A POLYNOMIAL-BASED NONLINEAR LEAST SQUARES OPTIMIZED PRECONDITIONER FOR CONTINUOUS AND DISCONTINUOUS ELEMENT-BASED DISCRETIZATIONS OF THE EULER EQUATIONS

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

Abstract. We introduce a method for constructing a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm. We show that this polynomial-based NLLS-optimized (PBNO) preconditioner significantly improves the performance of 2-D continuous Galerkin (CG) and discontinuous Galerkin (DG) fluid dynamical research models when run in an implicit-explicit time integration mode. When employed in a serially computed Schur-complement form of the 2-D CG model with positive definite spectrum, the PBNO preconditioner achieves greater reductions in GMRES iterations and model wall-clock time compared to the analogous linear least-squares-derived Chebyshev polynomial preconditioner. Whereas constructing a Chebyshev preconditioner to handle the complex spectrum of the DG model would introduce an element of arbitrariness in selecting the appropriate convex hull, construction of a PBNO preconditioner for the 2-D DG model utilizes precisely the same objective NLLS algorithm as for the CG model. As in the CG model, the PBNO preconditioner achieves significant reduction in GMRES iteration counts and model wall-clock time. Comparisons of the ability of the PBNO preconditioner to improve CG and DG model performance when employing the Stabilized Biconjugate Gradient algorithm (BICGS) and the basic Richardson (RICH) iteration are also included. In particular, we show that higher order PBNO preconditioning of the Richardson iteration (which is run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment, especially when employed in a DG model. Because the NLLS-based algorithm used to construct the PBNO preconditioner can handle both positive definite and complex spectra without any need for algorithm modification, we suggest that the PBNO preconditioner is, for certain types of problems, an attractive alternative to existing polynomial preconditioners based on linear least-squares methods.

Key words. preconditioning; polynomial preconditioner; nonlinear least squares; element-based; spectral elements; Galerkin methods; Euler Equations; nonhydrostatic atmospheric model

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

1. Introduction.

1.1. Background. In a previous paper [1] we described the development of an element-based spectrally-optimized (EBSO) preconditioner designed to increase the efficiency of an implicit-explicit (IMEX) 2-D CG model of the Euler equations in Schur-complement form. The usefulness of the EBSO preconditioner was demonstrated using a rising thermal bubble (RTB) test case. The ultimate goal of that previous work, as well as the PBNO polynomial preconditioning technique developed herein, is eventual incorporation into CG and DG variants of the Nonhydrostatic Unified Model of the Atmosphere (NUMA) that are currently under development for both regional and global domains [3, 5, 4].

1.2. Paper Format. Section 2 provides an overview of the fluid dynamical modeling context within which a preconditioner must function effectively in order to be useful for our purposes. This section also includes an overview of the iterative solvers that we are currently evaluating in terms of their response to the PBNO preconditioner and several comparison preconditioners. Section 3 provides some background on polynomial preconditioners and identifies two existing preconditioners against which we will compare the PBNO preconditioner when feasible. Section 4 lays out the mathematical development of the NLLS-based method for constructing the PBNO preconditioner. Section 5 presents the results of PBNO preconditioner performance in CG and DG model variants of the 2-D RTB problem in a serial computing environment. Section 6 summarizes the key results of the paper and provides an outline of planned future work.

2. Modeling Context.

The combination of the high-resolution associated with nonhydrostatic atmospheric modeling and the large domain size associated with global models 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

$$A\boldsymbol{q}^{n+1} = R(\boldsymbol{q}^n), \tag{2.1}$$

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

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2014		**00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Polynomial-Based Nonlinear Least Squares Optimized Preconditioner for Continuous and Discontinuous Element-Based Discretizations of the Euler Equations	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Applied Mathematics,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
SIAM Journal on Scientific Computing (in review 2014)

14. ABSTRACT

We introduce a method for constructing a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm. We show that this polynomial-based NLLS-optimized (PBNO) preconditioner signi cantly improves the performance of 2-D continuous Galerkin (CG) and discontinuous Galerkin (DG) uid dynamical research models when run in an implicit-explicit time integration mode. When employed in a serially computed Schurcomplement form of the 2-D CG model with positive de nite spectrum, the PBNO preconditioner achieves greater reductions in GMRES iterations and model wall-clock time compared to the analogous linear least-squares-derived Chebyshev polynomial preconditioner. Whereas constructing a Chebyshev preconditioner to handle the complex spectrum of the DG model would introduce an element of arbitrariness in selecting the appropriate convex hull construction of a PBNO preconditioner for the 2-D DG model utilizes precisely the same objective NLLS algorithm as for the CG model. As in the CG model, the PBNO preconditioner achieves signi cant reduction in GMRES iteration counts and model wall-clock time. Comparisons of the ability of the PBNO preconditioner to improve CG and DG model performance when employing the Stabilized Biconjugate Gradient algorithm (BICGS) and the basic Richardson (RICH) iteration are also included. In particular, we show that higher order PBNO preconditioning of the Richardson iteration (which is run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment, especially when employed in a DG model. Because the NLLS-based algorithm used to construct the PBNO preconditioner can handle both positive de nite and complex spectra without any need for algorithm modi cation, we suggest that the PBNO preconditioner is, for certain types of problems, an attractive alternative to existing polynomial preconditioners based on linear least-squares methods.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**21**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

or, using standard generic notation

$$A\boldsymbol{x} = \boldsymbol{b}. \tag{2.2}$$

In Eq. (2.1), $\boldsymbol{q}$ is the state vector, R is a right-hand side operator, and the matrix A is square, invertible, nonsymmetric, and fixed (unless adaptive mesh refinement is used). As discussed in [3, 5], if a Schur-complement form is derived for A, then the size of A is $N_g \times N_g$, but if A is left in the non-Schur-complement form, then the size of A is $4N_g \times 4N_g$ for 2-D modeling and $5N_g \times 5N_g$ for 3-D modeling.

To facilitate subsequent analysis and comparison of results it is useful to define the scaling parameter

$$\lambda_{mid} = \frac{|\lambda_{max}| + |\lambda_{min}|}{2}, \tag{2.3}$$

where λ_{max} and λ_{min} are the eigenvalues of system matrix A with the largest and smallest moduli, and which can be well-approximated via the Arnoldi method for a system of any size. Dividing both sides of Eq. 2.2 by λ_{mid} results in the equivalent system

$$(A/\lambda_{mid})\boldsymbol{x} = (\boldsymbol{b}/\lambda_{mid}), \tag{2.4}$$

which we will hereafter denote more concisely by

$$\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}. \tag{2.5}$$

For a CG model of the RTB problem employing Schur-complement form, the spectrum of $\tilde{A}$ is real and fully contained within the unit disk centered at (1,0) as shown in Fig. 2.1(a). For a DG model of the same RTB problem (for which a Schur-complement form is not presently available), the spectrum of $\tilde{A}$ is complex, confined to the right half of the complex plane, and largely contained within the unit disk Fig. 2.1(b).

Despite the need to solve Eq. (2.5) iteratively, the attraction of IMEX methods is that, under suitable dynamical circumstances, one may employ time-steps that are as much as 100 times (or more) greater than the maximum allowable explicit time-step. As a result, IMEX-based models can actually run faster than explicit models provided that the number of iterations needed to solve Eq. (2.5) is kept under control by a sufficiently effective preconditioner. The RTB test case is an example of a dynamical scenario that can be run at a large Courant Number in an IMEX model, and thus permits a large time-step. In [1] we described this test case and used it to show the ability of the EBSO preconditioner to accelerate the iterative solution of Eq. (2.5) when $\tilde{A}$ is in Schur-complement form. Herein we will use the same test case to illustrate the ability of the PBNO preconditioner to solve Eq. (2.5) more efficiently regardless of whether $\tilde{A}$ is Schur-complement form or not.

Because matrix $\tilde{A}$ is not symmetric positive definite (SPD) regardless of whether it is in Schur-complement or not, our selection of suitable iterative solvers is limited to those that can handle non-SPD systems. In [1] we compared the performance of GMRES [10] based on the Arnoldi process [9, p. 165] and transpose-free BICGS [12] based on the Lanczos process [9, p. 234]. In this paper we also include the RICH algorithm [12, p. 22]. All three of these algorithms will be evaluated on the basis of their PBNO-preconditioned performance in a serial computing environment. The rationale in selecting these three solvers is that, as a set, they involve diverse combinations of computational and communication costs. For example, GMRES applies the system matrix once per iteration, but if n is the number of iterations required for convergence, then the number of dot-products performed by GMRES is $n^2/2$. By contrast, transpose-free BICGS applies the system matrix twice, but only requires $6n$ dot-products. Like GMRES, RICH applies the system matrix once per iteration. However, unlike either GMRES or BICGS, RICH can be run in a dot-product-free mode, which can be potentially advantageous when using a massively-parallel computing environment, which will be the focus of a future paper. Appendix A describes the process by which the RICH algorithm can be run in a dot-product-free mode.

If a linear system is to be solved only once, then the cost to construct a preconditioner prior to its employment in the iterative solution process must be kept small. However, in many IMEX applications matrix $\tilde{A}$ in Eq. (2.5) remains the same for hundreds to thousands of time-steps. Moreover,

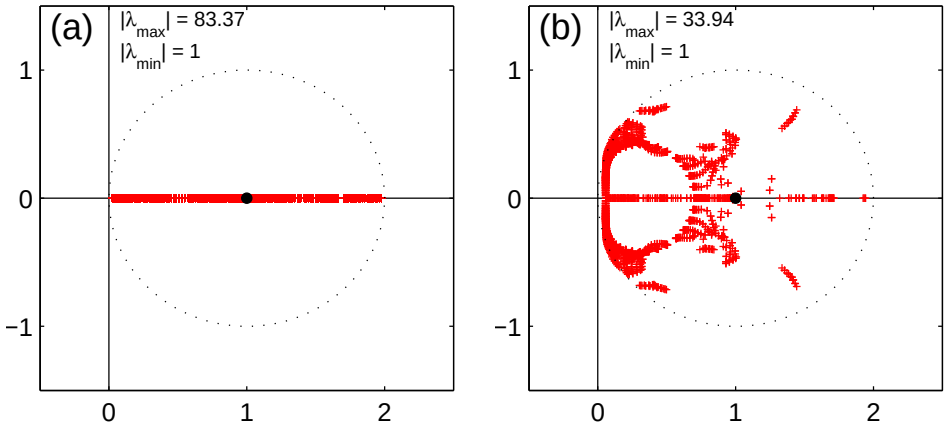

FIGURE 2.1. *(a) Example of a positive-definite spectrum associated with a Schur-complement form of the rescaled system matrix $\tilde{A}$ for a 2-D CG model of the RTB test case. The model had a coarse spatial resolution consisting of 5-by-5 elements and 5th order LGL polynomials, and employed a 2-stage ARK time-differencing scheme with the time-step set to produce a Courant No. of 16. (b) Example of a complex spectrum for a matrix $\tilde{A}$ associated with a 2-D DG model of the RTB test case using the same spatial and temporal discretization schemes as in (a). The eigenvalues of the unscaled system matrix A having the largest and smallest modulus, respectively, appear in the upper left of each panel.*

in an operational numerical weather prediction setting, matrix $\tilde{A}$ remains unchanged for typically many thousands of model runs. In such situations the cost to construct the preconditioner becomes comparatively unimportant, and as a result, it becomes feasible to construct the preconditioner using computationally more expensive methods. In [1] we showed how a NLLS algorithm could be used to construct the EBSO preconditioner in a process that required approximately an hour of computing time on a typical desktop PC. In this paper we show that a similar NLLS algorithm can be used to construct the PBNO preconditioner for the RTB test case much more efficiently even if matrix $\tilde{A}$ is not in Schur-complement form.

3. Polynomial Preconditioning Background. Polynomial preconditioning is a well-known (see [7] for a thorough discussion) type of explicit preconditioning in which we replace Eq.(2.5) by the equivalent system (here using left-preconditioning)

$$(K\tilde{A})\boldsymbol{x} = K\tilde{\boldsymbol{b}}, \tag{3.1}$$

where K is a low-order polynomial in $\tilde{A}$ given by

$$K = s(\tilde{A}) = \sum_{i=0}^{m} k_i \tilde{A}^i, \tag{3.2}$$

and has a spectrum which approximates that of $\tilde{A}^{-1}$. From a CG or DG modeling perspective, an attractive feature of making K a polynomial in $\tilde{A}$ is that K possesses the same degree of parallelization potential as the system matrix, and thus requires no additional preconditioner-specific parallelization machinery whatsoever.

The simplest polynomial preconditioner is the Neumann preconditioner

$$s_N(\tilde{A}) = \sum_{i=0}^{m} (I - \tilde{A})^i, \tag{3.3}$$

which is based on the fact that the right-hand side of Eq. (3.3) must converge to $\tilde{A}^{-1}$ as $m \to \infty$ as long as the spectrum of $\tilde{A}$ is contained within the open unit disk centered at (1,0). Since Neumann preconditioners contain no user-specified parameters, they may thought of as a zero-skill benchmark

Order	k_0	k_1	k_2	k_3	k_4	k_5
$m=1$	2	-1				
$m=2$	3	-3	1			
$m=3$	4	-6	4	-1		
$m=4$	5	-10	10	-5	1	
$m=5$	6	-15	20	-15	6	-1

TABLE I

Coefficient values in Eq. (3.2) for the first five Neumann polynomial preconditioners.

Order	k_0	k_1	k_2	k_3	k_4	k_5
$m=1$	5	-2				
$m=2$	14	-14	4			
$m=3$	30	-54	36	-8		
$m=4$	55	-154	176	-88	16	
$m=5$	91	-364	624	-520	208	-32

TABLE II

Coefficient values in Eq. (3.2) for the first five Chebyshev polynomial preconditioners when the spectrum of the scaled system matrix $\tilde{A}$ is positive definite and lies within the interval $[0, 2]$.

against which other polynomial preconditioners may be compared. The coefficients of the first five Neumann preconditioners are shown in Table I. These Neumann preconditioners will be used later to evaluate the relative performance of the PBNO preconditioner and the Chebyshev preconditioners (described in the next paragraph) when running the RTB problem using a Schur-complement form CG model.

A well-known class of polynomial preconditioners is based on the requirement that the function $s(\ )$ satisfy the linear least-squares optimization problem

$$\boldsymbol{k} \Leftarrow \min \|1 - \lambda s_c(\lambda)\|_w \qquad \|p(\lambda)\|_w \equiv \int_E p^2(\lambda) w(\lambda) d\lambda, \tag{3.4}$$

where:

- vector $\boldsymbol{k} = [k_0 \dots k_m]^T$ contains the optimal coefficients of polynomial $s_c(\tilde{A})$ in Eq. (3.2),
- $w(\lambda)$ is a weighting function that is nonnegative over the convex region E that encloses the spectrum of $\tilde{A}$,
- and $w(\lambda)$ is chosen such that function $s_c(\ )$ can be assembled recursively using Chebyshev polynomials of the first kind (hence the c subscript).

In our previous paper [1] we showed how the method described by Saad [9, p. 384-386] could be adapted to construct these so-called Chebyshev preconditioners for application to a system matrix with a positive definite spectrum confined to the interval $[0, 2]$. The coefficients of the first five of these Chebyshev preconditioners are shown in Table II, and these preconditioners will provide an existing-skill reference point against which to evaluate the PBNO preconditioner when running the RTB problem using a Schur-complement form CG model.

In order to create a Chebyshev preconditioner for a system with a complex spectrum, region E in the complex plane must be some convex form (*e.g.*, an ellipse or polygon) that approximately encloses the spectrum [8, 6]. This requirement introduces an element of arbitrariness into the selection of E, particularly when dealing with a spectrum like that in Fig. 2.1(b), which has a hull that cannot be well-represented by a convex form. As we show in the next section, the process by which the PBNO preconditioner is constructed is completely objective and eliminates the need to select the type and shape of the convex hull that is to enclose region E.

4. PBNO Preconditioner Development.

4.1. Optimization Problem Formulation. Since any polynomial in $\tilde{A}$ shares the same invariant subspaces that are common to $\tilde{A}$ and $\tilde{A}^{-1}$, the requirement that K approximate $\tilde{A}^{-1}$ is effectively the requirement that

$$\sigma(K) \approx \sigma(\tilde{A}^{-1}), \tag{4.1}$$

or alternatively, the requirement that

$$\sigma(K\tilde{A}) \approx \sigma(I). \tag{4.2}$$

In an effort to adequately satisfy criterion (4.2), we will require the optimal K to be the output of the optimization problem

$$K_{opt} \Leftarrow \min ||\boldsymbol{\sigma}(I - K\tilde{A})||_p, \tag{4.3}$$

where the bold font $\boldsymbol{\sigma}$ is a vector-valued operator that extracts the eigenvalues of its argument, and the subscript p is the index of a standard Euclidean p-norm.

Now if matrix $\tilde{A}$ has dimensions M-by-M, and we choose a power basis form for K as shown in Eq. (3.2), then the optimization problem (4.3) effectively becomes

$$\boldsymbol{k}_{opt} \Leftarrow \min \left[\sum_{i=1}^{M} |1 - s(\lambda_i)\lambda_i|^p \right]^{1/p}, \tag{4.4}$$

where vector $\boldsymbol{k}_{opt} = [k_0 \dots k_m]^T$ contains the optimal coefficients of the polynomial in Eq. (3.2). However, because of the poor behavior of power bases when doing numerical computations, we will instead represent coefficients of preconditioner K using the form

$$s_L(\lambda) = \sum_{i=1}^{m+1} c_i \mathcal{L}_j(\lambda), \tag{4.5}$$

where

$$\mathcal{L}_j(\lambda) = \prod_{\substack{i=1 \\ i \neq j}}^{m+1} \frac{(\lambda - n_i)}{(n_j - n_i)}, \tag{4.6}$$

are m^{th}-order Lagrange polynomials with nodal values n_i located at the Chebyshev points appropriately translated and scaled so as to be centered in the interval $[|\lambda_{\min}|, |\lambda_{\max}|]$. By virtue of representing K via Eq. (4.5), optimization problem (4.4) is replaced by

$$\boldsymbol{c}_{opt} \Leftarrow \min \left[\sum_{i=1}^{M} |1 - s_L(\lambda_i)\lambda_i|^p \right]^{1/p}, \tag{4.7}$$

where vector $\boldsymbol{c}_{opt} = [c_1 \dots c_{m+1}]^T$ contains the optimal coefficients of the polynomial in Eq. (4.5).

Now in principle, optimization problem (4.7) provides us with a method to construct matrix K by finding the coefficients in Eq. (4.5). However, in practice the large size of system matrix $\tilde{A}$ makes the optimization problem numerically intractable. Therefore, we will replace problem (4.7) with a tractable proxy via the procedure that follows.

First we recall that the Arnoldi algorithm applied to an arbitrarily large M-by-M matrix $\tilde{A}$ generates the factorization

$$\tilde{A}Q = Q\tilde{H}, \tag{4.8}$$

where $\tilde{H}$ is an upper Hessenberg matrix of size N-by-N and N is relatively small (< 150 in this paper). This upper Hessenberg matrix has a spectrum that we will denote by

$$\boldsymbol{\sigma}(\tilde{H}) = [\mu_1 \dots \mu_N]^T, \tag{4.9}$$

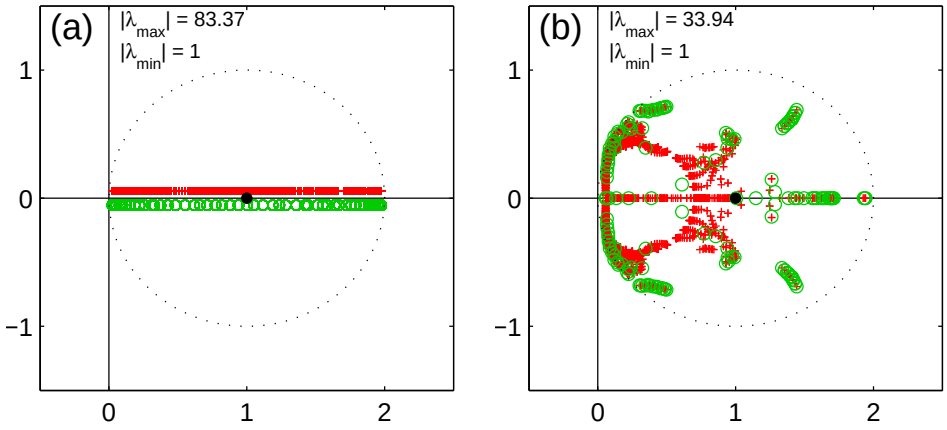

FIGURE 4.1. *(a) As in Fig. 2.1(a), except the spectrum of Hessenberg matrix $\tilde{H}$ (green circles) has been included along with the spectrum of system matrix $\tilde{A}$ (red crosses) The dimensions $\tilde{A}$ and $\tilde{H}$ respectively are $M = 676$ and $N = 100$. Both spectra are real, but have been displaced slightly upward (red) and downward (green) to avoid overlap. (b) As in Fig. 2.1(b), except the spectrum of $\tilde{H}$ (green circles) has been included along with the spectrum of $\tilde{A}$ (red crosses). The dimensions $\tilde{A}$ and $\tilde{H}$ respectively are $M = 3600$ and $N = 150$.*

and which can be quickly computed in its entirety using the QR algorithm [11, p. 211-224]. A key feature of $\boldsymbol{\sigma}(\tilde{H})$ is that it provides a good discrete approximation to the hull of the spectrum of matrix $\tilde{A}$ as long as N is adequately large. This property of $\boldsymbol{\sigma}(\tilde{H})$ is illustrated in Fig. 4.1(a) for the system matrix $\tilde{A}$ associated with a CG Schur-complement form model of the RTB problem, and in Fig. 4.1(b) for the scaled system matrix $\tilde{A}$ associated with a DG model of the same RTB problem.

The combination of the fact that $\boldsymbol{\sigma}(\tilde{H})$ contains the eigenvalues of $\tilde{H}$ farthest from $(1, 0)$ on the complex plane, and the fact that the residual polynomial

$$r(\lambda) = 1 - s_L(\lambda)\lambda, \tag{4.10}$$

contained inside optimization problem (4.7) tends to become large for those eigenvalues farthest from $(1, 0)$, suggests that we can replace (4.7) with the very tractable proxy problem

$$\boldsymbol{c}_{opt} \Leftarrow \min \left[\sum_{i=1}^{N} |1 - s_L(\mu_i)\mu_i|^p \right]^{1/p}. \tag{4.11}$$

In concluding this subsection we note that if $\boldsymbol{\sigma}(\tilde{H})$ is complex, then optimization problem (4.11) must be solved using a nonlinear least squares (NLLS) approach for all values of norm index $p \geq 2$. If $\boldsymbol{\sigma}(\tilde{H})$ is real, then a NLLS approach must be used for all values of $p > 2$.

4.2. Optimization Problem Solution. We can iteratively solve optimization problem (4.11) for any *even* value of norm index $p \geq 2$ using the Gauss-Newton (GN) algorithm as follows:

1. Create the vector-valued cost function

$$\boldsymbol{h}(\boldsymbol{c}) = \left[|1 - s_L(\mu_1)\mu_1|^{p/2}, \ldots, |1 - s_L(\mu_N)\mu_N|^{p/2} \right]^T, \tag{4.12}$$

   where vector $\boldsymbol{c} = [c_1 \ldots c_{m+1}]^T$ contains the coefficients c_i in Eq. (4.5) and the exponent form $p/2$ allows for the fact that the GN algorithm minimizes the *square* of the norm of the residual vector.
2. Approximate $\boldsymbol{h}(\boldsymbol{c})$ as a 1^{st}-order Taylor series

$$\boldsymbol{h}(\boldsymbol{c}) = \boldsymbol{h}(\boldsymbol{c_0}) + J\Delta\boldsymbol{c}, \tag{4.13}$$

   where $\boldsymbol{c_0}$ is an appropriate 1^{st}-guess[1] for $\boldsymbol{c}$, and J is a finite-difference approximated Jaco-

[1]For all cases in this paper we use $\boldsymbol{c}_0 = [1 \ldots 1]^T$, which, by virtue of the form of Eq. (4.5), makes $s_L(\lambda) = 1$.

bian matrix given by

$$J = \left[\frac{\partial \boldsymbol{h}}{\partial c_1}, \frac{\partial \boldsymbol{h}}{\partial c_2}, \ldots, \frac{\partial \boldsymbol{h}}{\partial c_{m+1}}\right]_{\boldsymbol{c}=\boldsymbol{c}_0}. \tag{4.14}$$

3. Obtain the QR factorization of the Jacobian $J = QR$ and apply the minimum residual criterion $Q^T\boldsymbol{h}(\boldsymbol{c}) = \boldsymbol{0}$ to Eq. (4.13) to obtain the normal system

$$R\Delta\boldsymbol{c} = -Q^T\boldsymbol{h}(\boldsymbol{c_0}), \tag{4.15}$$

which is then solved for $\Delta\boldsymbol{c}$ by back substitution.

4. If necessary, reduce $\Delta\boldsymbol{c}$ by repeated factors of 1/2 until the descent criterion

$$\|\boldsymbol{h}(\boldsymbol{c_0} + \Delta\boldsymbol{c})\|_2 < \|\boldsymbol{h}(\boldsymbol{c_0})\|_2 \tag{4.16}$$

is satisfied.

5. Replace $\boldsymbol{c_0}$ by $\boldsymbol{c_0} + \Delta\boldsymbol{c}$ and repeat steps 2-4 until the relative error criterion

$$\frac{\|\Delta\boldsymbol{h}\|}{\|\boldsymbol{h}\|} < \epsilon \tag{4.17}$$

is satisfied.

6. Convert the $\boldsymbol{c}_{opt}$ obtained from Steps 1-5 into the equivalent power basis coefficient vector $\boldsymbol{k}_{opt}$ so that Eq. (3.2) can be used when actually applying the PBNO preconditioner (via Horner's Method).

In all cases we found that letting $\epsilon = 10^{-5}$ in GN convergence criterion (4.17) was sufficient to ensure the GN algorithm produced a solution $\boldsymbol{c}_{opt}$ in optimization problem (4.11) that was of adequate precision for use in Eq. (4.5). The number of GN iterations required to satisfy criterion (4.17) depends on the order m of the PBNO preconditioner and the norm index p, varying from fewer than 10 for $m = 1$ and $p = 2$, to on the order of 50 for $m = 9$ and $p = 20$.[2]

Fig. 4.2(a)-(d) shows the effect of 5^{th}-order PBNO preconditioners with $p = 2$ and $p = 20$ on the spectrum of the preconditioned system matrices $K\tilde{A}$ and $K\tilde{H}$ for both CG Schur-complement form and DG form cases seen earlier in Fig. 4.1(a)-(d). The coefficients of these four PBNO preconditioners appear in Table I of this section to permit comparison with Row 5 of Tables I and II of Section 3 which display the coefficients of 5^{th}-order Neumann and Chebyshev preconditioners, respectively. In each panel of Fig. 4.2 we can see that the effect of the PBNO preconditioner is to drive all the eigenvalues of $K\tilde{A}$ and $K\tilde{H}$ closer to (1,0) on the complex plane in an effort to satisfy criterion (4.2). Most importantly, notice in all four panels of Fig. 4.2 that the spectral hull of $K\tilde{H}$ (green circles) lies virtually on top of the spectral hull of $K\tilde{A}$ (red crosses). This observation supports the assumption that tractable optimization problem (4.11) is a suitable proxy for the intractable problem (4.4).

By comparing Fig. 4.2(a) with 4.2(c) and Fig. 4.2(b) with 4.2(d), we can readily see that increasing the norm index from $p = 2$ and $p = 20$ in optimization problem (4.11) results in a more symmetric distribution of eigenvalues around (1,0). This is to be expected since increasing the p-norm index effectively results in a heavier weighting of the least squares residual vector components associated with the eigenvalues farthest from (1,0). We will see in Section 5 that the ability to change norm index p will allow us to create the optimal PBNO preconditioner for each of the GMRES, BICGS, and RICH iterative solvers that we will employ.

4.3. PBNO Preconditioner Construction and Application. For preconditioned system matrix spectra (red crosses) shown in Fig. 4.2a-d, the resolution of the RTB problem was sufficiently coarse so that we could actually construct matrix $K\tilde{A}$ and extract its *entire* spectrum via the QR method for comparison with the spectrum of the preconditioned Hessenberg matrix $K\tilde{H}$. For higher

[2]The range of iteration values just cited excludes the case when $p = 2$ and the spectrum of $\tilde{A}$ is positive definite, in which case optimization problem (4.11) is linear and GN converges in a single step.

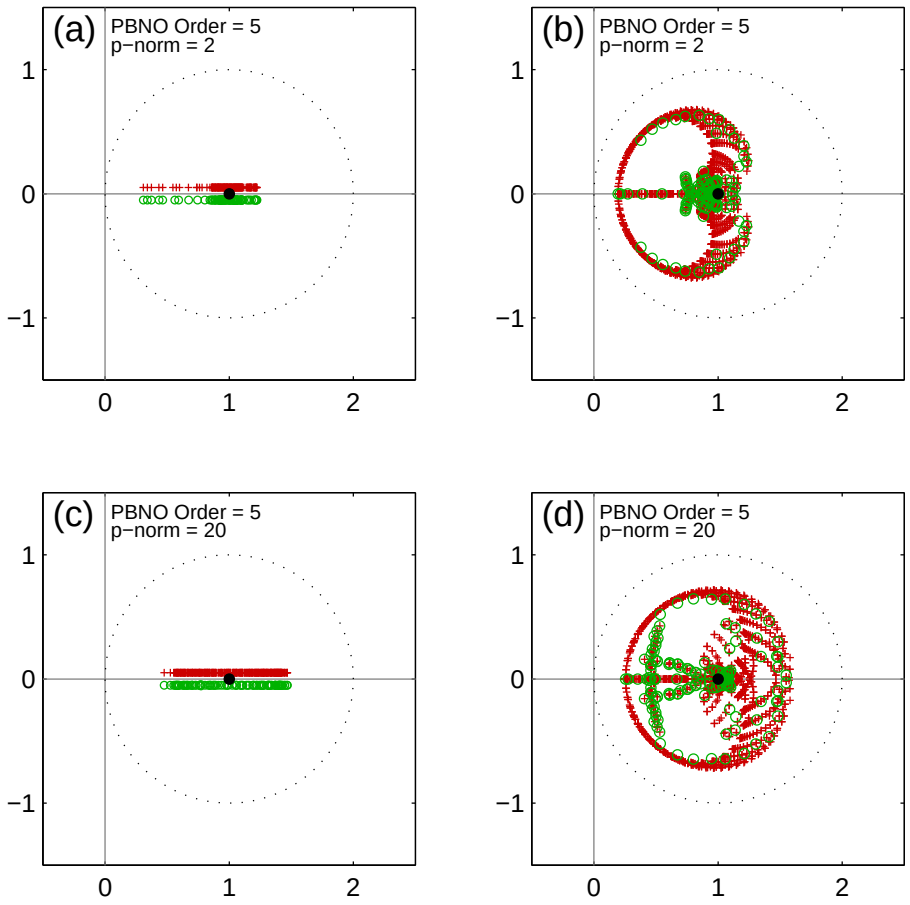

FIGURE 4.2. *(a) Spectra for the preconditioned system matrices $K\tilde{A}$ (red crosses) and KH (green circles) where $\tilde{A}$ is the same CG Schur-complement form system matrix as in Fig. (4.1a), and where K is a 5^{th}-order PBNO preconditioner with the norm index $p = 2$. (b) Spectra for the preconditioned system matrices $K\tilde{A}$ (red crosses) and KH (green circles) where $\tilde{A}$ is the same DG form system matrix as in Fig. (4.1b), and where K is a 5^{th}-order PBNO preconditioner with the norm index $p = 2$. (c)-(d) As in panels (a) and (b) of this figure, respectively, except for a PBNO preconditioner using a norm index $p = 20$.*

Model (p Norm)	k_0	k_1	k_2	k_3	k_4	k_5
CG ($p = 2$)	14.361	-58.760	102.84	-87.417	35.657	-5.5940
DG ($p = 2$)	3.8319	-8.0515	10.161	-7.4160	2.8788	-0.45999
CG ($p = 20$)	22.790	-117.92	238.63	-223.94	98.105	-16.235
DG ($p = 20$)	5.1638	-13.367	19.799	-16.502	7.1857	-1.2669

TABLE I

Coefficient values in Eq. (3.2) for the 5^{th}-order PBNO preconditioners that generated the spectra shown in Fig (4.2a-d).

resolution test case problems, and for any operational atmospheric prediction problem, we must begin with Eq. (2.2) in the more numerically relevant form

$$L_A(\boldsymbol{x}) = \boldsymbol{b}, \tag{4.18}$$

where $L_A(\ )$ is the linear operator that accomplishes the transformational action of the unscaled system matrix A. In this case the method for constructing and applying the PBNO preconditioner is as follows:

1. Prior to the time integration phase of the numerical model, construct an *unscaled* Hessenberg matrix H via an operator-based implementation of the Arnoldi algorithm, namely:
$$L_A(Q) = QH. \tag{4.19}$$
2. Apply the QR method to H to obtain λ_{max} and λ_{max} in order to construct λ_{mid} in Eq. (2.3), which in turn enables us to create a *scaled* system matrix operator
$$L_{\tilde{A}}(\ ) = \frac{1}{\lambda_{mid}} L_A(\ ), \tag{4.20}$$
for use during preconditioner construction and at each step of the time integration phase of the CG and DG model.
3. Employ the method outlined in Sections 4.1 and 4.2 to determine the coefficients of the PBNO preconditioner for the values of polynomial order m and norm index p specified by the user. We note here that Eq. (4.8) is replaced by the operator-based form
$$L_{\tilde{A}}(Q) = Q\tilde{H}. \tag{4.21}$$
4. After the preconditioner is constructed via Steps 1-3 above, it is then employed at each step in the time-integration phase to solve the operator-based form of Eq. (3.1)
$$L_K \circ L_{\tilde{A}}(\boldsymbol{x}) = L_K(\tilde{\boldsymbol{b}}), \tag{4.22}$$
via the particular iterative solver specified by the user.

We emphasize that for Steps 1-3 listed above, the greatest computational cost by far is incurred in Step 1 when the matrix H is constructed via the Arnoldi algorithm, particularly if system matrix A is large. This cost is also applicable when constructing the Chebyshev preconditioner, as is the cost to compute the spectrum of H in Step 2. The cost to complete Step 3, which is the only step unique to the PBNO preconditioner, is negligible compared to the cost of Step 1.

Fig. 4.3 shows the impact on the spectrum of $K\tilde{H}$ due to a set of PBNO preconditioners of increasing order constructed for a DG model RTB problem with five times the spatial resolution of the analogous coarse-resolution model having the spectra shown in Fig. 4.2. Notice the similarity of the spectra of the unpreconditioned scaled Hessenberg matrix $\tilde{H}$ in Figs. 4.1(b) and Figs. 4.3(a). Likewise, note the similarity of the spectra of the matrices $K\tilde{H}$ (green circles) in Figs. 4.2(d) and Figs. 4.3(d) where in each case a 5^{th}-order PBNO preconditioner with $p = 20$ was employed. Finally, notice in Fig. 4.3(f) the high degree of axisymmetry about (1,0) exhibited by the spectrum of $\tilde{H}$ when a 9^{th}-order the PBNO preconditioner is employed.

5. Results in a Serial Computing Environment. In order to adequately map out the large parameter space associated with:

- two variants of NUMA2d (CG-Schur and DG),[3]
- a range of possible time-steps with Courant No. as high as 32,
- three iterative solvers (GMRES, BICGS, RICH),
- three types of polynomial preconditioners (NEUM, CHEB, PBNO),
- preconditioner order as high as 9^{th}-order,
- and the adjustable p-norm index in PBNO optimization problem (4.11),

we initially use a coarse spatial resolution, a low order time integration scheme, and a relatively short simulation time of 100s. The model domain is spatially discretized using a 25-by-25 element grid and 5^{th}-order LGL points within each element, and a 2^{nd}-order accurate Additive Runge-Kutta (ARK2) time integration scheme [4]. A set of six different time-steps is employed and corresponds to a Courant No. as small as 2 and as large as 32.[4] As the analysis proceeds we will include selected results that employ 9^{th}-order spatial resolution, higher order ARK methods, and a model simulation time of 700s to show that conclusions based on the coarse simulation runs are justified.

[3] NUMA2d refers to the two-dimensional version of the NUMA model.

[4] A Courant No. of 32 is the largest for which the RTB problem will run to completion (*i.e.,* bubble at top of domain at 700s) without exceeding the explicit CFL limit.

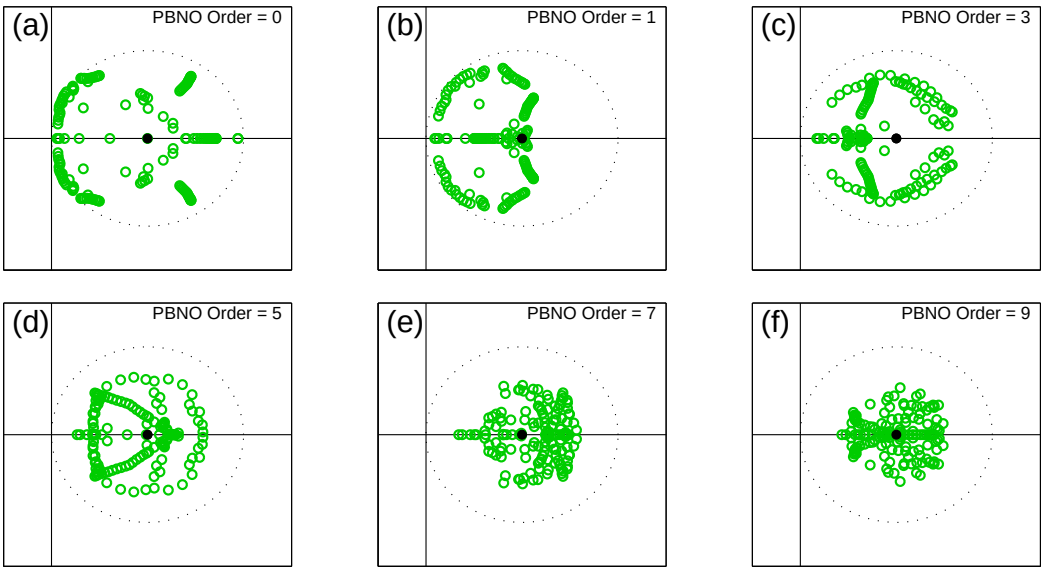

FIGURE 4.3. *(a) Spectrum of the scaled Hessenberg matrix $\tilde{H}$ derived from a DG model of the RTB problem using 25×25 elements, 5^{th}-order LGL polynomials, two-stage ARK time differencing, and a Courant No. of 16. The corresponding scaled system matrix have dimensions of $22,550 \times 22,500$. (b)-(f) As in panel (a), except for a scaled preconditioned Hessenberg matrix $K\tilde{H}$ where the PBNO preconditioners K have the order shown in the upper right of the panel, and where constructed using a norm index of $p = 20$.*

5.1. Solver Performance Dependence on PBNO p-Norm Index. Recall that in optimization problem (4.11) the index of the p-norm employed can be varied. The effect of changing index p on the performance of the GMRES, BICGS, and RICH iterative solvers using PBNO preconditioners of up to 9^{th}-order is illustrated in Figs. 5.1 and Figures 5.2 for CG-NUMA2d and DG-NUMA2d, respectively. For the CG version of NUMA2d (CG-NUMA2d), increasing the value of index p from 2 to 10 to 20 slightly degrades the performance of GMRES (Fig. 5.1(a)-(b)), has basically negligible effect of the performance of BICGS (Fig. 5.1(c)-(d)), but significantly improves the performance of RICH (Fig. 5.1(e)-(f)). Based on the results shown, when running CG-NUMA2d a setting of $p = 2$ would be appropriate when using either the GMRES or BICGS solver, and a setting of $p = 10$ would be an appropriate choice when using the RICH solver. The improved performance of the RICH solver as norm index p increases is not surprising given that:

- the spectrum of $K\tilde{H}$ becomes more axisymmetrically distributed around (1,0) as index p increases (compare Figs. 4.2(a) and (d)),
- and the convergence rate of the RICH solver is determined by the eigenvalue of $K\tilde{H}$ farthest from (1,0) as shown in Appendix A.

A comparison of Figs. 5.1(b), 5.1(d), and 5.1(f), reveals several important facts:

- Although the iterations/time-step for BICGS are roughly half those of GMRES, the wall-clock of BICGS is still greater than that of GMRES.
- Although the RICH solver is much slower than either GMRES or BICGS when PBNO preconditioner order m is low, the RICH solver does become increasingly competitive with the other two solvers as index m increases.

The first item above is a consequence of the fact that BICGS applies the system matrix twice during each iteration, and the fact that GMRES's use of an expanding Krylov-vector search space is not a significant detriment when the number of iterations/time-step are relatively low. The second

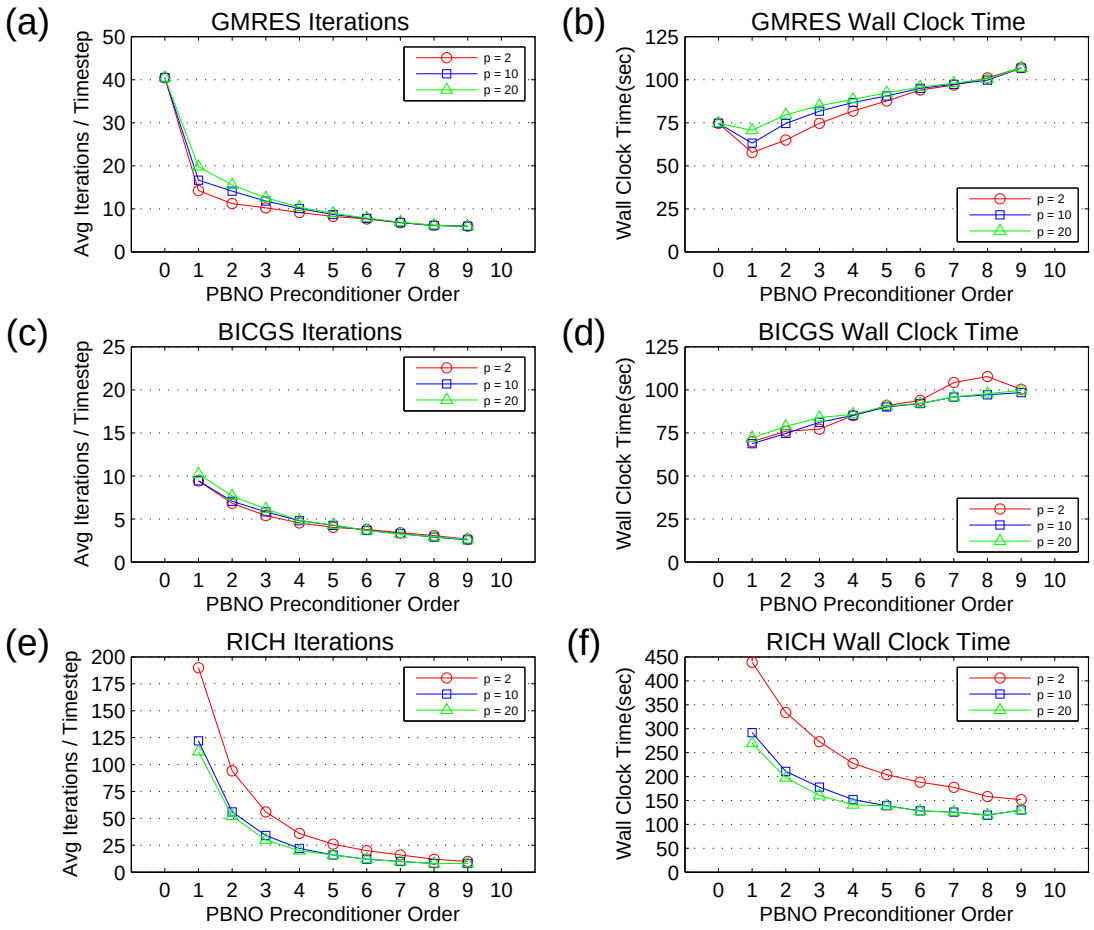

FIGURE 5.1. *(a)-(b) Effect of PBNO preconditioner p-norm index on GMRES Iterations/Time-step and Wall Clock Time, respectively, when running the CG Schur Complement form version of NUMA2d with a time-step corresponding to a Courant No. of 16. (c)-(d) As in (a)-(b), except for the BICGS iterative solver. The missing data points for PBNO order $m = 0$ indicates that unpreconditioned BICGS solver would not reliably converge at every time-step. (e)-(f) As in (a)-(b), except for the RICH iterative solver. The missing data points for PBNO order $m = 0$ indicates that the convergence rate of unpreconditioned RICH at each time-step was prohibitively slow.*

item above is significant given that the RICH iterative solver is dot-product-free, which should increase its competitiveness with the other solvers when we consider model performance in a parallel computing environment in future work.

For DG-NUMA2d, increasing the value of index p from 2 to 10 to 20 moderately improves the performance of GMRES (Fig. 5.2(a)-(b)) and BICGS (Fig. 5.2(c)-(d)), and significantly improves the performance of RICH (Fig. 5.2(e)-(f)). Based on the results shown, when running CG-NUMA2d a setting of $p = 10$ would be appropriate when using either the GMRES or BICGS solver, and a setting of $p = 20$ would be an appropriate choice when using the RICH solver. In comparing Figs. 5.2(b), 5.2(d), and 5.1(f), it is important to note that the wall-clock time of the RICH solver (with $p = 20$) is of the same order as both GMRES and BICGS regardless of PBNO preconditioner order m. A summary of the p-norm index values used in all subsequent runs of NUMA2d are provided in Table I.

5.2. PBNO Performance Relative to Existing Preconditioners. A comparison of the performance of PBNO preconditioners up to 9^{th}-order relative to analogous Neumann (NEUM) and Chebyshev (CHEB) preconditioners is shown in Fig. 5.3(a) and (b) for low resolution 100s runs of the Schur-complement form of CG-NUMA2d using an ARK2 time integrator. Both the CHEB and PBNO preconditioners significantly outperform the no-skill threshold set by the corresponding

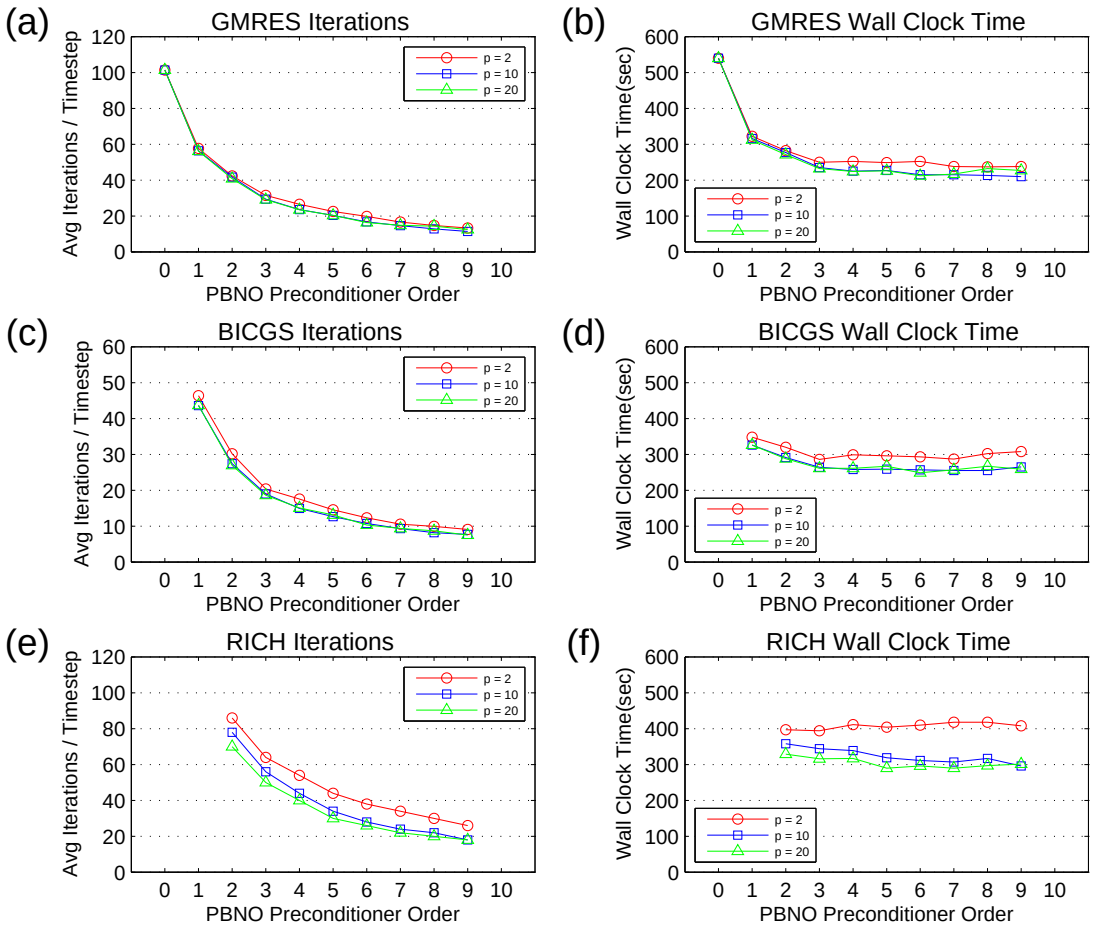

FIGURE 5.2. *(a)-(f) As in Fig. 5.1a-f, except for the DG form version of NUMA2d.*

NUMA2d Variant	GMRES	BIGCS	RICH
CG (Schur)	$p = 2$	$p = 2$	$p = 20$
DG	$p = 10$	$p = 10$	$p = 20$

TABLE I
Norm index values used in optimization problem (4.11) when constructing PBNO preconditioners for NUMA2d.

NEUM preconditioners for all polynomial orders. For all polynomial orders > 1, the PBNO preconditioners achieve a greater reduction in iterations/time-step and wall clock time than do the CHEB preconditioners. With regard to wall clock time, CG NUMA2d runs approximately 12 percent faster when preconditioned with PBNO versus CHEB for polynomial orders ≥ 2. In Fig. 5.3(c)-(d), in which a 3^{rd}-order accurate (ARK3) time integrator is employed, we see qualitatively similar results, except that in terms of wall clock time NUMA2d runs as much as 30 percent faster when a PBNO preconditioner is used compared to a CHEB preconditioner. We also made the same comparison using a 4^{th}-order accurate (ARK4) time integrator (not shown) and obtained results that were very similar to those shown for ARK3 in Fig. 5.3(c)-(d). Similar results were obtained using various backward differencing time integrators as well, with PBNO outperforming CHEB in all cases. Fig. 5.4 provides results analogous to Fig. 5.3, except that model resolution has been increased by using 9^{th}-order LGL nodal points. The comparison clearly shows that the improvement in GMRES performance achieved by PBNO preconditioners compared to CHEB and NEUM preconditioners is not dependent on model resolution.

In comparing the wall clock time results in Figs. 5.3(b)-(d) and 5.4(b)-(d), it is evident that the

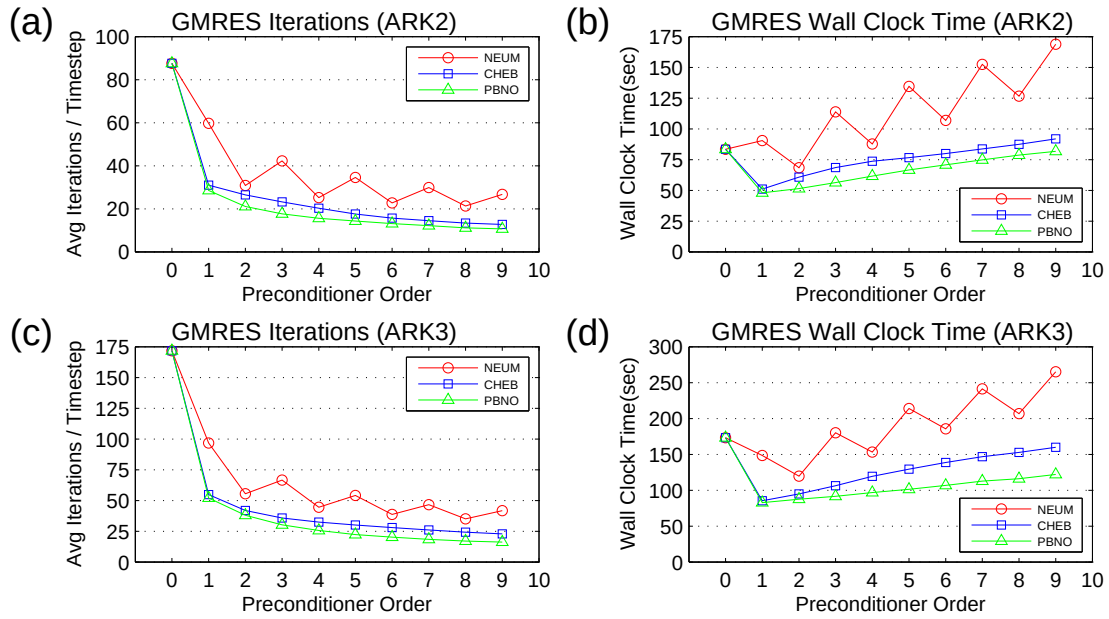

FIGURE 5.3. *(a)-(b) Iterations/time-step (a) and wall clock time (b) achieved by various orders of the NEUM, CHEB and PBNO preconditioners when employed in the lower spatial resolution Schur-complement form of CG-NUMA2d running at a Courant No. of 32, employing an ARK2 time integrator, and running for 100s of model simulation time. (c)-(d) As in (a) and (b), except for using a* 3^{rd}*-order accurate (ARK3) time integrator.*

CG-NUMA2d runs the fastest in a *serial* computing environment when 1^{st}-order preconditioning is used, and that there is little difference between the performance of the 1^{st}-order CHEB and PBNO preconditioners. However, it is important to notice, for example, in Fig. 5.3(d) that 9th-order PBNO-preconditioned GMRES still runs much faster than unpreconditioned GMRES, and runs significantly faster than CHEB-preconditioned GMRES. In addition, as preconditioner order is increased from 1 to 9, PBNO-preconditioned GMRES wall clock time increases by only a factor of $122/83 = 1.45$, whereas PBNO-preconditioned GMRES iterations decrease by a significantly larger factor of $52/16 = 3.25$. This observation combined with the fact that the number of dot products computed in GMRES is proportional to the square of the number of iterations, means that there is the potential for a reduction in wall clock time as PBNO order is increased when employing GMRES in a parallel computing environment.

Runs for CG-NUMA2d were also performed using BICGS with the NEUM, CHEB, and PBNO preconditioners (not shown). The results were roughly analogous to the GMRES results in Figs. 5.4(c), but have the following differences:

- The RTB problem would not run to completion (i.e., to 100s) when using the BICGS solver and an *odd* order NEUM preconditioner. By contrast, we can see in Fig. 5.4(b) and (d) that the RTB problem ran more slowly when using the GMRES solver and an *odd* order NEUM preconditioner compared to the runs using NEUM preconditioners of the next lower or higher *even* order.
- The performance of BICGS preconditioned with PBNO and CHEB is virtually the same. However, the wall clock times are about 10 percent slower than for PBNO-preconditioned GMRES for all preconditioner orders greater than 2.

In concluding this subsection, we note that preconditioner performance comparisons such as those shown in Figs. 5.3 and 5.4 for CG-NUMA2d cannot be made for DG-NUMA2d for several reasons. Firstly, DG-NUMA2d would not reliably run to completion when preconditioned with the NEUM preconditioner presumably due to fact that some of the eigenvalues of $\tilde{A}$ are located very nearly on the edge of the unit disk centered at (1,0) as seen in Fig. 2.1b. Secondly, an objective head-to-head comparison of a CHEB and PBNO preconditioner for a system matrix with

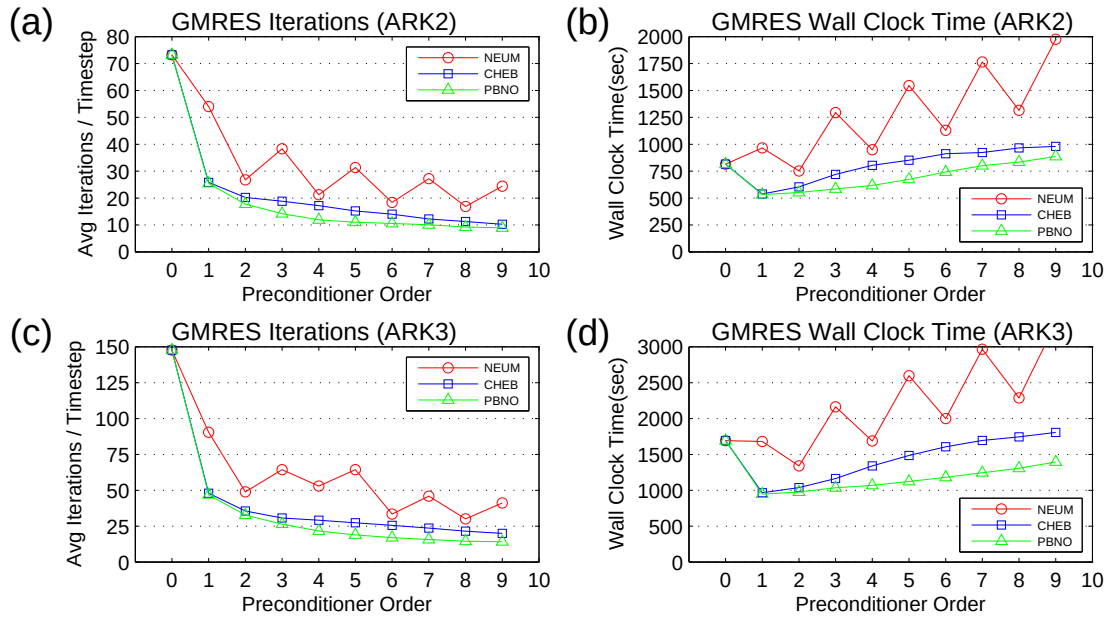

FIGURE 5.4. *(a)-(d) As in Fig. 5.3(a)-(d), expect for running NUMA2d at a higher spatial resolution employing* 9^{th}*-order LGL polynomials.*

a complex spectrum is problematic since the CHEB preconditioner must be constructed employing a subjectively chosen (*e.g.*, either elliptical or polygonal) *continuous* and *convex* hull that encloses the spectrum of the Hessenberg matrix $\tilde{H}$, whereas the PBNO preconditioner is based on an objective and effectively *discontinuous* and *non-convex* spectral hull represented by the actual spectrum of $\tilde{H}$. Since in the comparisons above the performance of the PBNO preconditioner either exceeds (for GMRES) or matches (for BICGS) the performance of the CHEB preconditioner when used in CG-NUMA2d, all subsequent results involving DG-NUMA2d will involve only the PBNO preconditioner.

5.3. Relative Performance of PBNO-preconditioned Solvers. The effect of PBNO preconditioners on the iterations/time-step and wall clock time of GMRES, BICGS, and RICH solvers as a function of RTB problem Courant No. are shown in Figs. 5.5 and 5.6 for CG-NUMA2d and DG-NUMA2d, respectively. For the CG case, comparing Figs. 5.5(b),(d) and (f) reveals that in terms of minimum wall clock time in a serial computing environment:

- For GMRES a 1^{st}-order preconditioner is optimal for Courant Nos. > 8.
- For BICGS using no preconditioner at all is optimal for all Courant Nos.
- For RICH higher order preconditioning becomes increasingly beneficial as the Courant No. increases, and results in the simple RICH solver becoming increasingly competitive with the more sophisticated GMRES and BICGS solvers.

By contrast, for the DG case, comparing Figs. 5.6(b),(d) and (f) reveals that in terms of minimum wall clock time all three solvers run more efficiently when higher order preconditioning is combined with large Courant No. In particular:

- when employing GMRES the model runs the fastest when a 9^{th}-order preconditioner is combined with the maximum Courant No. of 32.
- when employing BICGS the model runs the fastest when a 7^{th}-order preconditioner is combined with the maximum Courant No. of 32.
- when employing RICH the model runs the fastest when a 9^{th}-order preconditioner is combined with the maximum Courant No. of 32.

When looking over Figs. 5.5(b), (d) and (f) and 5.6(b), (d) and (f) it is clear that the lowest wall clock time tends to occur when using the largest Courant No. of 32 (and thus longest time-

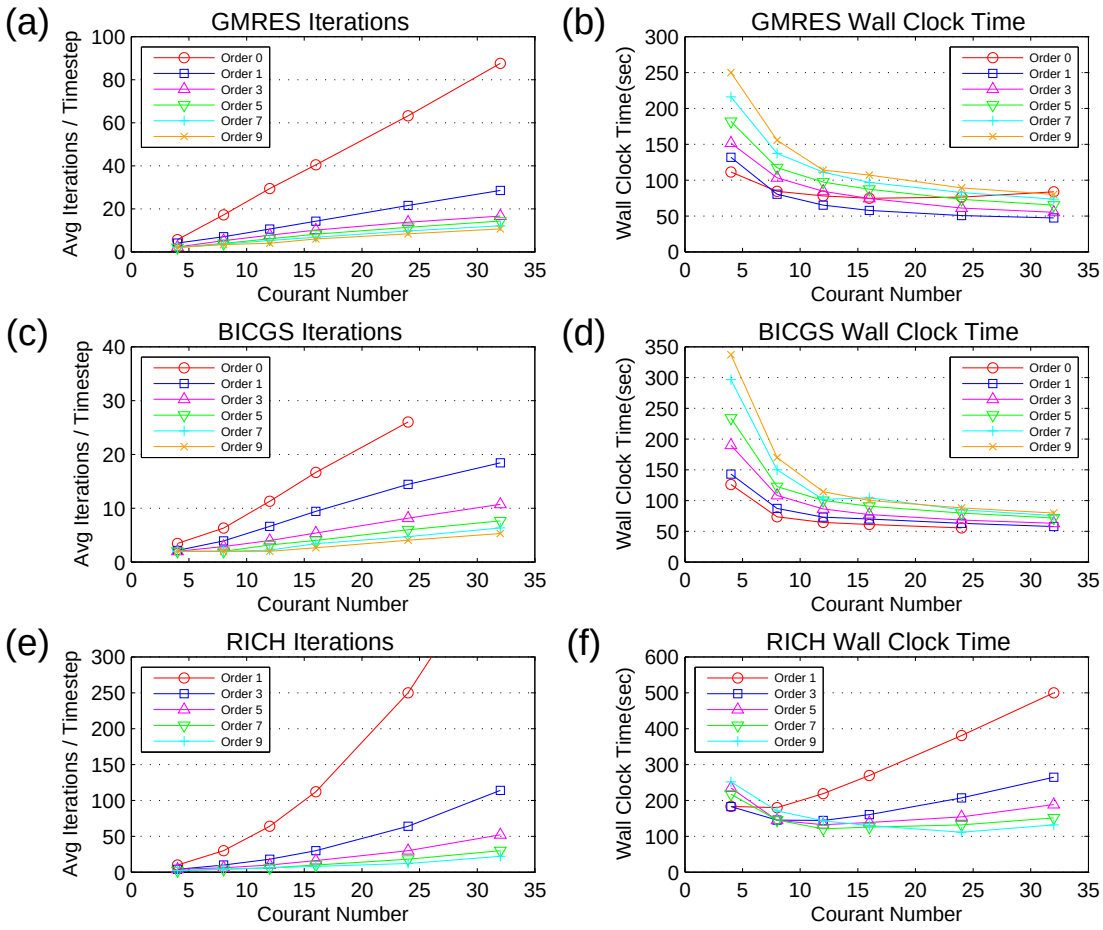

FIGURE 5.5. *(a)-(b) Iterations/time-step (a) and wall clock time (b) as a function of Courant No. exhibited by CG-NUMA2d preconditioned with the orders of the PBNO-preconditioner shown in the legend. For these results CG NUMA2d employed 5th-order LGL polynomials, an ARK2 time integrator, and RTB simulation time of 100s. (c)-(d) As in (a) and (b), except for using the BICGS solver. The missing data point for 0-order preconditioning* (i.e., *unpreconditioned) and a Courant No. of 32 indicates that CG NUMA2d would not run to completion when employing the BICGS solver. (e)-(f) As in (a) and (b), except for using the RICH solver.*

step). However, the preconditioner order needed to achieve the lowest wall clock time when running the model at Courant No. 32 varies with the iterative solver used and whether a CG or DG model is employed. Fig. 5.7 provides an example of the PBNO-preconditioned CG and DG model performance when running at the maximum Courant No. of 32 and employing an ARK4 time integrator. With regard to iterations/time-step (Fig. 5.7(a) and (c)), important points to note are that:

- for the CG model the performance of the RICH solver is much worse than the GMRES and BICGS solvers for low PBNO order, but becomes increasingly competitive as PBNO order increases,
- for the DG model the performance of the RICH solver is only slightly worse than the GMRES and BICGS solvers for low PBNO order, but nearly matches the performance of GMRES for PBNO order greater than 4,
- and, although BICGS has significantly fewer iterations/time-step than GMRES in both the CG and DG models, it must be remembered that this advantage tends to be offset to some degree by the fact BICGS applies the preconditioned system matrix twice at each time step.

With regard to wall clock time (Fig. 5.7(b) and (d)), important points to note are that:

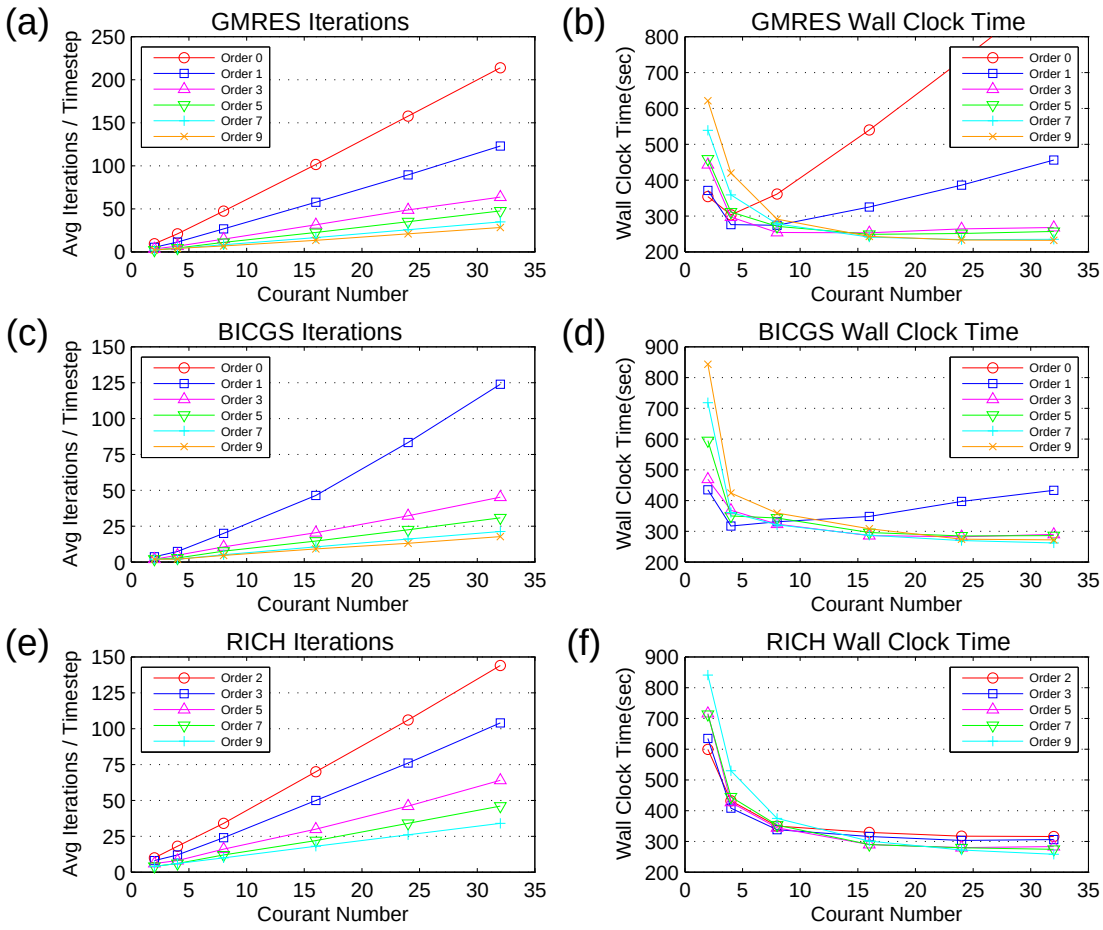

FIGURE 5.6. *(a)-(f) As in Figs. 5.5(a)-(f), except for using DG-NUMA2d.*

- CG model wall clock time tends to slow down with increasing PBNO order using either the GMRES or BICGS solver; however, the rate of slowdown is modest compared to the proportionally larger reduction in iterations/time-step.
- in the CG model the performance of the RICH solver is much worse than the GMRES and BICGS solvers for low PBNO order, but becomes increasingly competitive as PBNO order increases,
- in the DG model 1^{st}-order preconditioning of GMRES or BICGS results in a dramatic reduction in wall clock time compared to unpreconditioned GMRES (the only solver able to run unpreconditioned in the DG model),
- in the DG model the performance of all three solvers is very similar for PBNO order $m \geq 2$ and we can see that a modest reduction in wall clock time occurs as PBNO order increases,
- and, the competitiveness of the RICH algorithm for PBNO order $m \geq 2$ is particularly noteworthy given that the algorithm is run in a dot-product-free mode in NUMA2d.

5.4. Long-Duration, High-Resolution Simulation Comparisons. As mentioned earlier, to expeditiously map out the large parameter space associated with multiple versions of NUMA2d, preconditioners, solvers, etc, we initially presented results based on coarse spatial resolution, a low-order time integration scheme, and relatively short simulation time of 100s. To ensure that PBNO preconditioning up to order $m = 9$ produces stable and consistent results for longer RTB simulation times, we ran the problem out to 700s at which time the bubble impacts the top of the domain. For these runs in both the CG and DG models we employ a problem domain that is spatially discretized using a 25-by-25 element grid and 9^{th}-order LGL points within each element, and we use a 3^{nd}-order

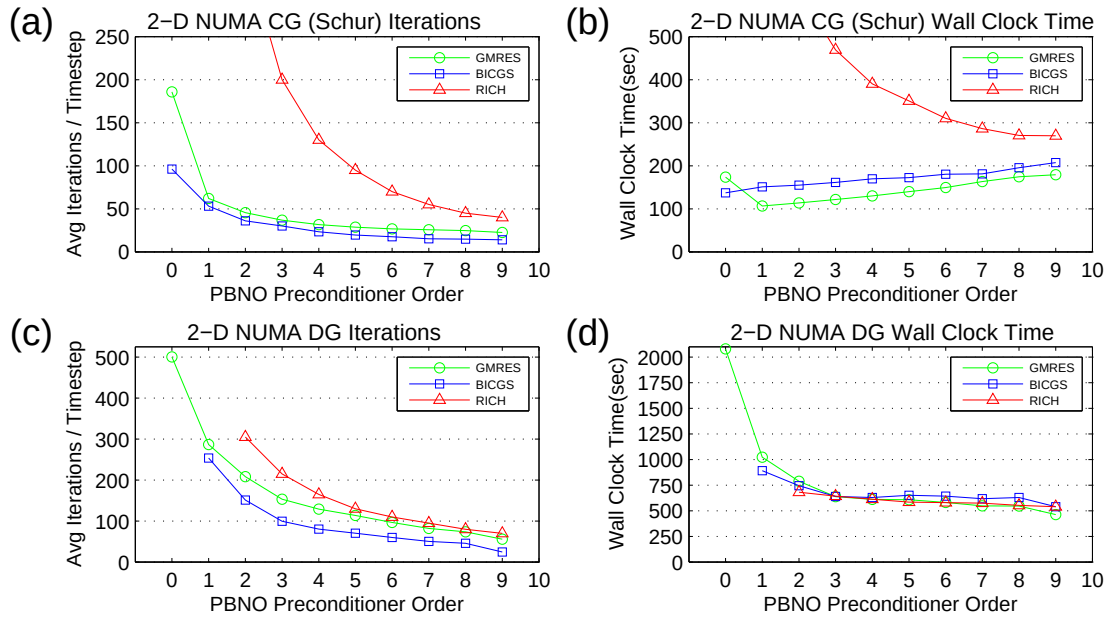

FIGURE 5.7. *(a)-(b) Comparison of iterations/time-step (a) and wall clock time (b) as a function of PBNO preconditioner order for the Schur-complement form of CG-NUMA2d running at the maximum Courant No. of 32 and using the GMRES (green circles), BICGS (blue squares), and RICH (red triangles) iterative solvers. (c)-(d) As in (a)-(b), except for DG-NUMA2d. Missing data points in DG results indicates that the model would not run to completion using that particular combination of iterative solver and PBNO order. For both the CG and DG results the model employed 5th-order LGL polynomials, an ARK4 time integrator, and RTB simulation time of 100s.*

accurate Additive Runge-Kutta (ARK3) time integration scheme [5].

Potential temperature fields that result from running the Schur-complement form of the CG model using the above specifications and employing unpreconditioned GMRES, 9th-order PBNO-preconditioned GMRES, BICGS, and RICH are depicted in Fig. 5.8(a)-(d), respectively. The fields shown are at the 600s point in the 700s simulation, which is approximately the time when the bubble attains its maximum upward speed. As a result, 600s is the time when the advective Courant No. reaches its maximum and comes very close to the advective CFL limit. It is visually evident that all four runs closely reproduce the fine structure of the rising bubble. The small values of $|\Delta T|_{max}$ (described in the caption) shown in Fig. 5.8(b)-(d) provide numerical confirmation that 9^{th}-order preconditioning of the GMRES, BICGS, and RICH solvers is not causing any significant dispersion or phase lag relative to the unpreconditioned GMRES field (Fig. 5.8(a)). A comparison of wall clock time values in the lower right of each panel shows that relative to unpreconditioned GMRES:

- preconditioned GMRES is running ≈ 1.32 times faster,
- preconditioned BICGS is running ≈ 1.21 times faster,
- and preconditioned dot-product-free RICH is running ≈ 2.20 times slower.

A comparison of average iterations/time-step values in the upper right of each panel of Fig. 5.8 shows that relative to unpreconditioned GMRES:

- preconditioned GMRES uses ≈ 12.0 fewer iterations,
- preconditioned BICGS uses ≈ 20.8 fewer iterations,
- and preconditioned RICH uses ≈ 3.05 fewer iterations.

Potential temperature fields obtained from the DG-NUMA2d that are analogous to those in

[5] The total number of grid points in this simulation is 50,625 with each grid point having 4 variables for a total of 202,500 degrees of freedom. Using a time-step of dt=0.148287 means that the linear system has to be solved $\frac{700 \cdot k}{dt} = 14,161$ times during the 700s simulation, where k=3 denotes the number of Runge-Kutta stages used in the ARK method.

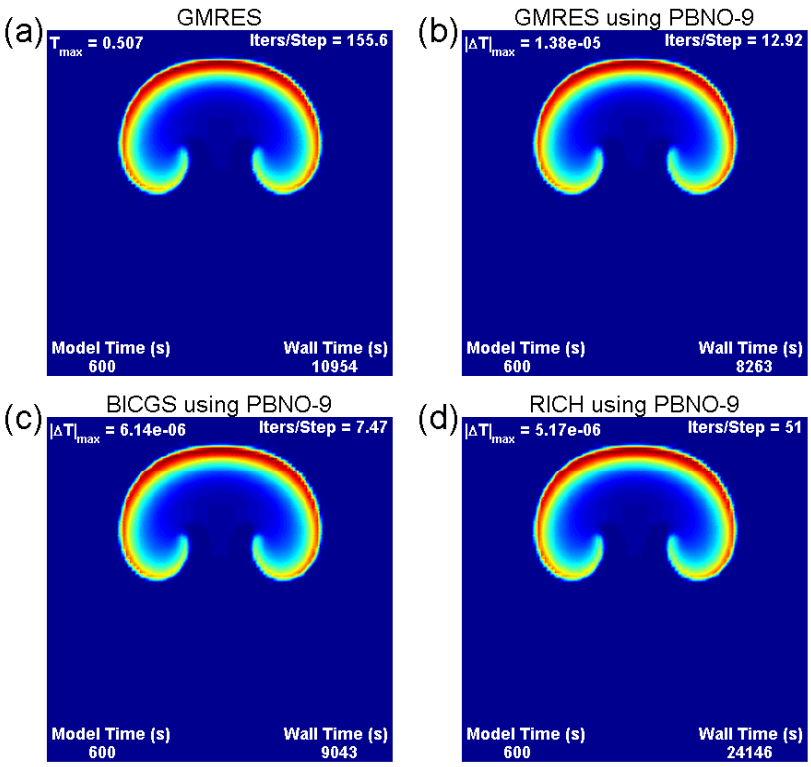

FIGURE 5.8. *(a)-(d) Potential temperature fields for the RTB test case at 600s in a 700s simulation time run of the Schur-complement CG-NUMA2d. Model setup specifications are as described in the text. The heading on each panel identifies the iterative solver and order of PBNO preconditioning employed. Iterations/time-step and wall clock time in seconds appears in the upper and lower right of each panel, respectively. The value in the upper left of panel (a) is the maximum potential temperature in the bubble. The value in the upper left of panels (b)-(c) is the infinity norm of the difference between the unpreconditioned GMRES potential temperature state vector in (a) and the respective potential temperature state vector in panels (b), (c), and (d).*

Fig. 5.8(a)-(d) are shown in Fig. 5.9(a)-(d). Again, it is visually evident that all four runs closely reproduce the fine structure of the rising bubble. The small values of $|\Delta T|_{max}$ shown in Fig. 5.9(b)-(d) provide numerical confirmation that 9^{th}-order preconditioning of the GMRES, BICGS, and RICH solvers is not causing any significant dispersion or phase lag relative to the unpreconditioned GMRES field (Fig. 5.9(a)). A comparison of wall clock time values in the lower right of each panel shows that relative to unpreconditioned GMRES (which is the only solver that runs without preconditioning in DG NUMA2d):

- preconditioned GMRES is running ≈ 3.27 times faster,
- preconditioned BICGS is running ≈ 2.761 times faster,
- and even preconditioned dot-product-free RICH is running ≈ 1.90 times faster.

A comparison of average iterations/time-step values in the upper right of each panel of Fig. 5.9 shows that relative to unpreconditioned GMRES:

- preconditioned GMRES uses ≈ 8.11 fewer iterations,
- preconditioned BICGS uses ≈ 13.1 fewer iterations,
- and preconditioned RICH uses ≈ 4.08 fewer iterations.

6. Conclusion. We have introduced a method for constructing a polynomial preconditioner using a nonlinear least squares (NLLS) algorithm and have shown that this polynomial-based NLLS-optimized (PBNO) preconditioner significantly improves the performance of 2-D continuous Galerkin

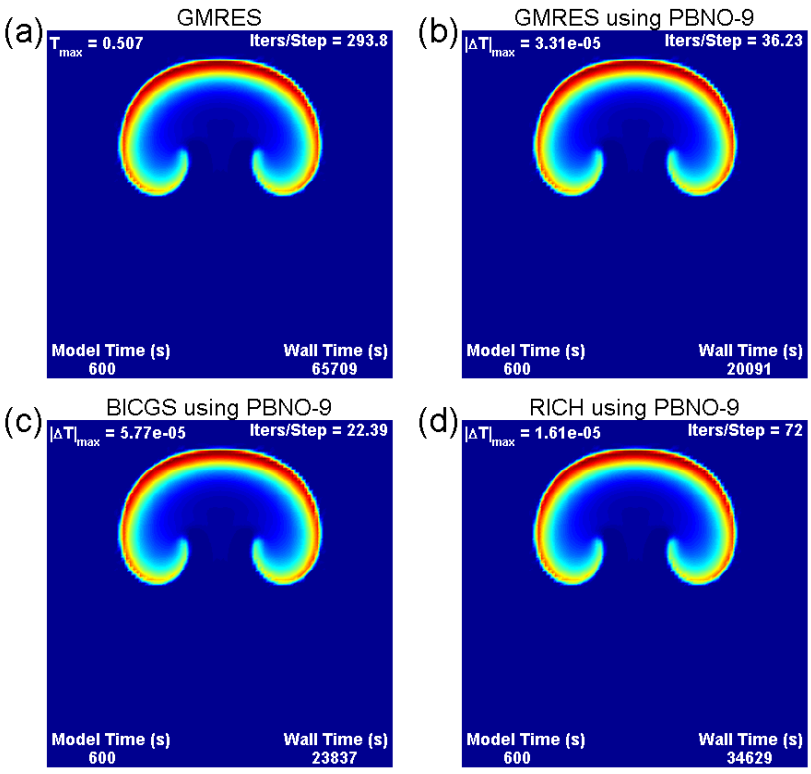

FIGURE 5.9. *(a)-(d) As in 5.8, except for DG-NUMA2d.*

(CG) and discontinuous Galerkin (DG) fluid dynamical research models when running the rising bubble (RTB) test case using implicit-explicit time integrators. When employed in a serially computed Schur-complement form of the 2-D CG model with positive definite spectrum, the PBNO preconditioner achieves greater reductions in GMRES iterations and model wall-clock time compared to the analogous linear least-squares-derived Chebyshev polynomial preconditioner. Whereas constructing a Chebyshev preconditioner to handle the complex spectrum of the DG model would introduce an element of arbitrariness in selecting the appropriate convex hull, construction of a PBNO preconditioner for the 2-D DG model utilizes precisely the same objective NLLS algorithm as for the CG model. As in the CG model, the DG model with PBNO preconditioner achieves significant reduction in GMRES iteration counts and model wall-clock time. Comparisons of the ability of the PBNO preconditioner to improve CG and DG model performance when employing BICGS and RICH have also been included. In particular, we have shown that higher order PBNO preconditioning of RICH (which is run in a dot product free mode) makes the algorithm competitive with GMRES and BICGS in a serial computing environment, especially when employed in a DG model. Because the NLLS-based algorithm used to construct the PBNO preconditioner can, without modification, handle both positive definite and complex spectra, we believe that the PBNO preconditioning approach is, for certain types of problems, an attractive alternative existing polynomial preconditioners based on linear least-squares methods.

In future work we plan to evaluate the performance of the PBNO preconditioned GMRES, BICGS, and dot-product-free RICH solver in implicit-explicit and fully implicit CG and DG versions of the three-dimensional NUMA model that are under development for use in massively parallel computing environments. The current study described in this work has employed the serial two-dimensional version of NUMA to allow us to focus our efforts on preconditioner formulation. The next study will take place on the full three-dimensional NUMA model on large core-count computing platforms; however, the emphasis of that future work will be entirely on the implementation and

scalability of the model since the method for constructing the PBNO preconditioner will remain unchanged.

Acknowledgments. The authors gratefully acknowledge the support of the Computational Mathematics program of the Air Force Office of Scientific Research. FXG also gratefully acknowledges the support of the Office of Naval Research through program element PE-0602435N, and the National Science Foundation (Division of Mathematical Sciences) through program element 121670. We also would like to thank Michal Kopera and several anonymous reviewers for their helpful suggestions for improving the manuscript.

Appendix A. Dot-Product-Free Richardson Iteration.

As described in [12, p. 22], the Richardson iteration for solving Eq. (2.5) is defined by the solution update equation

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{r}_n, \tag{A.1}$$

where residual vector $\boldsymbol{r}_n$ is given by

$$\boldsymbol{r}_n = \tilde{\boldsymbol{b}} - \tilde{A}\boldsymbol{x}_n. \tag{A.2}$$

Appropriately combining Eq. (A.1) and Eq. (A.2) results in the residual update equation

$$\boldsymbol{r}_n = (I - \tilde{A})\boldsymbol{r}_{n-1}, \tag{A.3}$$

or after $n-1$ recursive substitutions

$$\boldsymbol{r}_n = (I - \tilde{A})^n \boldsymbol{r}_0, \tag{A.4}$$

which we can write more concisely as

$$\boldsymbol{r}_n = B^n \boldsymbol{r}_0. \tag{A.5}$$

Now to determine both the sufficient condition for $\boldsymbol{r}_n \to \boldsymbol{0}$ as $n \to \infty$ as well as a lower limit on the rate of convergence we proceed as follows. First we expand the initial residual vector $\boldsymbol{r}_0$ in terms of the normalized eigenvectors $\boldsymbol{v}_i$ of matrix B

$$\boldsymbol{r}_0 = \sum_{i=0}^{M} c_i \boldsymbol{v}_i, \tag{A.6}$$

and substitute Eq. (A.6) into Eq. (A.5) giving

$$\boldsymbol{r}_n = B^n \sum_{i=0}^{M} c_i \boldsymbol{v}_i, \tag{A.7}$$

or equivalently,

$$\boldsymbol{r}_n = \sum_{i=0}^{M} c_i \lambda_i^n \boldsymbol{v}_i, \tag{A.8}$$

where here the λ_i's are the eigenvalues of B. From the form of Eq. (A.8) it is clear that if $|\lambda_i| < 1 \; \forall \; i$ (*e.g.*, as in Fig. 2.1a), then $\boldsymbol{r}_n \to \boldsymbol{0}$ as $n \to \infty$. Now, if we denote the largest eigenvalue of B by λ_ρ and its associated eigenvector by $\boldsymbol{v}_\rho$, then the slowest possible rate of convergence will occur when Eq. (A.8) simplifies to

$$\boldsymbol{r}_n = \|\boldsymbol{r}_0\| \lambda_\rho^n \boldsymbol{v}_\rho. \tag{A.9}$$

That is, the slowest convergence occurs when $\boldsymbol{r}_0$ projects solely onto the eigenvector of matrix B associated with the eigenvalue λ_ρ that determines the spectral radius of B. Recall that since $B = I - \tilde{A}$, λ_ρ is also the eigenvalue of $I - A$ farthest from $(1, 0)$.

Next, taking the norm of both sides of Eq. (A.9) and dividing both sides by $\|\boldsymbol{r}_0\|$ gives

$$\frac{\|\boldsymbol{r}_n\|}{\|\boldsymbol{r}_0\|} = |\lambda_\rho|^n, \tag{A.10}$$

where the left-hand side gives the relative residual after the n^{th} Richardson iteration. Finally, if an iterative solver relative residual stopping criterion is specified as

$$\frac{\|\boldsymbol{r}_n\|}{\|\boldsymbol{r}_0\|} \leq \epsilon, \tag{A.11}$$

then substituting criterion (A.11) into Eq. (A.10) and solving for n gives

$$n = \frac{\log(\epsilon)}{\log|\lambda_\rho|}, \tag{A.12}$$

as the number of Richardson iterations required to satisfy residual reduction criterion (A.11).[6] A final point to emphasize here is that although the analysis above regards λ_ρ as the largest eigenvalue of $I - \tilde{A}$, in practice λ_ρ is obtained from $I - \tilde{H}$, since $I - \tilde{A}$ cannot be constructed for large systems, and the spectral radius of $I - \tilde{H}$ very closely approximates the spectral radius of $I - \tilde{A}$.

REFERENCES

[1] L.E. CARR III, C.F. Borges and F.X. Giraldo, *An Element-Based Spectrally-Optimized Approximate Inverse Preconditioner for the Euler Equations*, SIAM J. Sci. Comput., 34 (2012) pp. B392-420.

[2] 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.

[3] 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.

[4] F.X. GIRALDO, J.F. Kelly, and E.M. Constantinescu, *Implicit-Explicit formulations for a 3D nonhydrostatic unified model of the atmospheric (NUMA)*, SIAM J. Sci. Comput., 35 (2013), pp. B1162-1194.

[5] 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., Vol. 231, pp.7988-8008.

[6] Y. LIANG, *Generalized Least-Squares Polynomial Preconditioners for Symmetric Indefinite Linear Equations*, Parallel Comput., 28 (2002), 323-341.

[7] Y. LIANG, *Polynomial Preconditioner for the Solution of Linear Equations*, Ph.D. dissertation, University of Ulster, 2005.

[8] Y. SAAD, *Least Squares Polynomials in the Complex Plane and Their Use for Solving Nonsymmetric Linear Systems*, SIAM J. Numer. Anal., 24 (1987), 155-169.

[9] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia PA, 2003.

[10] 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.

[11] L.N. TREFETHEN and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia PA, 1997.

[12] 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.

[13] H.A. van der VORST, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, New York NY, 2003.

[6] To simplify the analysis here, we have assumed that λ_ρ is real, as is the case for the spectra shown in Figs. 2.1a and 4.1a. If λ_ρ is actually a complex conjugate pair, an appropriately modified mathematical analysis still leads to Eq. (A.12).