

Hermite Collocation Solution of Partial Differential Equations via Preconditioned Krylov Methods

Stephen H. Brill
Department of Mathematics and Computer Science
Boise State University
Boise, Idaho, USA

This is a preprint of an article published in *Numerical Methods for Partial Differential Equations*, Vol. 17, No. 2, pp. 120-136, March 2001. © copyright 2001 John Wiley & Sons, Inc.

Received June 1, 2000

We are concerned with the numerical solution of partial differential equations (PDEs) in two spatial dimensions discretized via Hermite collocation. To efficiently solve the resulting systems of linear algebraic equations, we choose a Krylov subspace method. We implement two such methods: Bi-CGSTAB [1] and GMRES [2]. In addition, we utilize two different preconditioners: one based on the Gauss-Seidel method with a block red-black ordering (RBGS); the other based upon a block incomplete LU factorization (ILU). Our results suggest that, at least in the context of Hermite collocation, the RBGS preconditioner is superior to the ILU preconditioner and that the Bi-CGSTAB method is superior to GMRES. © 2001 John Wiley & Sons, Inc.

Keywords: collocation, preconditioned Krylov

1. INTRODUCTION

In this paper, we study the numerical solution of the PDE

$$\mathcal{L}[u](x, y) = A \frac{\partial^2 u}{\partial x^2} + B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^2 u}{\partial y^2} + D \frac{\partial u}{\partial x} + E \frac{\partial u}{\partial y} + Fu = H \quad (1.1)$$

with Dirichlet and/or Neumann boundary conditions, discretized by Hermite collocation. (The coefficients A , B , C , D , E , F , and H in (1.1) may all depend upon x and y .) The matrix corresponding to the resulting system of linear algebraic equations is not symmetric, positive definite, or diagonally dominant, and no one method is generally recognized as being best for solving such systems.

Methods of solution other than those presented in this paper include a direct method introduced in [3] and a block SOR (successive overrelaxation) method proposed in [4]. The direct method of [3] applies to PDEs less general than (1.1) and requires uniform

meshes in each of the coordinate directions, a restriction that does not apply to our methods. (We did, however, use uniform meshes in our example problems.)

In [4], only Poisson's equation is studied and the best methods of solution reported are the block SOR method and GMRES with block diagonal preconditioners. Because the coefficient matrices are not diagonally dominant, our Gauss-Seidel preconditioner will be far superior to their preconditioners. Also, as our numerical results show, Bi-CGSTAB is a much better choice than GMRES.

This paper is organized as follows. We first describe Hermite collocation, followed by the numberings of equations and unknowns that we employ. We then describe our two preconditioners and provide results of numerical experiments. A short section summarizing our results concludes this work.

II. HERMITE COLLOCATION

We begin with an introduction to Hermite collocation which, unlike finite difference methods or many other finite element methods, provides solutions that are C^1 continuous. More detailed derivations of the material in this section may be found in [4], [5], [6], and [7].

Let u be a function of x and y defined on a rectangular domain $\mathcal{D} = [a_x, b_x] \times [a_y, b_y]$. If we define $X = \{a_x = x_0 < x_1 < x_2 < \dots < x_{m_x} = b_x\}$ for the x -direction and $Y = \{a_y = y_0 < y_1 < y_2 < \dots < y_{m_y} = b_y\}$ for the y -direction, then the set of points $\{(x_q, y_r) : x_q \in X \text{ and } y_r \in Y\}$ defines a set of nodes in \mathcal{D} . These nodes partition \mathcal{D} into $m_x m_y$ rectangular finite elements. Then the bi-cubic piecewise polynomial interpolating the values $u_{q,r} = u(x_q, y_r)$, $u_{q,r}^x = \frac{\partial u}{\partial x}(x_q, y_r)$, $u_{q,r}^y = \frac{\partial u}{\partial y}(x_q, y_r)$, and $u_{q,r}^{xy} = \frac{\partial^2 u}{\partial x \partial y}(x_q, y_r)$ for $q = 0, 1, 2, \dots, m_x$ and $r = 0, 1, 2, \dots, m_y$ is

$$\hat{u}(x, y) = \sum_{q=0}^{m_x} \sum_{r=0}^{m_y} [u_{q,r} f_q(x) f_r(y) + u_{q,r}^x g_q(x) f_r(y) + u_{q,r}^y f_q(x) g_r(y) + u_{q,r}^{xy} g_q(x) g_r(y)]. \quad (2.1)$$

If z represents both x and y and $h_j = z_j - z_{j-1}$, the Hermite basis functions, defined for $\eta \in [-\frac{1}{2}, \frac{1}{2}]$, are

$$f_j(z) = \begin{cases} \frac{1}{2}(1+2\eta)^2(1-\eta), & z_{j-1} \leq z = z_j + (\eta - \frac{1}{2})h_j \leq z_j \\ \frac{1}{2}(1-2\eta)^2(1+\eta), & z_j \leq z = z_j + (\eta + \frac{1}{2})h_{j+1} \leq z_{j+1} \\ 0, & \text{otherwise} \end{cases} \quad (2.2)$$

and

$$g_j(z) = \begin{cases} \frac{h_j}{8}(2\eta+1)^2(2\eta-1), & z_{j-1} \leq z = z_j + (\eta - \frac{1}{2})h_j \leq z_j \\ \frac{h_{j+1}}{8}(2\eta-1)^2(2\eta+1), & z_j \leq z = z_j + (\eta + \frac{1}{2})h_{j+1} \leq z_{j+1} \\ 0, & \text{otherwise.} \end{cases} \quad (2.3)$$

Note that the interpolation works because $f_j(z_k) = \delta_{jk}$, $\frac{df_j}{dz}(z_k) = 0$, $g_j(z_k) = 0$, and $\frac{dg_j}{dz}(z_k) = \delta_{jk}$. Here δ_{jk} is the Kronecker symbol.

We are given the PDE (1.1) defined on \mathcal{D} , along with Dirichlet and/or Neumann boundary conditions. If we introduce (2.1) into (1.1), we obtain

$$\mathcal{L}[\hat{u}](x, y) - H(x, y) = E(x, y), \quad (2.4)$$

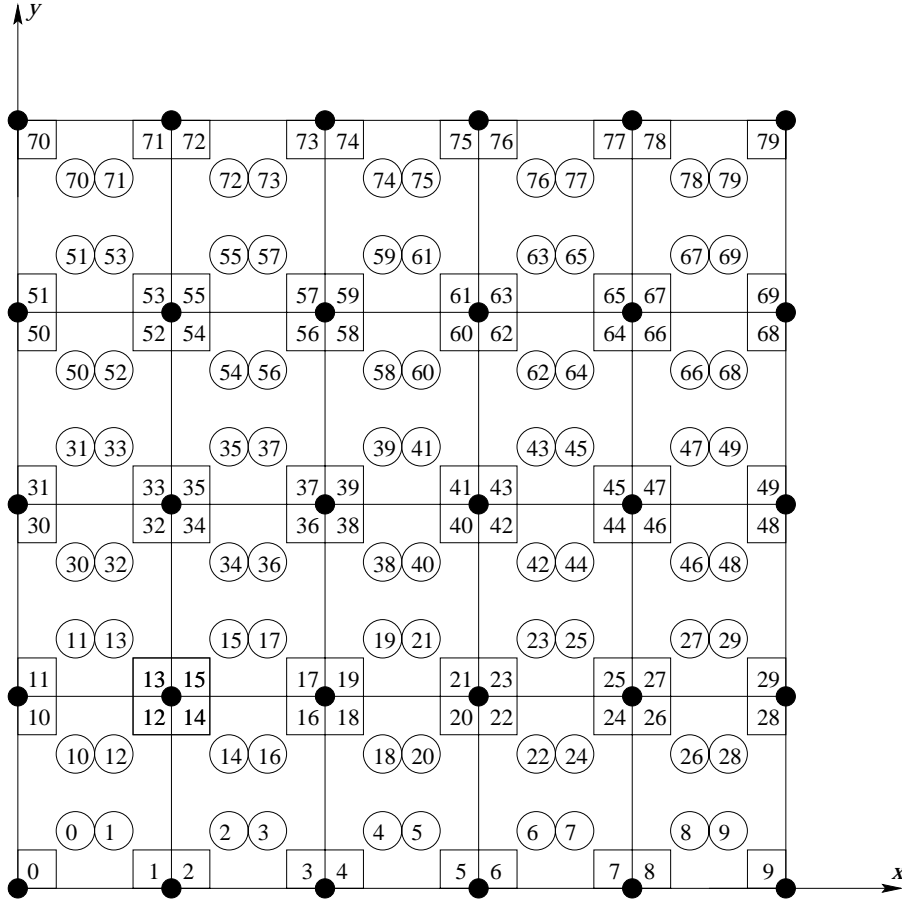


FIG. 1. Numbering of equations and unknowns for ILU preconditioner

and

$$A_{RBGS} = \left[\begin{array}{cc|cc} A_F & & B_F & \\ & A_1 & C_0 & B_1 \\ \hline & & A_L & C_L \\ C_F & B_0 & A_0 & \\ & C_1 & B_L & A_2 \end{array} \right] = \left[\begin{array}{c|c} R & U \\ \hline L & B \end{array} \right].$$

Note that the blocks in A_{ILU} and in A_{RBGS} are identical; only their order has been permuted. Note also that the diagonal blocks A_j , $j = F, 0, 1, 2, \dots, m_y - 2, L$, all have a block tridiagonal structure. When $j = F, L$, these blocks are 2×2 matrices; when $j = 0, 1, 2, \dots, m_y - 2$, the blocks are 4×4 matrices.

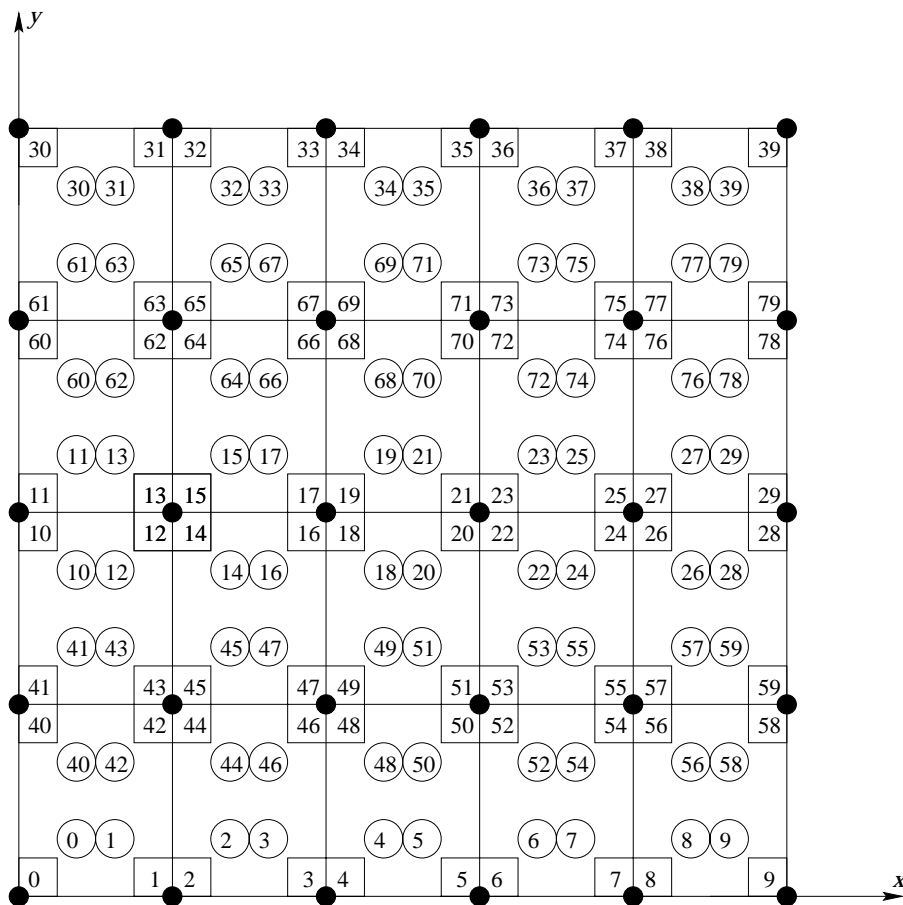


FIG. 2. Numbering of equations and unknowns for RBGS preconditioner

IV. THE PRECONDITIONERS

In order to obtain rapid convergence using either the Bi-CGSTAB or GMRES methods to solve (3.1), a suitable preconditioner P must be chosen. In these methods, the action of P is manifested by solving linear systems of the form

$$P\mathbf{y} = \mathbf{c}. \quad (4.1)$$

The preconditioning matrix P is chosen such that $P \approx A$ and (4.1) is easily solved.

In the Bi-CGSTAB method, every step of the form (4.1) is immediately followed by a matrix-vector multiplication of the form

$$\mathbf{v} = A\mathbf{y}. \quad (4.2)$$

We notice that we need not perform the entire matrix-vector multiplication (4.2). Let $E = A - P$. Then it is easily seen that

$$\mathbf{v} = \mathbf{c} + E\mathbf{y}. \quad (4.3)$$

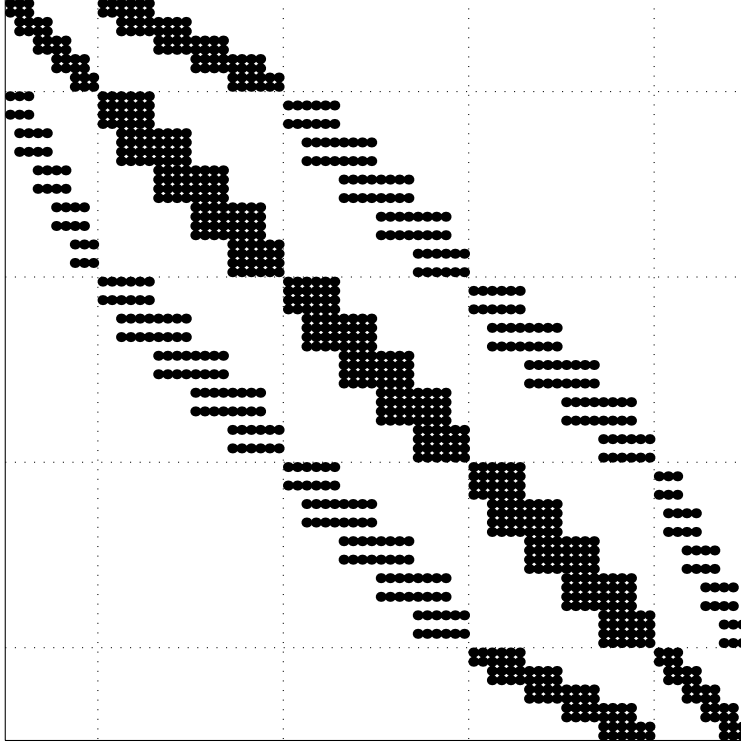


FIG. 3. Structure of matrix A for ILU preconditioner

This fact will be exploited below.

We consider two choices for P : a Red-Black Gauss-Seidel (RBGS) preconditioner and an Incomplete LU (ILU) preconditioner.

A. Red-Black Gauss-Seidel Preconditioner

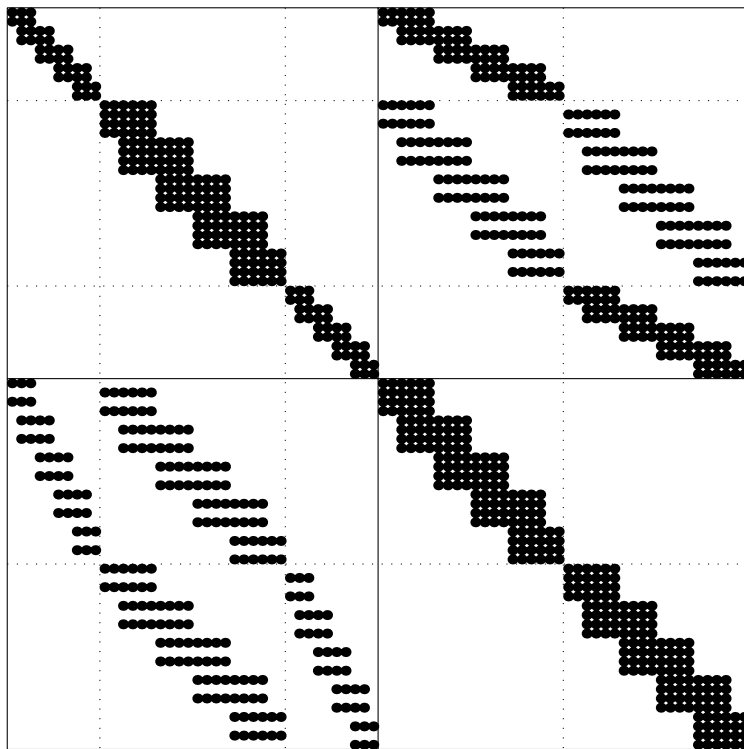
The RBGS preconditioner is

$$P_{RBGS} = \left[\begin{array}{ccc|ccc} A_F & & & & & \\ & A_1 & & & & \\ & & A_L & & & \\ \hline C_F & B_0 & & A_0 & & \\ & & C_1 & B_L & & A_2 \end{array} \right] = \left[\begin{array}{c|c} R & \\ \hline L & B \end{array} \right].$$

Note that this matrix partitioning implies that m_y is an even number. To solve (4.1) with this preconditioner, we write

$$\begin{bmatrix} R \\ L & B \end{bmatrix} \begin{bmatrix} \mathbf{y}_R \\ \mathbf{y}_B \end{bmatrix} = \begin{bmatrix} \mathbf{c}_R \\ \mathbf{c}_B \end{bmatrix}.$$

We first solve $R\mathbf{y}_R = \mathbf{c}_R$, which is easily accomplished because all the nontrivial entries of R are in block tridiagonal matrices on its main diagonal. Once we have obtained \mathbf{y}_R ,

FIG. 4. Structure of matrix A for RBGS preconditioner

we solve $B\mathbf{y}_B = \mathbf{c}_B - L\mathbf{y}_R$, which again is straightforward because all the nontrivial entries of B are in block tridiagonal matrices on its main diagonal.

Recall now matrix-vector multiplication via (4.3). For the RBGS preconditioner, this may be written

$$\mathbf{v} = \begin{bmatrix} \mathbf{c}_R + U\mathbf{y}_B \\ \mathbf{c}_B \end{bmatrix}.$$

This is much less expensive than computing \mathbf{v} via (4.2), that is:

$$\mathbf{v} = \begin{bmatrix} R\mathbf{y}_R + U\mathbf{y}_B \\ L\mathbf{y}_R + B\mathbf{y}_B \end{bmatrix}.$$

B. Incomplete LU (ILU) Preconditioner

One of the attractive properties of the ILU preconditioner P is that one utilizes the sparsity structure of matrix A in both the construction and storage of P . The specifics of both these issues are discussed below.

Construction The construction of the ILU preconditioner is based upon the well-known Crout LU factorization for tridiagonal matrices (see, for example, [11]). This factorization is $A = LU$, where matrix A is factored into the product of the lower triangular matrix L and unit upper triangular matrix U . For the case of a 4×4 matrix A , the factorization is:

$$\begin{bmatrix} a_1 & b_1 & & \\ c_1 & a_2 & b_2 & \\ & c_2 & a_3 & b_3 \\ & & c_3 & a_4 \end{bmatrix} = \begin{bmatrix} a_1 & & & \\ c_1 & \bar{a}_2 & & \\ & c_2 & \bar{a}_3 & \\ & & c_3 & \bar{a}_4 \end{bmatrix} \begin{bmatrix} 1 & \bar{b}_1 & & \\ & 1 & \bar{b}_2 & \\ & & 1 & \bar{b}_3 \\ & & & 1 \end{bmatrix}. \quad (4.4)$$

Note that the entries a_1 and all the c_j 's appear unchanged from matrix A to matrix L . Note also that A can be overwritten by the relevant information in matrices L and U .

The ILU preconditioner is constructed analogously. We obtain

$$A \approx P = LU, \quad (4.5)$$

where A is given in (3.2),

$$L = \begin{bmatrix} A_F & & & & \\ C_F & \bar{A}_0 & & & \\ & C_0 & \bar{A}_1 & & \\ & & C_1 & \bar{A}_2 & \\ & & & C_L & \bar{A}_L \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} I & \bar{B}_F & & & \\ & I & \bar{B}_0 & & \\ & & I & \bar{B}_1 & \\ & & & I & \bar{B}_L \\ & & & & I \end{bmatrix}.$$

Here the symbol I represents the identity matrix of appropriate size.

We now discuss solving (4.1) with this choice of P . We first solve $Lz = c$, i.e.,

$$\begin{bmatrix} A_F & & & & \\ C_F & \bar{A}_0 & & & \\ & C_0 & \bar{A}_1 & & \\ & & C_1 & \bar{A}_2 & \\ & & & C_L & \bar{A}_L \end{bmatrix} \begin{bmatrix} z_F \\ z_0 \\ z_1 \\ z_2 \\ z_L \end{bmatrix} = \begin{bmatrix} c_F \\ c_0 \\ c_1 \\ c_2 \\ c_L \end{bmatrix}.$$

We first solve $A_F z_F = c_F$. Now that we have z_F , we may solve $\bar{A}_0 z_0 = c_0 - C_F z_F$, obtaining z_0 . We continue in this manner, finally obtaining z_L . Because the diagonal blocks of L are block tridiagonal, solving these subsystems is easily accomplished. Now that we have z , we solve $Uy = z$, i.e.,

$$\begin{bmatrix} I & \bar{B}_F & & & \\ & I & \bar{B}_0 & & \\ & & I & \bar{B}_1 & \\ & & & I & \bar{B}_L \\ & & & & I \end{bmatrix} \begin{bmatrix} y_F \\ y_0 \\ y_1 \\ y_2 \\ y_L \end{bmatrix} = \begin{bmatrix} z_F \\ z_0 \\ z_1 \\ z_2 \\ z_L \end{bmatrix}.$$

We first see that $y_L = z_L$. Then we obtain $y_2 = z_2 - \bar{B}_L y_L$. We continue in this manner, finally obtaining y_F .

Note that the relationship in (4.5) is approximate equality, not exact equality. This is because we have maintained the philosophy of the Crout factorization (4.4) whereby we enforce that the sparsity structure of L and U mirrors that of A . To have exact equality in (4.5) would require that the blocks with the overscore decoration in L and U

have many more non-zero entries than their corresponding blocks of A . The entries in the blocks of L and U with the overscore are determined by requiring that the product $P = LU$ may differ from A only outside the sparsity structure of A . That is, if a particular entry a_{ij} of A is non-zero, then L and U are computed by requiring that the corresponding entry in the product LU is also equal to a_{ij} . If, however, a particular entry of A is zero, then the corresponding entry in the product LU need not also be zero.

Storage Considerations and Their Consequences Both the Bi-CGSTAB and GMRES methods require matrix-vector multiplications involving matrix A . Thus, if we overwrite the entries of A by those of L and U , we lose the ability to readily calculate with A . We have examined three alternatives to deal with this issue:

- Strategy A: Form L and U without overwriting A . Thus we may easily solve linear systems with matrix P and easily multiply with matrix A . The disadvantage of this is, of course, that we need to store both A and P explicitly.
- Strategy B: Overwrite A with L and U but recompute A from L and U when necessary. This is the best choice from the point of view of minimizing storage requirements. However, it is a poor choice from the perspective of minimizing solving time because in this case A must be regenerated twice per Bi-CGSTAB or GMRES iteration.
- Strategy C: Overwrite A with L and U but also calculate (and store) the error matrix E from (4.3). This strategy applies only to Bi-CGSTAB because GMRES does not have steps of the form (4.1) followed by steps of the form (4.2). The disadvantages here are the expense of the calculation of E and the need to store both P (which overwrites A) and E , whose structure is depicted in Figure 5.

It is obvious that Strategy B should be rejected if minimizing solving time is our main concern. So we now compare Strategies A and C, which differ in two ways:

- Both strategies require matrix-vector multiplication. However, Strategy A uses A while Strategy C uses E . It is easily (but tediously) shown that the number of non-zero entries of A is $64m_xm_y - 32m_x - 32m_y + 16$ while the number of non-zero entries of E is $64m_xm_y - 64m_x - 80m_y + 72$. Thus multiplying with E instead of A requires $32m_x + 48m_y - 56$ fewer scalar multiplications per matrix-vector multiplication.
- Strategy C requires generation of matrix E . No such expense is needed in Strategy A.

So, if the number of iterations required for convergence is sufficiently large, then the savings produced by using E instead of A for the two matrix-vector multiplications per Bi-CGSTAB iteration will be large enough to justify the expense of constructing E .

Since E has $\mathcal{O}(m_xm_y)$ non-zero entries, the expense of constructing it is $\mathcal{O}(m_xm_y)$ also. Since the savings of using E instead of A is only $M = \max\{\mathcal{O}(m_x), \mathcal{O}(m_y)\}$ per matrix-vector multiplication, the number of iterations required for convergence must be at least on the order of M to justify the expense of constructing E .

However, in examining Table I, we see that it is rarely worth the expense of computing E and results in negligible savings when the expense is worthwhile. This table shows the amount of time (in seconds) required for Bi-CGSTAB to reach convergence

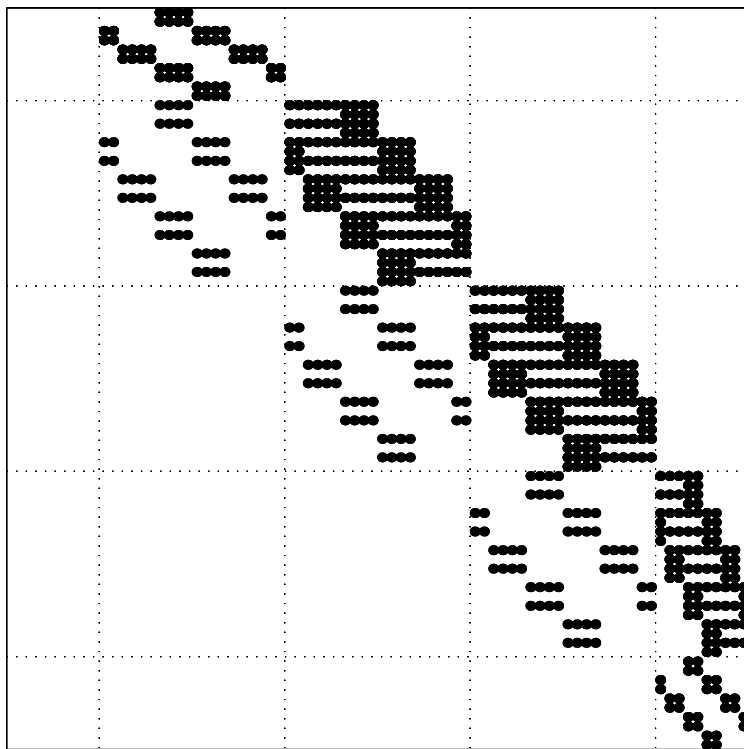


FIG. 5. Structure of matrix E associated with ILU preconditioner

for three model problems using the three strategies outlined above. (The specifics of these problems are discussed in the next section.) Convergence is considered to have occurred when the residual vector $\mathbf{b} - \mathbf{A}\mathbf{x}_n$ has infinity norm less than 10^{-8} . Here \mathbf{x}_n is the value of the solution vector \mathbf{x} at the conclusion of the n th Bi-CGSTAB iteration. All these examples were run with $m = m_x = m_y$. Four runs were performed for each model problem and for each value of m . The average of the times of these runs is reported in Table I.

As we can see in Table I, Strategy C is faster than Strategy A only in two cases: when $m = 20$ or 40 in Example 1. It is also apparent that using Bi-CGSTAB/ILU with Strategy B is roughly twice as expensive as using Strategies A or C. Because of these observations, we will elect to use only Strategy A when we report on our results in the next section.

V. NUMERICAL EXPERIMENTS AND RESULTS

In this section we compare Bi-CGSTAB to GMRES and the RBGS preconditioner to the ILU preconditioner using the same three example problems whose results were cited

TABLE I. Comparison of the three Bi-CGSTAB strategies with respect to three model problems with various problem sizes.

	Strategy	$m = 10$	$m = 20$	$m = 30$	$m = 40$
Example 1	A	0.0632	0.7792	2.7328	7.7203
	B	0.1278	1.3674	5.0887	13.6850
	C	0.0658	0.7565	2.9806	7.7130
Example 2	A	0.1412	1.3279	5.3282	15.0665
	B	0.2555	2.6392	9.7671	26.3386
	C	0.1475	1.4239	5.5911	15.5682
Example 3	A	0.0711	0.7416	3.3176	7.8091
	B	0.1363	1.4164	7.1282	14.5833
	C	0.0739	0.7924	3.3416	8.2669

in Table I for Bi-CGSTAB/ILU. There are four different combinations to consider: Bi-CGSTAB/RBGS, GMRES/RBGS, Bi-CGSTAB/ILU, and GMRES/ILU. The results clearly show that Bi-CGSTAB/RBGS is the best of these four.

There is a complication due to the fact that a “restart” parameter must be specified in the practical implementation of GMRES. This method requires the storage of and computations using a finite sequence of k mutually orthogonal vectors of length $4m_x m_y$. (The user selects k . After the vectors in the sequence number k , all these vectors are discarded and the computations begin anew with the current approximation to the solution.) If k is too large, the storage and extra computations make obtaining the solution computationally expensive. If k is too small, then convergence can be very slow. We implement both versions of GMRES with $k = 10, 20, 30$, and 40 . We denote the method as GMRES(k).

For each of the three examples, convergence is considered to have occurred when the residual vector has infinity norm less than 10^{-8} . We use four different problem sizes: $m = m_x = m_y = 10, 20, 30$, and 40 . The domain for all examples is the unit square $[0, 1] \times [0, 1]$. We use Strategy A for all examples using the ILU preconditioner. We report the average time of four runs for each method for each example in our results below.

A. Example 1

Here we solve Poisson’s equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \quad (5.1)$$

with Dirichlet boundary conditions on all sides. The function $f(x, y)$ and the boundary conditions are chosen so that the exact solution of (5.1) is $u(x, y) = \sin \pi x \cos \pi y$.

The results appear in Figure 6 for the RBGS preconditioner and in Figure 7 for the ILU preconditioner. We see that for the RBGS preconditioner, all the methods perform roughly the same for all problem sizes, with the exception of GMRES(10), which is noticeably worse. For the ILU preconditioner, Bi-CGSTAB is clearly the fastest for $m = 20, 30$, and 40 . Comparison of the vertical scale of Figures 6 and 7 clearly shows that RBGS is much faster than ILU.

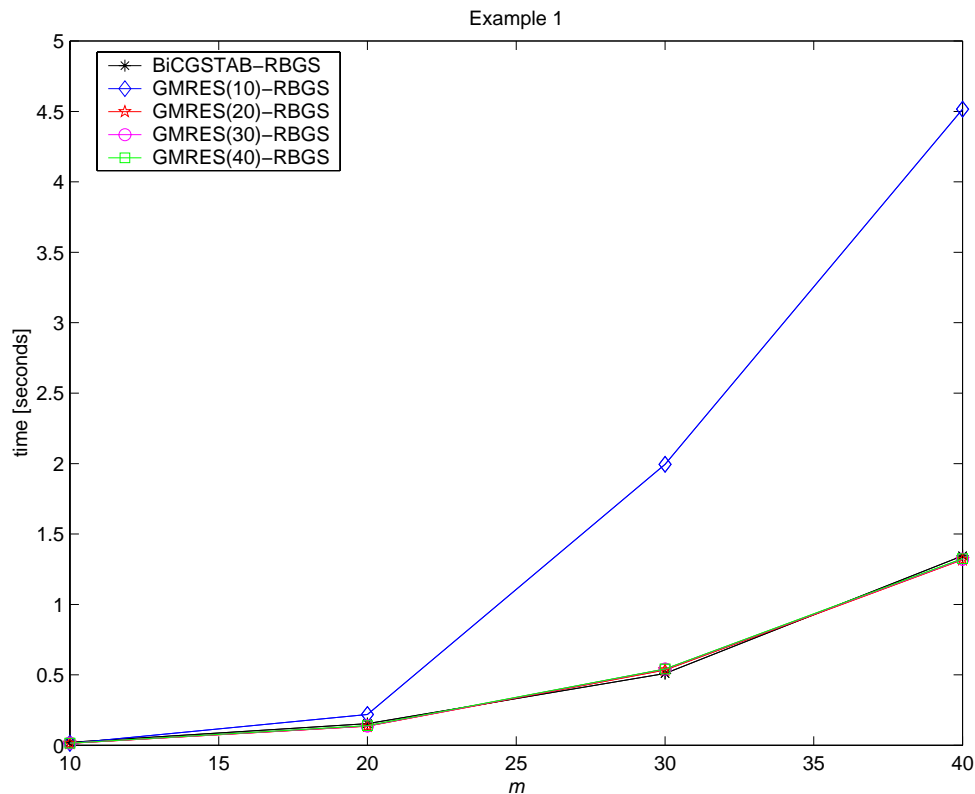


FIG. 6. Results for RBGS preconditioner

B. Example 2

Here we solve the PDE (1.1) with $A = \sin xy$, $B = xy$, $C = \exp(x + y)$, $D = x$, $E = y$, $F = 3$, $H = 3xy$ and with Dirichlet boundary conditions on all sides such that the exact solution is $u(x, y) = xy$. The results appear in Figures 8 and 9.

Here we see that for both preconditioners, the Bi-CGSTAB method performs much better than do all versions of GMRES. Comparing the vertical scales of the figures reveals again that RBGS performs much better than ILU.

C. Example 3

Here we solve the steady-state flow problem

$$\frac{\partial}{\partial x} \left(K_x \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_y \frac{\partial u}{\partial y} \right) = 0. \quad (5.2)$$

Both derivatives are expanded using the product rule: thus we are solving (1.1) with $A = K_x$, $C = K_y$, $D = \frac{\partial K_x}{\partial x}$, $E = \frac{\partial K_y}{\partial y}$, $B = F = H = 0$. The boundary conditions used are

$$u(0, y) = 75 + 25 \cos 2\pi y$$

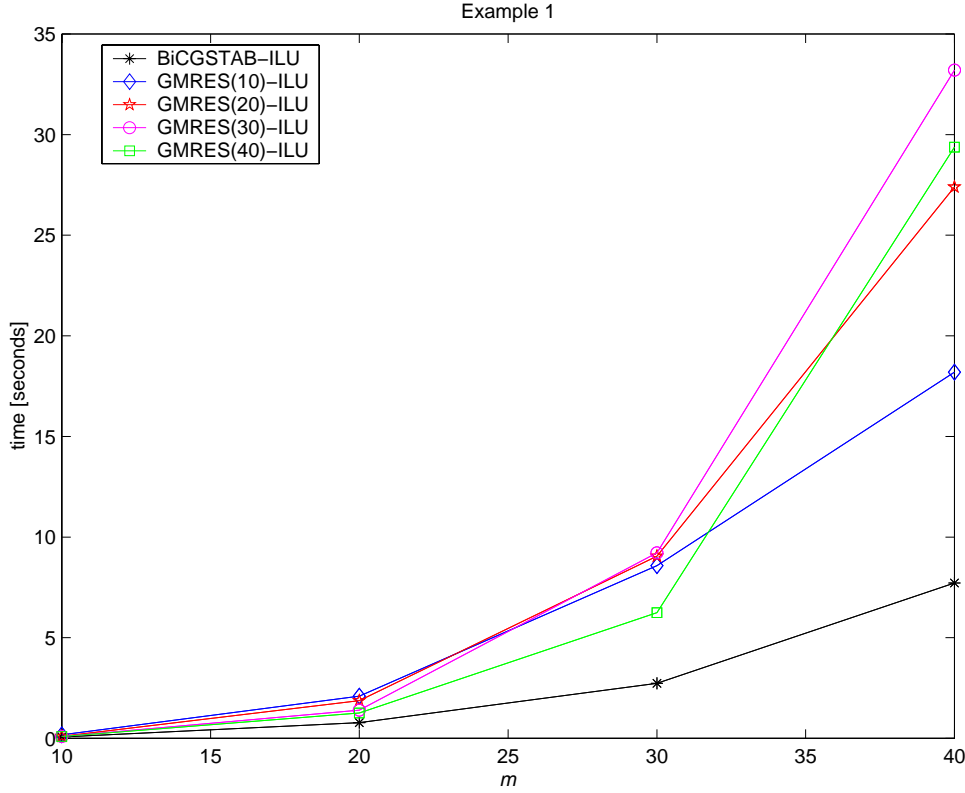


FIG. 7. Results for ILU preconditioner

$$\begin{aligned}
 u(1, y) &= 0 \\
 \frac{\partial u}{\partial y}(x, 0) &= 0 \\
 \frac{\partial u}{\partial y}(x, 1) &= 0.
 \end{aligned}$$

The unit square is divided into three separate regions, as indicated in Figure 10, thus providing a heterogeneous domain. Each of the three regions, however, is homogeneous. In Region A, we stipulate that $K_x = 0.3$, $K_y = 0.2$, and thus $\frac{\partial K_x}{\partial x} = \frac{\partial K_y}{\partial y} = 0$. In Region B, we stipulate that $K_x = 0.2$, $K_y = 0.1$, and thus $\frac{\partial K_x}{\partial x} = \frac{\partial K_y}{\partial y} = 0$. In Region C, we stipulate that $K_x = 1.0$, $K_y = 0.8$, and thus $\frac{\partial K_x}{\partial x} = \frac{\partial K_y}{\partial y} = 0$.

On the boundary between Regions A and C, we set $K_x = 0.65$ and $K_y = 0.5$. On the horizontal portion of this boundary, i.e., $0 \leq x \leq 0.3$ and $y = 0.7$, we use $\frac{\partial K_x}{\partial x} = 0$ and $\frac{\partial K_y}{\partial y} = \frac{0.6}{\Delta y}$, where $\Delta y = \frac{1}{m_y}$. On the vertical portion, i.e., $0 \leq y \leq 0.7$ and $x = 0.3$, we use $\frac{\partial K_x}{\partial x} = \frac{0.7}{\Delta x}$ and $\frac{\partial K_y}{\partial y} = 0$, where $\Delta x = \frac{1}{m_x}$.

On the boundary between Regions B and C, we set $K_x = 0.6$ and $K_y = 0.45$. On the horizontal portion of this boundary, i.e., $0.7 \leq x \leq 1.0$ and $y = 0.3$, we use $\frac{\partial K_x}{\partial x} = 0$ and

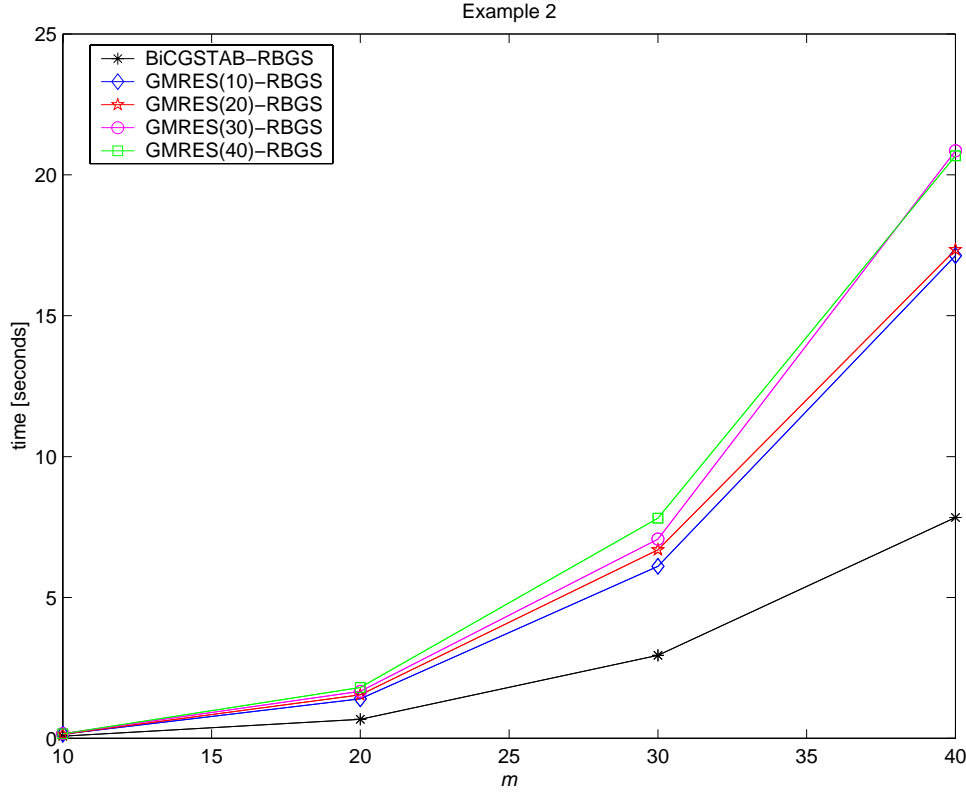


FIG. 8. Results for RBGS preconditioner

$\frac{\partial K_y}{\partial y} = \frac{-0.7}{\Delta y}$. On the vertical portion, i.e., $0.3 \leq y \leq 1.0$ and $x = 0.7$, we use $\frac{\partial K_x}{\partial x} = \frac{-0.8}{\Delta x}$ and $\frac{\partial K_y}{\partial y} = 0$.

A contour plot of the solution of (5.2) is in Figure 11.

The results appear in Figures 12 and 13, which are qualitatively very similar to the figures in the previous example. Again, we see that for both preconditioners, the Bi-CGSTAB method performs much better than do all versions of GMRES. Comparing the vertical scales of the figures reveals again that RBGS performs much better than ILU.

VI. SUMMARY AND CONCLUSIONS

We studied herein the use of the Bi-CGSTAB and GMRES methods with RBGS and ILU preconditioners for solving PDEs in two spatial dimensions discretized by Hermite collocation. The results clearly indicate that Bi-CGSTAB is superior to GMRES and that RBGS is superior to ILU. These results hold for a wide range of problem sizes, different types of boundary conditions, and heterogeneity associated with the coefficients of the PDEs.

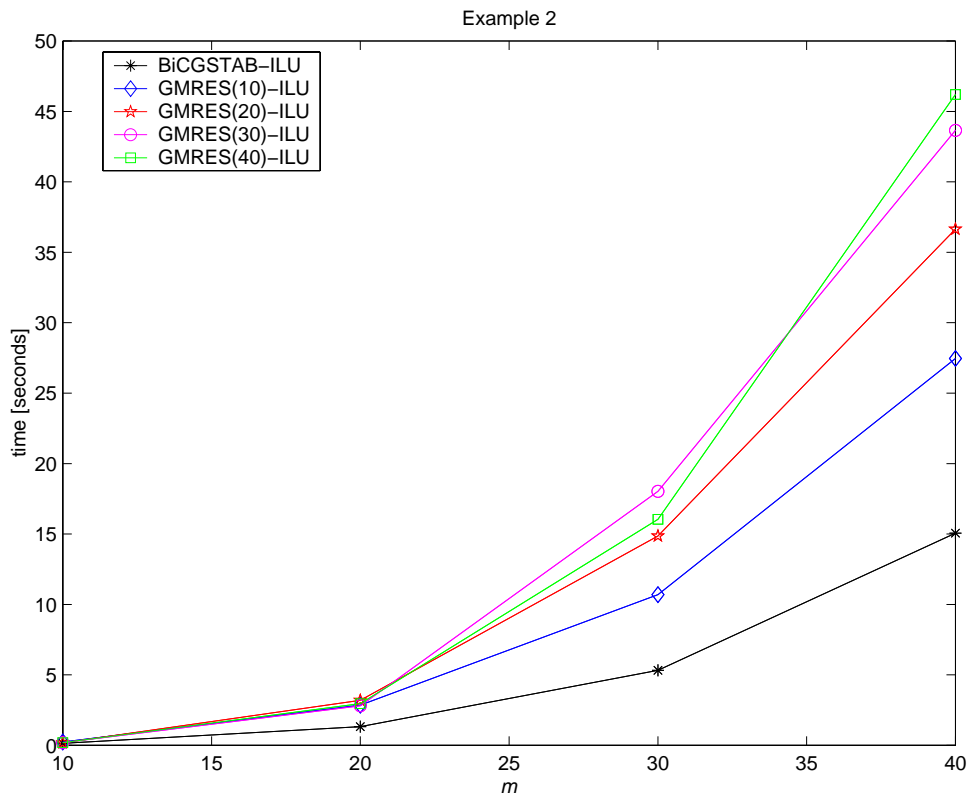


FIG. 9. Results for ILU preconditioner

The result that RBGS is superior to ILU is particularly satisfying because the RBGS preconditioner requires no effort to construct while considerable work is done to form the ILU preconditioner. Also, no additional storage is required for the RBGS preconditioner while the ILU preconditioner requires either extra storage in memory or the computational expense of repeated constructions. In addition, although the issue is not addressed in this paper, Bi-CGSTAB/RBGS for Hermite collocation is readily implemented in a parallel processing mode.

REFERENCES

1. van der Vorst, H. A. (1992). Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644.
2. Saad, Y. and Schultz, M. H. (1986). GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869.
3. Bialecki, B., Fairweather, G. and Bennett, K. R. (1992). Fast Direct Solvers for Piecewise Hermite Bicubic Orthogonal Spline Collocation Equations. *SIAM J. Numer. Anal.*, 29:156–

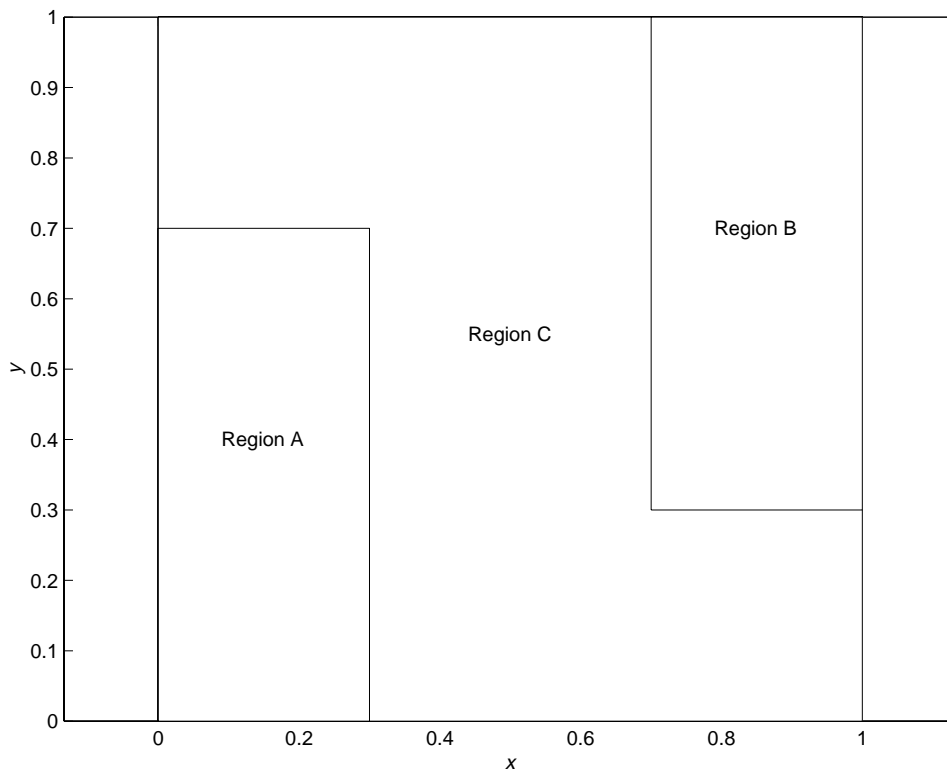


FIG. 10. Regions of the domain for Example 3

173.

4. Lai, Y.-L., Hadjidimos, A., Houstis, E. N., and Rice, J. R. (1995). On the Iterative Solution of Hermite Collocation Equations. *SIAM J. Matrix Anal. Appl.*, 16:254–277.
5. Brill, S. H. (1998). Documentation for Collocation Software. RCGRD Publication #98-01, Research Center for Groundwater Remediation Design, University of Vermont, Burlington, Vermont.
6. Frind, E. O. and Pinder, G.F. (1979). A Collocation Finite Element Method for Potential Problems in Irregular Domains. *International Journal for Numerical Methods in Engineering*, 14:681–701.
7. Lapidus, L. and Pinder, G. F. (1982). *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley & Sons, Inc., New York.
8. Brill, S. H. (1998). *The Solution of Two-Dimensional Partial Differential Equations via Hermite Collocation with Block Red-Black Gauss-Seidel Preconditioner*, Ph.D. thesis, University of Vermont, Burlington, Vermont.
9. Prenter, P. M. and Russell, R. D. (1976). Orthogonal Collocation for Elliptic Partial Differential Equations. *SIAM J. Numer. Anal.*, 13:923–939.
10. Papatheodorou, T. S. (1983). Block AOR Iteration for Nonsymmetric Matrices. *Mathematics of Computation*, 41:511–525.

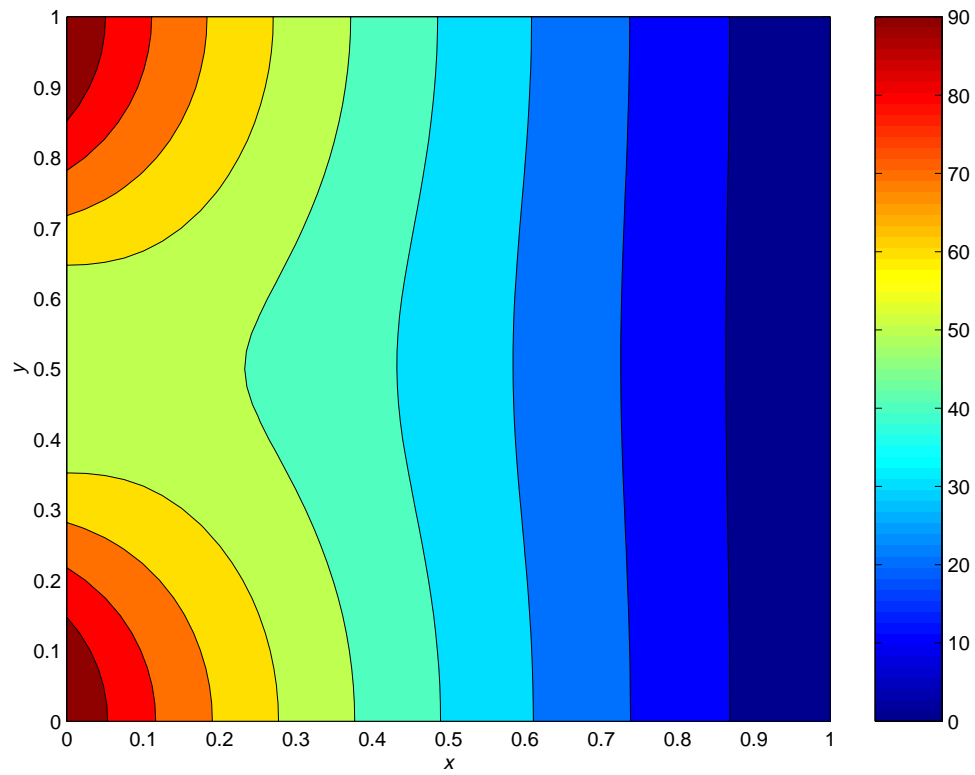


FIG. 11. Solution for Example 3

11. Burden, R. L. Burden and Faires, J. D. (1997). *Numerical Analysis*, Sixth Edition, Brooks/Cole Publishing Co., Pacific Grove.

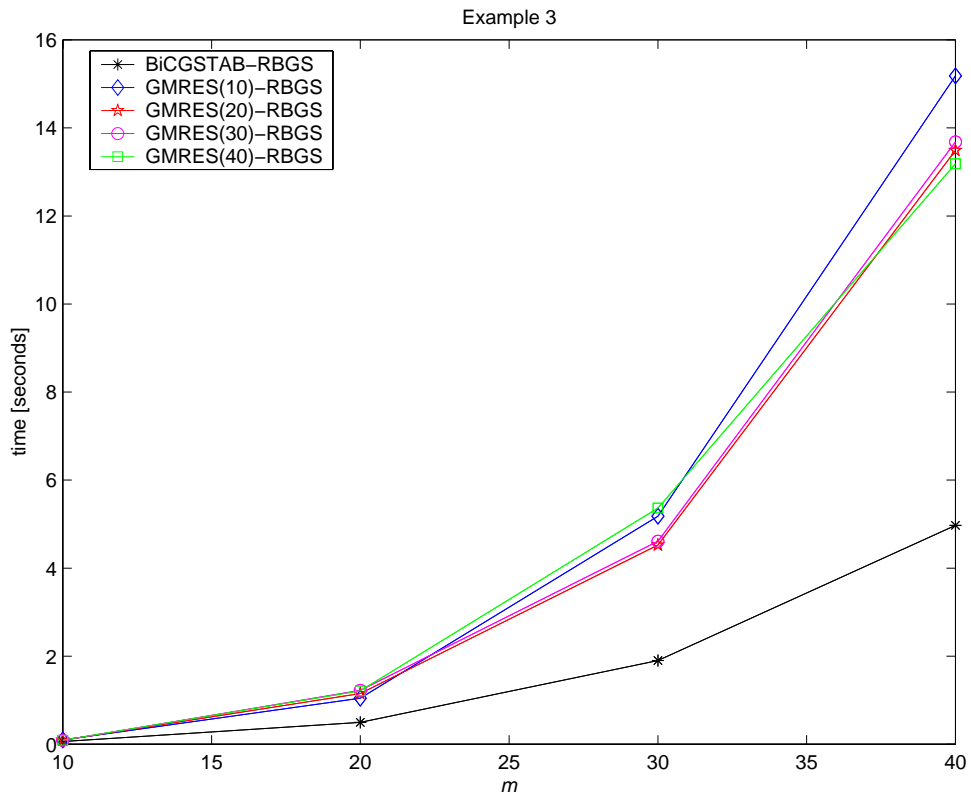


FIG. 12. Results for RBGS preconditioner

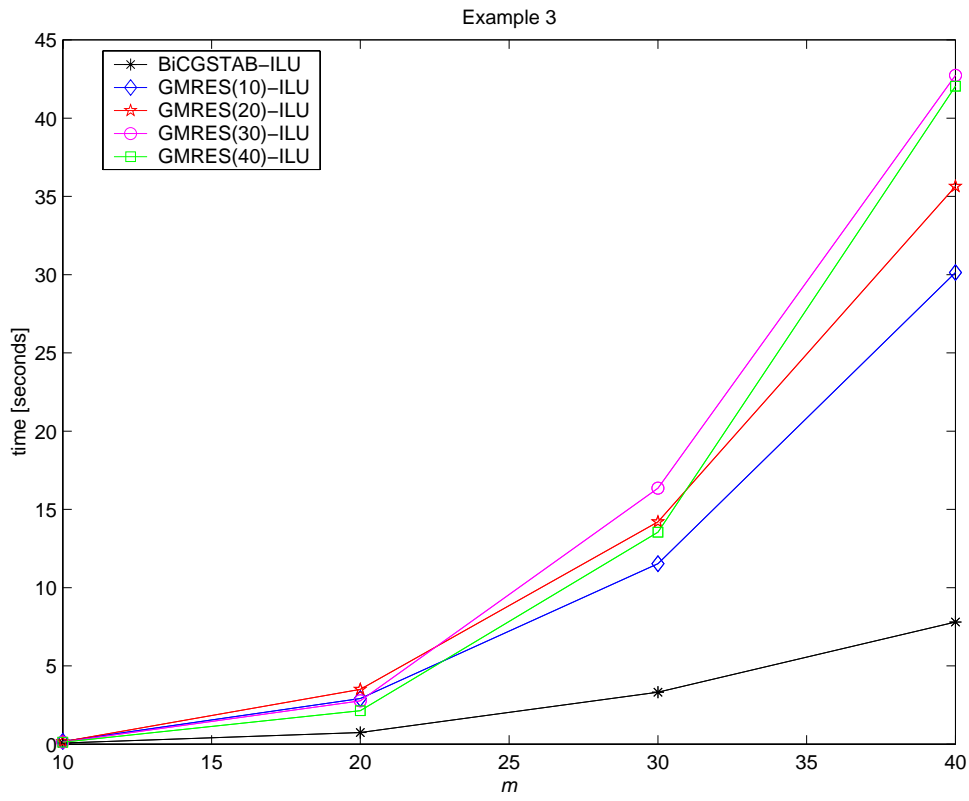


FIG. 13. Results for ILU preconditioner