Preconditioning Spectral Element Schemes for Definite and Indefinite Problems

Yair Shapira,[1] **Moshe Israeli,**[1] **Avram Sidi,**[1] **Uzi Zrahia**[2]
[1] *Computer Science Department*
Technion, Israel Institute of Technology
Haifa 32000, Israel

[2] *Department of Mechanical Engineering*
Technion, Israel Institute of Technology
Haifa 32000, Israel

Received July 15, 1998; accepted: December 15, 1998

Spectral element schemes for the solution of elliptic boundary value problems are considered. Preconditioning methods based on finite difference and finite element schemes are implemented. Numerical experiments show that inverting the preconditioner by a single multigrid iteration is most efficient and that the finite difference preconditioner is superior to the finite element one for both definite and indefinite problems. A multigrid preconditioner is also derived from the finite difference preconditioner and is found suitable for the CGS acceleration method. It is pointed out that, for the finite difference and finite element preconditioners, CGS does not always converge to the accurate algebraic solution. © 1999 John Wiley & Sons, Inc. Numer Methods Partial Differential Eq 15: 535–543, 1999

Keywords: indefinite Helmholtz equation; multigrid; preconditioning; spectral elements

I. INTRODUCTION

Consider the elliptic symmetric second-order boundary value problem

$$-\nabla(D\nabla u) + \beta u = f, \;\; x \in \Omega \subset R^d, \tag{1}$$

where R is the set of real numbers, d is a positive integer denoting the dimension of the problem, $\Omega \subset R^d$ is a domain, and D, u, β, and f are functions of the independent variable $x \in R^d$ (D is a symmetric uniformly positive definite $d \times d$ matrix and u is the unknown function) with the boundary conditions

$$u = g_D, \quad x \in \Gamma_D \subset \partial\Omega \tag{2}$$

Correspondence to: Yair Shapira

and

$$u_n + \alpha u = g_N, \quad x \in \Gamma_N = \partial\Omega \setminus \Gamma_D, \tag{3}$$

where $\vec{n}$ is the outer normal vector and g_D, g_N and α are functions defined on $\partial\Omega$. For some particular cases, such as when Ω is a square (or a box in 3-d) and the above functions are smooth, the spectral method for the numerical solution of (1)–(3) has an exponential rate of convergence, namely, its accuracy increases exponentially with the number of degrees of freedom used (see, e.g., [1] and [2]). For more general cases, such as when Ω is a polygon and D and the other functions mentioned above are piecewise smooth in Ω, the spectral element method [3] might also yield convergence rates better than those of the usual finite element, finite volume or finite difference discretizations. However, the sparsity and conditioning of the coefficient matrices corresponding to the spectral and spectral element methods are much worse than those of more standard discretization methods. (Typically, the condition number of the spectral method for the Poisson equation is $O(N^2)$, where N is the number of degrees of freedom [2], whereas it is only $O(N^{2/d})$ for the usual finite difference scheme.) Therefore, efficient preconditioning methods for the linear systems resulting from the spectral and spectral element schemes are of great importance for the rapid and stable solution of these systems.

The following preconditioning methods are of special interest. In [4], it is suggested to use a finite difference scheme based on the spectral collocation nodes for preconditioning the spectral scheme. In [5], finite element schemes are used for this purpose. It is shown in [6] that for symmetric positive definite problems the condition number of the resulting preconditioned linear system is bounded independently of the number of degrees of freedom, which implies fast convergence of the corresponding preconditioned conjugate gradient method. Similar preconditioning methods were applied to spectral element schemes by Babuska, Mandel, Ronquist, and others. However, little success was reported in applying these methods to indefinite problems such as the indefinite Helmholtz equation.

Here we apply these preconditioning methods to spectral element schemes for the solution of definite and indefinite problems. We emphasize the importance of scaling the coefficient matrices of both the spectral element scheme and the preconditioning finite element or finite difference scheme such that both have constant main diagonals. We recommend solving the preconditioning systems approximately by one multigrid V-cycle. We observe good convergence rates for both definite and indefinite examples, the finite difference preconditioner being superior to the finite element one. A multigrid preconditioner is then derived from the finite difference preconditioner and found suitable for the Conjugate Gradient Squared (CGS) acceleration method of [7]. It is also pointed out that, for certain preconditioners, CGS does not converge to the accurate algebraic solution.

II. IMPLEMENTATION AND NUMERICAL RESULTS

Our model problem is

$$-u_{xx} - u_{yy} + \beta u = f, \quad (x, y) \in (0,1) \times (0,1) \tag{4}$$

with the boundary conditions

$$u = g_D, \quad x = 1 \text{ or } y = 1 \tag{5}$$

and

$$u_n + \alpha u = g_N, \quad x = 0 \text{ or } y = 0. \tag{6}$$

We consider the following examples:

1. Symmetric positive definite Poisson equation with Dirichlet–Neumann boundary conditions: $\beta = \alpha = 0$.
2. Indefinite Helmholtz equation with mixed complex boundary conditions: $\beta = -200$ and $\alpha = 10i$ (this example is taken from [4]).

The spectral element scheme uses sixteen elements for the uniform partitioning of the unit square (see Fig. 1). Each element uses a 9×9 grid consisting of the Gauss–Lobatto points corresponding to the Legendre polynomials of degree at most eight in each spatial direction. (Since Gauss–Lobatto points are used, some of them lie on the boundary and on the interfaces between adjacent elements.) Each element contains 64 cells (see Fig. 1) and, therefore, the whole domain contains $32^2 = 1024$ cells. Nodes that lie on boundaries on which Dirichlet conditions are imposed are not considered as variables, but their contributions are included in the right hand

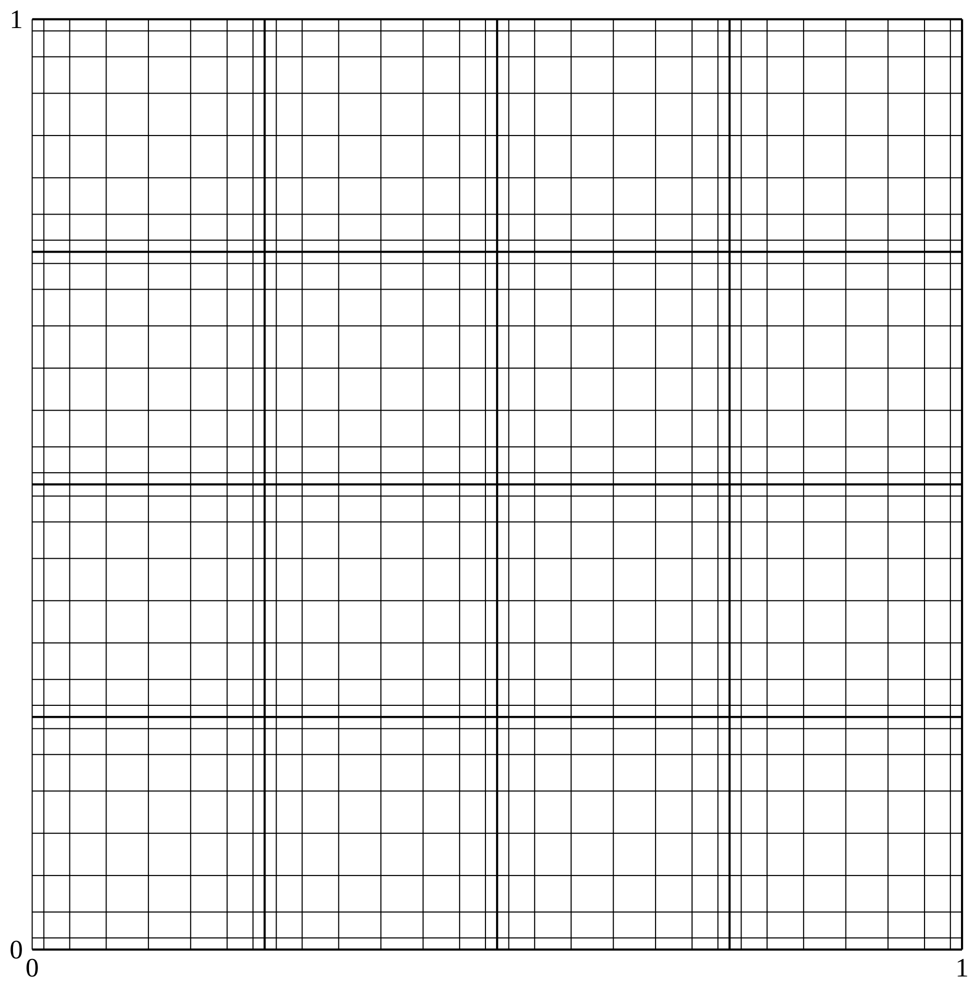

FIG. 1. The spectral element mesh consists of the 16 elements defined by the thick lines, each of which uses polynomials of degree at most eight. The bilinear finite element mesh used for preconditioning consists of the 32^2 elements defined by the thick and thin lines.

side of the linear system of equations. Consequently, the total number of variables is also 1024. The spectral element space is the space of functions continuous on the unit square, which are polynomials of degree at most eight in each of the 16 spectral elements. The resulting linear system is

$$Ax = b. \tag{7}$$

The code used for generating the coefficient matrix A is that of [9]. The preconditioned system is denoted by

$$Q^{-1}diag(Q)diag(A)^{-1}Ax = Q^{-1}diag(Q)diag(A)^{-1}b, \tag{8}$$

where Q corresponds to the finite difference or finite element scheme based on the spectral element nodes. The finite element scheme uses the bilinear finite element mesh in Fig. 1, which yields a 9-coefficient stencil. The finite difference scheme is constructed as follows. The usual 5-coefficient second-order stencil is used. At the boundary $x = 0$ (resp., $y = 0$), where Neumann or mixed boundary conditions are imposed, the grid is extended symmetrically around $x = 0$ (resp., $y = 0$) to allow a 5-coefficient stencil there and to define a first-order approximation for the normal derivative based on an external point and a boundary point. (The results may be slightly improved by using a nonsymmetric extension, that is, external points that are further away from the boundary. However, for the sake of uniformity we report results with the symmetric extension only.) Of course, the stencils at points on these boundaries cannot use these dummy external points; therefore, the 5-coefficient stencil of the finite difference scheme is combined with the discrete boundary conditions to yield stencils that use only the mesh points of Fig. 1. (It was observed that using only the discrete boundary conditions for constructing the stencils at these boundary points yields poor convergence.) The scaling used in (8) is crucial; using the naive preconditioning $Q^{-1}Ax = Q^{-1}b$ yields poor convergence.

Equation (8) is solved by the Reduced Rank Extrapolation (RRE) method of [10] and [11] using the code of [12]. More precisely, RRE(10) is used, which means restarting RRE after every 10 iterations on (8) to avoid storage difficulties. The number of RRE(10) cycles required for convergence (i.e., the number of times RRE is restarted) is reported in Table I. Each RRE(10) cycle requires the computation of 11 preconditioned iterations.

A preconditioning problem of the form

$$diag(Q)^{-1}Qe = r \tag{9}$$

is to be solved at each iteration on (8). For the finite difference preconditioning, this is done by the automatic multigrid (AutoMUG) method of [13] (using the code of [14]). AutoMUG is implemented with a V(1,1) cycle with the red-black point Gauss–Seidel relaxation at each level.

TABLE I. Number of RRE(10) cycles (under the heading "*cycles*") applied to the preconditioned linear system. Each RRE(10) cycle requires 11 preconditioned iterations. The cost of a single preconditioned iteration in terms of work units is under the heading "cost".

preconditioner	β	α	cost	*cycles*	$\|\|$error$\|\|_\infty$	$\|\|$preconditioned residual$\|\|_2$
finite differences	0	0	1.6	2	$3.8 \cdot 10^{-11}$	$7.6 \cdot 10^{-11}$
finite elements	0	0	1.9	4	$6.3 \cdot 10^{-11}$	$1.3 \cdot 10^{-10}$
multigrid	0	0	2.1	2	$1.1 \cdot 10^{-9}$	$2.0 \cdot 10^{-9}$
finite differences	-200	$10i$	1.8	8	$1.0 \cdot 10^{-11}$	$1.1 \cdot 10^{-11}$
finite elements	-200	$10i$	2.1	13	$3.6 \cdot 10^{-11}$	$3.4 \cdot 10^{-11}$
multigrid	-200	$10i$	2.3	7	$3.9 \cdot 10^{-11}$	$7.2 \cdot 10^{-11}$

For the definite example, six levels are used, which is the maximum number of levels allowed. The coarsest level consists of one variable only. For the indefinite example, only three levels are used, in light of the guidelines developed in [15] and [16]. As recommended in [16], 20 consecutive point Kacmarz relaxations are performed for the approximate solution of the coarsest grid problem within AutoMUG in the indefinite case. For the examples tested here, it was found that the most efficient implementation is to use only one AutoMUG iteration for the approximate solution of (9) (with a zero initial guess), presumably because it efficiently reduces all the Fourier modes present in the error. (Of course, for other applications it might be efficient to use more than one iteration.) This implementation is referred to as the finite difference preconditioning in the sequel.

The finite element preconditioner is also approximated, that is, it is solved approximately by a single V(1,1) multigrid cycle. In fact, we just replace the stencil used at the finest grid within AutoMUG by the 9-coefficient stencil of the finite element scheme (scaled such that its central element is equal to 1). This stencil is then used for both relaxation and residual computation at the finest level within AutoMUG. The coarse levels of AutoMUG remain unchanged, namely, they still use 5-coefficient stencils as before. The relaxation with the above 9-coefficient stencil at the finest level of AutoMUG is done by the red-black Jacobi method, that is, a point Jacobi sweep on the red-colored points followed by a point Jacobi sweep on the black-colored points. [The usual (damped) point Jacobi relaxation did not work well here.] The relaxation on the coarse levels is the red-black point Gauss–Seidel method as before. This implementation is referred to as the finite element preconditioning in the sequel.

The above implementation generates another possible approach: replace the finite difference stencil at the finest level of AutoMUG by the original spectral element stencil (scaled such that its central element is equal to 1). This stencil is then used on the finest level of AutoMUG for both residual computation and relaxation, using the above red-black point Jacobi method. [The usual (damped) point Jacobi relaxation did not work well here]. The operators on all other levels remain unchanged. Since this is an iterative method for the solution of the original problem (7), there is no need to apply it to a preconditioning problem of the form (9) but rather directly to (7), which avoids computing the residual of (7) before applying the multigrid method. The preconditioned system for this approach is of the form

$$\tilde{Q}^{-1} diag(A)^{-1} Ax = \tilde{Q}^{-1} diag(A)^{-1} b, \tag{10}$$

where $\tilde{Q}$ represents the multigrid preconditioner (see [17]) resulting from this multigrid iteration. Here we find it more efficient to use a V(0,1) multigrid cycle, the cost of which is only slightly larger than 2 work units, where a work unit is the amount of arithmetical operations required for computing the residual for the original problem (7). This approach was first tested in [14] and [18] and is referred to in the sequel as "multigrid."

The exact solution of (4)–(6) in our examples is $u = xy$. The initial guess in all our experiments is zero. In Table I, we show the number of RRE(10) cycles required for the solution of (8) (or (10) for the multigrid preconditioning). As can be seen from Table I, during this process the l_2 norm of the residual of (8) or (10) (that is, the preconditioned residual of (7)) is reduced by 10–11 orders of magnitude, as well as the l_∞ norm of the error. As discussed in [16], the preconditioned residual yields a more realistic convergence criterion than the residual of the original problem (7), since (8) and (10) are better conditioned than (7). (This is an advantage of left preconditioning over right and symmetric ones.) The l_2 norms of the preconditioned residuals are available in the RRE algorithm for no additional cost (see [12]). Replacing RRE(10) by RRE(5) (namely, restarting RRE after every 5 iterations) yields no essential change in the convergence behavior of all three methods (Table II).

TABLE II. Number of RRE(5) cycles (under the heading "*cycles*") applied to the preconditioned linear system. Each RRE(5) cycle requires 6 preconditioned iterations. The cost of a single preconditioned iteration in terms of work units is under the heading "cost".

preconditioner	β	α	cost	*cycles*	$\|\text{error}\|_\infty$	$\|\text{preconditioned residual}\|_2$
finite differences	0	0	1.6	4	$7.0 \cdot 10^{-11}$	$6.7 \cdot 10^{-11}$
finite elements	0	0	1.9	8	$7.9 \cdot 10^{-11}$	$4.5 \cdot 10^{-10}$
multigrid	0	0	2.1	5	$5.1 \cdot 10^{-11}$	$1.8 \cdot 10^{-10}$
finite differences	-200	$10i$	1.8	16	$6.2 \cdot 10^{-11}$	$5.6 \cdot 10^{-11}$
finite elements	-200	$10i$	2.1	28	$6.5 \cdot 10^{-11}$	$6.6 \cdot 10^{-11}$
multigrid	-200	$10i$	2.3	15	$6.0 \cdot 10^{-11}$	$8.8 \cdot 10^{-11}$

It was observed in [14] that restarted RRE (as well as restarted GMRES) is not optimal for certain highly indefinite problems. For these difficult cases, better convergence may be achieved by the Conjugate Gradient Squared (CGS) method of [7] (used also in [16]) or the Transpose Free Quasi Minimal Residual (TFQMR) method of [19], Algorithm 5.2 (used in [13]). Indeed, even for the original system (7) with the simple point Jacobi preconditioner, CGS converges for the indefinite example to 10 orders of magnitude accuracy in 325 iterations, whereas RRE(5) and RRE(10) practically stagnate. (It is possible that this could be fixed by restarting RRE after performing several initial iterations as in [20]. We did not use this strategy here, because it could cause numerical explosion when the sensitive preconditioners are used.) The advantage of CGS for indefinite examples is also indicated in Table III, with the present finite difference, finite element, and multigrid preconditioners. However, it turns out that CGS (and also TFQMR) does not always converge to the accurate solution of (7), and its accuracy depends on the particular preconditioner used. For the multigrid preconditioner, no special treatment is necessary and CGS always converges accurately. For the finite difference and finite element preconditioners, however, it is impossible to reduce the error (for the definite example) by more than six orders of magnitude. To obtain the accuracy reported in Table III for these preconditioners, we had to replace the red-black point Gauss–Seidel smoother used at the coarse levels within AutoMUG (from the third level on) by the more stable point Kacmarz relaxation. A yet severer phenomenon was observed when we tried to solve (9) more accurately by an inner accelerated AutoMUG iteration; both CGS and TFQMR applied to (8) did not converge to more than three orders of magnitude accuracy. We suppose that the reason for this is that, because of roundoff errors, the direction vectors in CGS and TFQMR are no longer related to the approximate solution $\tilde{x}$ through the theoretical relation

$$Q^{-1} diag(Q) diag(A)^{-1}(A\tilde{x} - b).$$

TABLE III. Number of CGS iterations (under the heading "*iterations*") applied to the preconditioned linear system. Each CGS iteration requires two preconditioned iterations. The cost of a single preconditioned iteration in terms of work units is under the heading "cost".

preconditioner	β	α	cost	*iterations*	$\|\text{error}\|_\infty$	$\|\text{preconditioned residual}\|_2$
finite differences	0	0	1.6	14	$1.5 \cdot 10^{-11}$	$5.4 \cdot 10^{-11}$
finite elements	0	0	1.9	28	$2.4 \cdot 10^{-12}$	$3.1 \cdot 10^{-11}$
multigrid	0	0	2.1	13	$1.7 \cdot 10^{-11}$	$3.6 \cdot 10^{-11}$
finite differences	-200	$10i$	1.8	29	$1.5 \cdot 10^{-13}$	$4.3 \cdot 10^{-13}$
finite elements	-200	$10i$	2.1	50	$1.4 \cdot 10^{-12}$	$8.6 \cdot 10^{-12}$
multigrid	-200	$10i$	2.3	26	$4.2 \cdot 10^{-12}$	$4.0 \cdot 10^{-11}$

In our CGS implementation, the l_2 norm of the residuals is computed automatically from the CGS algorithm. These scalars converge nicely, and the user might have no idea that he has the wrong solution. In fact, the maximum norm of the error vectors does not converge to zero. In our TFQMR implementation we compute the residuals explicitly after each iteration and calculate their l_2 norm; these scalars also converge to a positive scalar. This leads us to believe that the methods suffer from numerical instability when the above finite difference and finite element preconditioners are used. We do not expect that using right preconditioning rather than left preconditioning would help here.

The code used for CGS is that of [16]. Each CGS iteration requires two preconditioned iterations. One additional preconditioned iteration is performed in the beginning of the CGS algorithm for computing the initial preconditioned residual, which serves as an initial direction vector. The l_2 norms of the preconditioned residuals in the CGS iteration are available from the CGS algorithm for no additional cost (see [19]).

Finally, we report a slight deterioration in the convergence rates when the problem size is increased. Specifically, we have increased the number of degrees of freedom per spectral element from $8^2 = 64$. to $16^2 = 256$. For the definite example ($\beta = 0$), RRE(5) requires six cycles to converge to ten orders of magnitude accuracy (with the finite difference preconditioner). We suppose that this deterioration is because of a larger number of error components (corresponding to eigenvectors of the iteration matrix) that have to be annihilated by the outer acceleration. CGS also suffers from a similar deterioration in the number of iterations needed for convergence when the number of degrees of freedom per element is increased to $16^2 = 256$. We expect that the preconditioned conjugate gradients method may be more robust here than CGS and restarted RRE, since it uses Krylov subspaces of unlimited dimension. Next, we have also employed this spectral element scheme, again with polynomials of degree seventeen, for the indefinite example ($\beta = -200$) and observed similar deterioration in the convergence rate in comparison with the results in Table III. Here we had to increase the number of point Kacmarz relaxations on the coarsest (namely, third) level in AutoMUG to 140 in order to achieve convergence to six orders of magnitude accuracy in 22 CGS iterations with the multigrid preconditioner. The reason for this necessity is probably that, with the increase in problem size, the third level contains four times as many degrees of freedom as before, which, in view of the high indefiniteness in the third-level equation, requires many Kacmarz relaxations to reduce the error modes corresponding to that level. Indeed, when $\beta = -400$ is used in this experiment, 200 Kacmarz relaxations are needed on the third level to achieve a similar convergence rate. (Block Kacmarz relaxation may also be considered, although it is not used here.)

III. DISCUSSION

In this article, we test three preconditioners for a spectral element scheme for the solution of certain elliptic PDEs in the unit square, namely, the Poisson equation with Dirichlet–Neumann boundary conditions and the indefinite Helmholtz equation with Dirichlet-mixed complex boundary conditions. The first two preconditioners are based on known ideas, that is, finite difference and finite element schemes based on the spectral mesh. The new implementation here uses a single V(1,1) multigrid iteration as an approximate solver for the finite difference and finite element preconditioners. It turns out that this implementation is more efficient than solving the finite difference and finite element schemes accurately. Furthermore, it works well also for the indefinite example, with CGS or restarted RRE as the outer acceleration method.

For the indefinite example, CGS turns out to be somewhat more efficient than restarted RRE as the acceleration technique. However, some care is needed when CGS (or TFQMR) is employed. In particular, the multigrid cycle for the approximate solution of the finite difference and finite element preconditioning schemes must use the stable Kacmarz smoother rather than more common smoothers, or the outer CGS iteration would not converge to the accurate algebraic solution, presumably because of numerical roundoff errors in the CGS algorithm. This difficulty can be overcome by using a new preconditioner, that is, a multigrid preconditioner that is constructed directly for the original spectral element scheme. In this multigrid preconditioner, the fine-grid residual is computed from the original spectral element scheme and the error is smoothed by applying the point red-black Jacobi smoother to that scheme, and the finite difference operators are used on the coarse grids as before. With this preconditioner, the outer CGS iteration suffers no instability and converges nicely to the algebraic solution for both definite and indefinite examples.

References

1. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, Spectral methods in fluid dynamics, Springer–Verlag, Berlin, Heidelberg, 1988.
2. D. Gottlieb and S. A. Orszag, Numerical analysis of spectral methods, Soc Indust Appl Math, Philadelphia, 1977.
3. A. T. Patera, A spectral element method for fluid dynamics; Laminar flow in a channel expansion, J Comp Phys 54 (1984).
4. S. A. Orszag, Spectral methods for problems in complex geometries, J Comp Phys 37 (1980), 70–92.
5. M. Deville and E. Mund, Chebyshev pseudospectral solution of second order elliptic equations with finite element preconditioning, J Comp Phys 60 (1985), 517–533.
6. A. Quarteroni and E. Zampieri, Finite element preconditionering for Legendre spectral collocation approximations to elliptic equation and systems, SIAM J Num Anal 29 (1992), 917-936.
7. P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J Sci Statist Comp 10 (1989), 36–52.
8. R. W. Freund, Conjugate gradients type methods for linear systems with complex symmetric coefficient matrices, SIAM J Sci Statist Comp 13 (1992), 425–448.
9. U. Zrahia, Space time spectral elements for dynamic analysis of composite laminates, D.Sc. thesis, Technion, Israel Inst Tech, Haifa, Israel, 1993.
10. R. P. Eddy, "Extrapolating to the limit of a vector sequence," Information linkage between applied mathematics and industry, P. C. C. Wang (Editor), Academic, New York, 1970, pp. 387–396.
11. M. Mešina, Convergence acceleration for the iterative solution of the equations $X = AX + f$, Comp Meth Appl Mech Eng 10 (1977), 165–173.
12. A. Sidi, Efficient implementation of minimal polynomial and reduced rank extrapolation methods, J Comp Appl Math 36 (1991), 305–337.
13. Y. Shapira, M. Israeli, and A. Sidi, Towards automatic multigrid algorithms for SPD, nonsymmetric, and indefinite problems, SIAM J Sci Comp 17 (1996), 439–453.
14. Y. Shapira, Iterative solution of elliptic PDEs and implementation on parallel computers, D.Sc. thesis, Technion, Israel Inst Tech, Haifa, Israel, 1993.
15. Y. Shapira, Multigrid methods for 3-D definite and indefinite problems, Appl Numer Math 26 (1998), 377–398.

16. Y. Shapira, Analysis of matrix-dependent multigrid algorithms, Numer Linear Algebra Appl, 5 (1998), 165–201.

17. W. Hackbusch, Multigrid methods and applications, Springer–Verlag, Berlin, Heidelberg, 1985.

18. Y. Shapira, M. Israeli, and A. Sidi, An automatic multigrid method for the solution of sparse linear systems, Sixth Copper Mnt Multigrid Meth, N. D. Melson, S. F. McCormick, and T. A. Manteuffel (Editors), NASA, Langley Research Center, Hampton, VA, 1993, pp. 567–582.

19. R. W. Freund, Transpose free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J Sci Statist Comp 14 (1993), 470–482.

20. A. Sidi and Y. Shapira, Upper bounds for convergence rates of acceleration methods with initial iterations, Numer Algorithms 18 (1998), 113–132.