Modeling: Equation Sets and Test Cases*, J. Comp. Phys., 227 (2008), pp. 3849-3877.
[7] F.X. GIRALDO, M. Restelli, and M. Lauter, *Semi-Implicit Formulations of the Navier-Stokes Equations: Applications to Nonhydrostatic Atmospheric Modeling*, SIAM J. Sci. Comput., 32 (2010), pp. 3394-3425.
[8] M.J. GROTE and T. Huckle, *Parallel preconditionings with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838-853.
[9] J. F. KELLY and F.X. GIRALDO, *Continuous and Discontinuous Galerkin Methods for a Scalable 3D Nonhydrostatic Atmospheric Model: Limited Area Mode*, J. Comp. Phys., in review.
[10] Y. LIANG, *Generalized Least-Squares Polynomial Preconditioners for Symmetric Indefinite Linear Equations*, Parallel Comput., 28 (2002), 323-341.
[11] Y. LIANG, *Polynomial Preconditioner for the Solution of Linear Equations*, Ph.D. dissertation, University of Ulster, 2005.
[12] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia PA, 2003.
[13] Y. SAAD and M.H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.
[14] L.N. TREFETHEN and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia PA, 1997.
[15] P. S. VASSILEVSKI, *Multilevel Block Factorization Preconditioners*, Springer, New York NY, 2008.
[16] H.A. van der VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631-644.
[17] H.A. van der VORST, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, New York NY, 2003.