

# A Parallel Preconditioned Bi-Conjugate Gradient Algorithm for Two-Dimensional Elliptic and Parabolic Equations Using Hermite Collocation

Stephen H. Brill <sup>\*</sup>      George F. Pinder <sup>†</sup>

## Abstract

A fast and parallelizable method to solve sparse matrix equations that arise from the Hermite collocation discretization of elliptic and parabolic partial differential equations (PDEs) can be obtained using a bi-conjugate gradient algorithm. A Red-Black Gauss-Seidel matrix preconditioner provides a structure which makes the algorithm amenable to parallel processing. Increased efficiency is achieved through optimally locating the “collocation points”. Results concerning eigenvalues and the efficacy of the method when applied to model problems can be demonstrated.

## 1 Hermite Cubic Polynomials and Collocation Discretization of PDEs

Let  $u(x, y)$  be a function defined on the unit square  $\mathcal{S} = [0, 1] \times [0, 1]$ . Partition  $\partial\mathcal{S}$  by using  $m + 1$  equally spaced nodes  $\{x_q\}_{q=0}^m$  and  $\{y_r\}_{r=0}^m$  in the  $x$ - and  $y$ -directions, respectively.  $\mathcal{S}$  is then partitioned into  $m^2$  square elements  $[x_{q-1}, x_q] \times [y_{r-1}, y_r]$ , each of dimension  $h \times h$ , where  $h = \frac{1}{m}$  and where  $q, r = 1, 2, \dots, m$ .

Consider the Hermite polynomials, defined for  $\eta$  in the interval  $[-\frac{1}{2}, \frac{1}{2}]$ :

$$(1) \quad \begin{aligned} f_j(x) &= \begin{cases} \frac{1}{2}(1+2\eta)^2(1-\eta), & x_{j-1} \leq x = x_{j-\frac{1}{2}} + \eta h \leq x_j \\ \frac{1}{2}(1-2\eta)^2(1+\eta), & x_j \leq x = x_{j+\frac{1}{2}} + \eta h \leq x_{j+1} \\ 0, & \text{otherwise} \end{cases} \\ g_j(x) &= \begin{cases} \frac{h}{8}(1+2\eta)^2(2\eta-1), & x_{j-1} \leq x = x_{j-\frac{1}{2}} + \eta h \leq x_j \\ \frac{h}{8}(1-2\eta)^2(2\eta+1), & x_j \leq x = x_{j+\frac{1}{2}} + \eta h \leq x_{j+1} \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Then the bi-cubic polynomial interpolating  $u_{qr} = u(x_q, y_r)$ ,  $u_{qr}^x = \frac{\partial u}{\partial x}(x_q, y_r)$ ,  $u_{qr}^y = \frac{\partial u}{\partial y}(x_q, y_r)$ ,  $u_{qr}^{xy} = \frac{\partial^2 u}{\partial x \partial y}(x_q, y_r)$ ;  $q, r = 0, 1, 2, \dots, m$ ; is

$$(2) \quad \hat{u}(x, y) = \sum_{q=0}^m \sum_{r=0}^m [u_{qr} f_q(x) f_r(y) + u_{qr}^x g_q(x) f_r(y) + u_{qr}^y f_q(x) g_r(y) + u_{qr}^{xy} g_q(x) g_r(y)].$$

If we introduce the interpolating polynomial (2) into Poisson’s equation

$$(3) \quad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = H(x, y),$$

<sup>\*</sup>Department of Mathematics and Statistics, University of Vermont, Burlington, VT

<sup>†</sup>Departments of Civil and Environmental Engineering and of Mathematics and Statistics, University of Vermont, Burlington, VT

we obtain

$$(4) \quad \frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial y^2} - H(x, y) = E(x, y),$$

where  $E(x, y)$  is an error function. It is seen (in [1]) that (4) has  $4m^2$  unknowns (after appropriate boundary conditions are introduced) and we thus require  $4m^2$  equations, or 4 equations per element, to uniquely determine these unknowns. To achieve this, we choose four points  $(x_k, y_\ell)$  in the interior of each element and enforce  $E(x_k, y_\ell) = 0$  at each of these  $4m^2$  ‘‘collocation points’’. In the literature (e.g. [1], [5], [7]), assuming a square element  $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ , the usual choices for the collocation points are the points of Gaussian quadrature, namely  $(-\xi, -\xi)$ ,  $(-\xi, \xi)$ ,  $(\xi, -\xi)$ , and  $(\xi, \xi)$ , where  $\xi = \frac{1}{2\sqrt{3}}$ . In this paper, we will allow  $\xi$  to assume any value in the interval  $(0, \frac{1}{2})$ . Transforming the four collocation points into each of the  $m^2$  elements defines the full set of  $4m^2$  collocation equations. These can be written

$$(5) \quad \sum_{q=0}^m \sum_{r=0}^m \{ [f_q''(x_k) f_r(y_\ell) + f_q(x_k) f_r''(y_\ell)] u_{qr}^x + [g_q''(x_k) f_r(y_\ell) + g_q(x_k) f_r''(y_\ell)] u_{qr}^{xy} \\ + [f_q''(x_k) g_r(y_\ell) + f_q(x_k) g_r''(y_\ell)] u_{qr}^y + [g_q''(x_k) g_r(y_\ell) + g_q(x_k) g_r''(y_\ell)] u_{qr}^{xy} \} = H(x_k, y_\ell),$$

where  $(x_k, y_\ell)$  varies over all  $4m^2$  collocation points.

## 2 Red-Black Numbering of Equations and Unknowns

In this paper, we number the equations and unknowns in such a way as to produce a method that is amenable to parallel processing. In particular, we employ the numbering discussed in [1], which, for the case where  $m = 4$ , is depicted in Figure 1(a). Here the unknowns appear in small boxes at the node at which they are defined while the collocation points (equations) appear in the interior of the element over which they are defined. This numbering (see [1] for details) leads us to consider the matrix equation

$$(6) \quad M \mathbf{v} = \mathbf{k},$$

where (for the case  $m = 4$ )  $M$  has the structure given in Figure 1(b). Corresponding to the blocks in Figure 1(b), we write

$$M = \begin{bmatrix} R & -U \\ -L & B \end{bmatrix}.$$

In [1], we considered solving (6) using a block red-black SOR (successive over-relaxation) scheme, the efficiency of which is dependent on the relaxation parameter  $\omega$ . If we take  $\omega = 1$ , we obtain the block red-black Gauss-Seidel method

$$(7) \quad \left[ \begin{array}{c|c} R & \\ \hline -L & B \end{array} \right] \begin{bmatrix} \mathbf{v}_R^{(n+1)} \\ \mathbf{v}_B^{(n+1)} \end{bmatrix} = \left[ \begin{array}{c|c} & U \\ \hline & \end{array} \right] \begin{bmatrix} \mathbf{v}_R^{(n)} \\ \mathbf{v}_B^{(n)} \end{bmatrix} + \begin{bmatrix} \mathbf{k}_R \\ \mathbf{k}_B \end{bmatrix},$$

where the superscript indicates iteration number.

At each iteration, we first solve for  $\mathbf{v}_R^{(n+1)}$ . Since the non-zero entries of  $R$  lie only in its diagonal blocks, these blocks of equations are uncoupled from each other; hence we may simultaneously solve these blocks of equations in parallel. Once we have obtained  $\mathbf{v}_R^{(n+1)}$ , we then solve for  $\mathbf{v}_B^{(n+1)}$ . Considering the structure of  $B$ , we may likewise solve for  $\mathbf{v}_B^{(n+1)}$  in parallel.

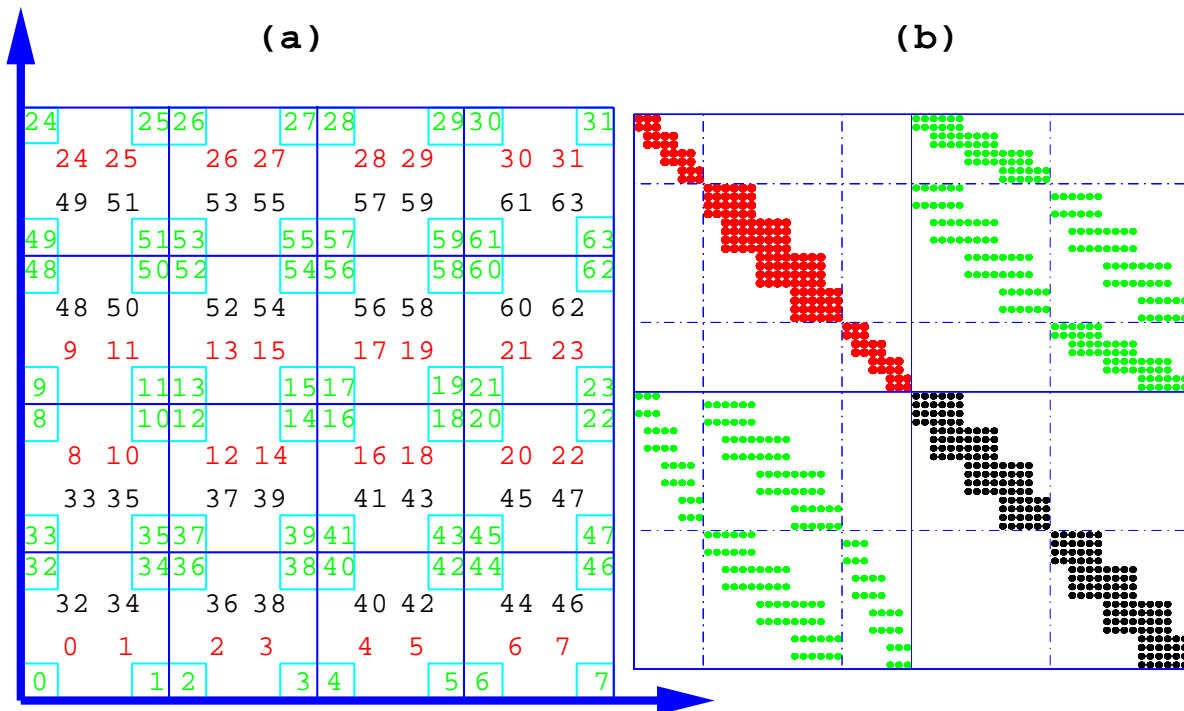


FIG. 1. red-black numbering and matrix structure

### 3 Our Preconditioning Matrix

In this paper, we employ the bi-conjugate gradient method discussed in [2] and [8] to solve (6).

It is well known (see [4] and [8]) that conjugate gradient methods work best when the matrix  $M$  is “close” to the identity matrix. We thus seek a preconditioning matrix  $P$  such that  $P^{-1}M$  is “close” to the identity matrix, that linear systems of the form  $P\mathbf{v} = \mathbf{k}$  are easily solved, and that  $P$  has a structure that is amenable to parallel processing. In this paper, we choose  $P$  to be the Gauss-Seidel matrix found in (7), namely

$$P = \left[ \begin{array}{c|c} R & \\ \hline -L & B \end{array} \right].$$

It is clear that  $P$  satisfies the last two conditions above. We now study the “closeness” of  $P^{-1}M$  to the identity matrix.

A direct computation shows that

$$P^{-1}M = \left[ \begin{array}{c|c} I & -R^{-1}U \\ \hline I - B^{-1}LR^{-1}U & \end{array} \right],$$

where  $I$  represents the identity matrix of appropriate size. It is clear that  $P^{-1}M$  being “close” to the identity matrix is equivalent to

$$I - P^{-1}M = \left[ \begin{array}{c|c} & R^{-1}U \\ \hline & B^{-1}LR^{-1}U \end{array} \right]$$

being “close” to the “null” matrix (i.e., the matrix whose entries are all equal to zero). The null matrix has all its eigenvalues equal to zero. Since  $I - P^{-1}M$  is a block upper

triangular matrix, its set of eigenvalues is given by the union of the sets of eigenvalues of those matrices on its diagonal blocks, namely the null matrix and  $J' = B^{-1}LR^{-1}U$ . We therefore expect the preconditioned bi-conjugate gradient (PBCG) method to converge most quickly when the eigenvalues of  $J'$  are clustered near the origin of the complex plane.

## 4 Eigenvalues and Results

### 4.1 Poisson's Equation

To simplify our eigenvalue analysis, we make two changes to the formulation given above. First, since the eigenvalues of  $J = LR^{-1}UB^{-1}$  and those of  $J'$  are identical, we perform our analysis on  $J$ , which is much easier than using  $J'$ . Second, we replace  $g_j(x)$  in (1) by  $g_j^*(x) = \frac{g_j(x)}{h}$  as done in [1], [5], and [7]. It is clear that this replacement is equivalent to changing  $M$  to  $M^* = MD$  where  $D$  is a diagonal matrix whose diagonal entries are non-positive integer powers of  $h$ . In this case, we choose the preconditioning matrix  $P^* = PD$ . Then  $(P^*)^{-1}M^* = D^{-1}P^{-1}MD$ , which clearly has the same eigenvalues as  $P^{-1}M$ .

The  $2m^2$  eigenvalues of  $J$  are determined using an analysis similar to that found in [5], which is based upon the theory given in [3]. We find that the eigenvalues of  $J$  are given by the following recipe. We note that these eigenvalues are all functions of  $\xi$ , the parameter which determines the location of the collocation points in each element. We also require that  $m$  is even and assume we have Dirichlet boundary conditions.

$$\theta_i = \frac{i\pi}{m} \quad c_i = \cos \theta_i$$

$$q_i = 2\xi\sqrt{(16\xi^4 - 24\xi^2 + 21) + c_i(-32\xi^4 + 18) + c_i^2(16\xi^4 + 24\xi^2 - 3)}$$

$$\alpha_i^\pm = \frac{96\xi^4 - 92\xi^2 + 5 + c_i(1 - 12\xi^2)(-1 + 8\xi^2) \pm q_i}{-96\xi^4 - 56\xi^3 + 76\xi^2 + 42\xi + 5 + c_i(-1 - 2\xi)(1 + 4\xi - 4\xi^2 - 48\xi^3)}$$

$$\beta_i^\pm = \frac{128\xi^6 - 192\xi^4 + 60\xi^2 - 1 + c_i(-128\xi^6 + 96\xi^4 - 12\xi^2 + 1) \pm q_i}{(1 + 2\xi)(-1 - 16\xi - 28\xi^2 + 80\xi^3 + 32\xi^4 - 64\xi^5) + c_i(1 - 2\xi)(1 + 2\xi)^2(1 + 4\xi + 8\xi^2 - 16\xi^3)}$$

where  $i = 0, 1, 2, \dots, m$ . Then form the sets

$$\{d_1, d_2, d_3, \dots, d_{2m}\} = \{\alpha_0^+, \alpha_1^+, \alpha_1^-, \alpha_2^+, \alpha_2^-, \dots, \alpha_{m-1}^+, \alpha_{m-1}^-, \alpha_m^+\},$$

$$\{\bar{d}_1, \bar{d}_2, \bar{d}_3, \dots, \bar{d}_{2m}\} = \{\beta_0^-, \beta_1^-, \beta_1^+, \beta_2^-, \beta_2^+, \dots, \beta_{m-1}^-, \beta_{m-1}^+, \beta_m^-\}.$$

Now, let  $z_{ik} = (d_i - \bar{d}_i)\cos\theta_k$  for  $i = 1, 2, 3, \dots, 2m$  and  $k = 1, 2, 3, \dots, \frac{m}{2} - 1$ . Then  $\sigma(J)$ , the set of eigenvalues of  $J$ , is

$$\sigma(J) = \left\{ \mu : \mu = d_i^2; i = 1, \dots, 2m \right\} \cup \left\{ \mu : \mu = d_i \bar{d}_i; i = 1, \dots, 2m \right\}$$

$$\cup \left\{ \mu : \mu = \frac{1}{2} \left( z_{ik}^2 + 2d_i \bar{d}_i \pm \sqrt{z_{ik}^2(z_{ik}^2 + 4d_i \bar{d}_i)} \right); i = 1, \dots, 2m; k = 1, \dots, \frac{m}{2} - 1 \right\}.$$

Now that we have analytic formulas for the eigenvalues of  $J$ , we seek to manipulate  $\xi$  to maximize the rate of convergence of the PBCG method. Let  $\mathbf{w}$  be the vector whose  $(2m^2)$  entries are the eigenvalues of  $J$ . Since we want the eigenvalues to be clustered near the origin of the complex plane, we want to find  $\xi$  to minimize  $\|\mathbf{w}\|_2$ .

In Figure 2, we see justification that minimizing  $\|\mathbf{w}\|_2$  gives the fastest convergence of the PBCG method. Figure 2(a) shows how  $\|\mathbf{w}\|_2$  varies as a function of  $\xi$  for

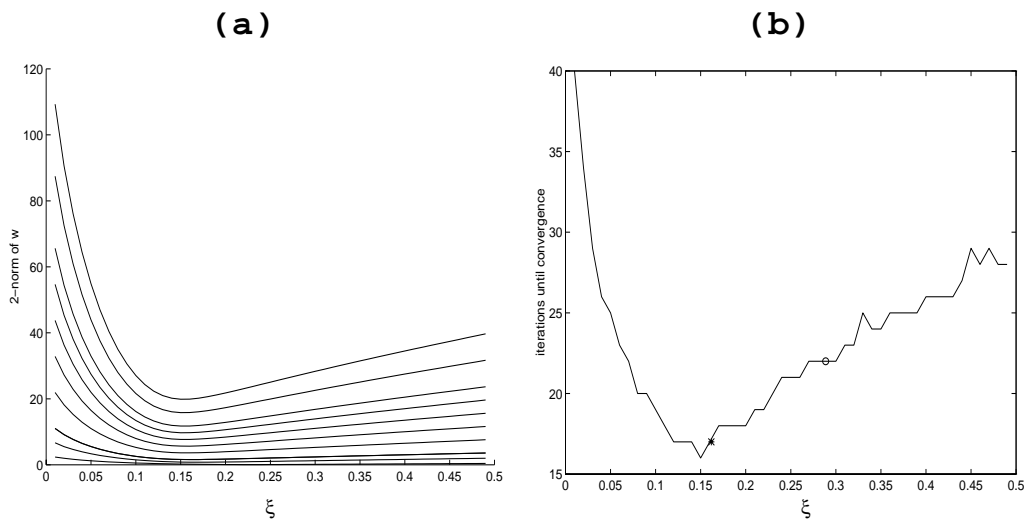


FIG. 2. numerical results for Poisson's equation

$m=2,6,10,20,30,40,50,60,80,100$  (the curves appear from bottom to top in this order). We see that  $\|\mathbf{w}\|_2$  is large for small values of  $\xi$ . As  $\xi$  increases,  $\|\mathbf{w}\|_2$  decreases rapidly to a minimum, then increases slowly. This behavior is mirrored in Figure 2(b), where we solved Poisson's equation (3) for various values of  $\xi$  and recorded the number of iterations required for the PBCG method to converge. (We used  $m = 10$ ; chose the boundary conditions and forcing function such that the exact solution was  $u(x, y) = x^2y$ ; and chose  $\epsilon = 10^{-8}$ , where the stopping criterion was that  $\|\mathbf{r}\|_\infty < \epsilon$ , where  $\mathbf{r}$  is the usual residual vector calculated during the PBCG algorithm.)

To determine the value of  $\xi$  that minimizes  $\|\mathbf{w}\|_2$ , we employed the optimization software MINOS (see [6] for documentation). For the case where  $m = 10$ , the optimal value of  $\xi$  was found to be 0.16182, which agrees well with our numerical results. In Figure 2(b), the circle indicates using the Gaussian quadrature value  $\xi = \frac{1}{2\sqrt{3}}$ , while the asterisk indicates use of the optimal value  $\xi = 0.16182$ . We thus see a substantial increase in efficiency when we use the optimal  $\xi$  instead of the Gaussian value of  $\xi$ .

We also had MINOS determine the optimal value of  $\xi$  for all even  $m$  between 2 and 100, inclusive. The results are shown in Figure 3. We see that the optimal value of  $\xi$  is rather insensitive with respect to  $m$  for sufficiently large  $m$ . Therefore, performing an optimization is not critical each time we wish to solve Poisson's equation (3) for a new value of  $m$ .

## 4.2 A Model Parabolic Equation

We now consider solving the parabolic PDE

$$(8) \quad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - H(x, y, t),$$

with Dirichlet boundary conditions and appropriate initial conditions. We approximate the time derivative by

$$(9) \quad \frac{\partial u}{\partial t} = \frac{u^{(n+1)} - u^{(n)}}{\Delta t},$$

where the superscript  $(n)$  indicates the value of  $u$  after  $n$  time steps.

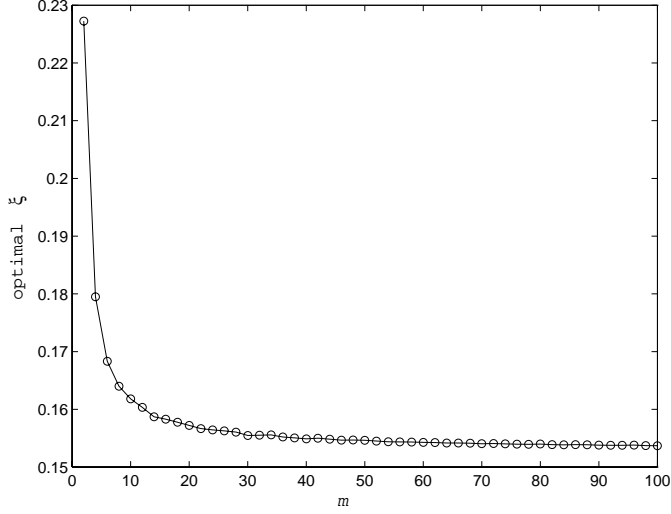


FIG. 3. *optimal  $\xi$  as a function of  $m$  for Poisson's equation*

Recalling (2), (5), and (6), we see that matrix  $M$  was formed by evaluating  $\frac{\partial^2 \hat{u}}{\partial x^2} + \frac{\partial^2 \hat{u}}{\partial y^2}$  at the collocation points. Correspondingly, we form matrix  $Q$  by evaluating  $\hat{u}$  at the collocation points. Clearly,  $Q$  has precisely the same structure as that of  $M$ .

If we now introduce (9) and the interpolating polynomial (2)<sup>1</sup> into (8) and evaluate the right side of (8) at the collocation points at time  $\theta t^{(n+1)} + (1-\theta)t^{(n)}$ , where  $0 \leq \theta \leq 1$ , then we obtain the matrix form of the collocation discretization of the parabolic PDE (8):

$$(10) \quad \frac{Q\mathbf{v}^{(n+1)} - Q\mathbf{v}^{(n)}}{\Delta t} = \theta [M\mathbf{v}^{(n+1)} - \mathbf{k}^{(n+1)}] + (1-\theta) [M\mathbf{v}^{(n)} - \mathbf{k}^{(n)}].$$

Letting  $\tau = \theta \Delta t$  and  $\bar{\tau} = (1-\theta) \Delta t$ , we may express (10) as

$$(11) \quad (Q - \tau M)\mathbf{v}^{(n+1)} = (Q + \bar{\tau} M)\mathbf{v}^{(n)} - (\tau \mathbf{k}^{(n+1)} + \bar{\tau} \mathbf{k}^{(n)}).$$

In examining (11), we see that this equation defines how we may move from time step ( $n$ ) to time step ( $n+1$ ), because, at time step ( $n$ ), all the vectors on the right side of (11) contain known values. We may therefore apply to (11) the block red-black PBCG algorithm that we developed for (6). That is, at each time step in (11) we iterate to convergence using block red-black PBCG.

At this point, we may apply the analysis used to determine the eigenvalues arising from using our PBCG method on Poisson's equation (3) with Dirichlet boundary conditions to the parabolic equation (8) with Dirichlet boundary conditions. The recipe is identical to that given above, except for the definitions of  $\alpha_i^\pm$  and  $\beta_i^\pm$ . We first define  $\nu = \frac{\tau}{h^2}$ .

Let  $a_{\alpha_i} = [(5 - 64\xi^2 + 288\xi^4 - 512\xi^6 + 256\xi^8) + c_i(-1 + 16\xi^2 - 96\xi^4 + 256\xi^6 - 256\xi^8)] + [(176 - 2240\xi^2 + 7424\xi^4 - 5120\xi^6) + c_i(-16 + 448\xi^2 - 2816\xi^4 + 5120\xi^6)]\nu + [(1280 - 23552\xi^2 + 24576\xi^4) + c_i(-256 + 5120\xi^2 - 24576\xi^4)]\nu^2$ .

Let  $b_{\alpha_i} = [(5 + 20\xi - 24\xi^2 - 176\xi^3 - 64\xi^4 + 448\xi^5 + 384\xi^6 - 256\xi^7 - 256\xi^8) + c_i(-1 - 4\xi + 8\xi^2 + 48\xi^3 - 192\xi^5 - 128\xi^6 + 256\xi^7 + 256\xi^8)] + [(176 + 1024\xi + 448\xi^2 - 5120\xi^3 - 5888\xi^4 + 4096\xi^5 + 5120\xi^6)$

<sup>1</sup>The interpolating polynomial (2) and forcing function now have time dependence, i.e.  $u_{qr}$ ,  $u_{qr}^x$ ,  $u_{qr}^y$ ,  $u_{qr}^{xy}$ , and  $H$  are now functions also of  $t$ .

$$\begin{aligned}
& + c_i(-16 - 128\xi + 64\xi^2 + 1536\xi^3 + 1280\xi^4 - 4096\xi^5 - 5120\xi^6)]\nu \\
& + [(1280 + 10752\xi + 19456\xi^2 - 14336\xi^3 - 24576\xi^4) \\
& + c_i(-256 - 1536\xi - 1024\xi^2 + 14336\xi^3 + 24576\xi^4)]\nu^2. \\
& \text{Let } a_{\beta_i} = [(-5 + 69\xi^2 - 352\xi^4 + 800\xi^6 - 768\xi^8 + 256\xi^{10}) \\
& + c_i(1 - 17\xi^2 + 112\xi^4 - 352\xi^6 + 512\xi^8 - 256\xi^{10})] \\
& + [(-96 + 1712\xi^2 - 7872\xi^4 + 11520\xi^6 - 5120\xi^8) + c_i(-272\xi^2 + 2496\xi^4 - 6912\xi^6 + 5120\xi^8)]\nu \\
& + [(-192 + 11520\xi^2 - 36864\xi^4 + 24576\xi^6) + c_i(192 - 2304\xi^2 + 18432\xi^4 - 24576\xi^6)]\nu^2. \\
& \text{Let } b_{\beta_i} = [(-5 - 30\xi - 21\xi^2 + 184\xi^3 + 320\xi^4 - 288\xi^5 - 800\xi^6 + 128\xi^7 + 768\xi^8 - 256\xi^{10}) \\
& + c_i(1 + 6\xi + 1\xi^2 - 56\xi^3 - 80\xi^4 + 160\xi^5 + 352\xi^6 - 128\xi^7 - 512\xi^8 + 256\xi^{10})] \\
& + [(-96 - 816\xi - 1456\xi^2 + 2688\xi^3 + 7872\xi^4 - 1792\xi^5 - 11520\xi^6 + 5120\xi^8) \\
& + c_i(48\xi + 16\xi^2 - 1152\xi^3 - 2496\xi^4 + 1792\xi^5 + 6912\xi^6 - 5120\xi^8)]\nu \\
& + [(-192 - 3456\xi - 11520\xi^2 + 4608\xi^3 + 36864\xi^4 - 24576\xi^6) \\
& + c_i(192 + 1152\xi + 2304\xi^2 - 4608\xi^3 - 18432\xi^4 + 24576\xi^6)]\nu^2. \\
& \text{Then } \alpha_i^\pm = \frac{a_{\alpha_i} \pm 256q_i\nu^2}{b_{\alpha_i}} \text{ and } \beta_i^\pm = \frac{a_{\beta_i} \pm 192q_i\nu^2}{b_{\beta_i}}.
\end{aligned}$$

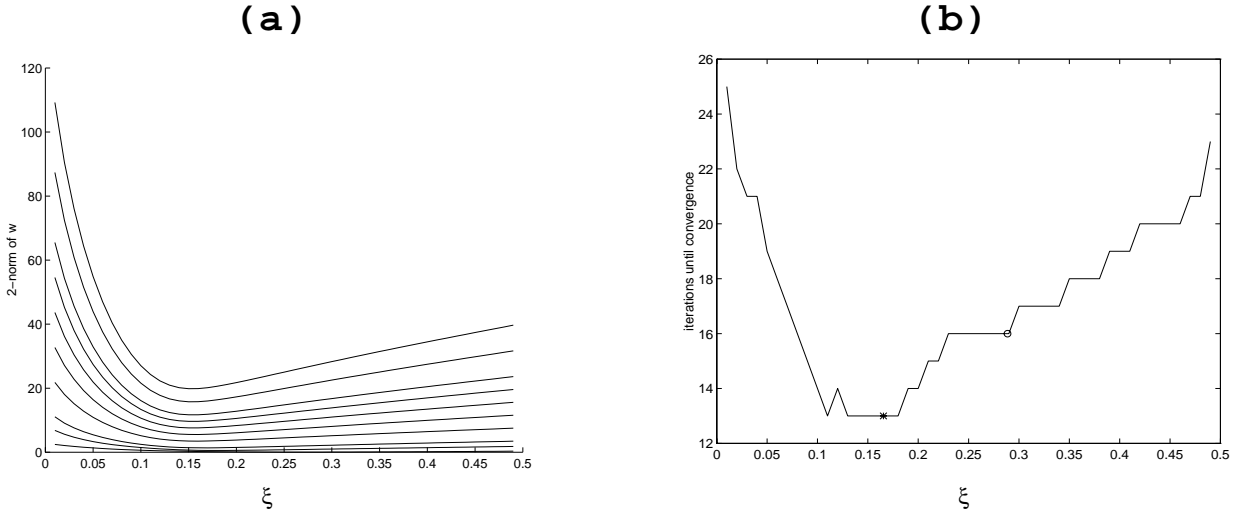


FIG. 4. numerical results for the model parabolic equation

In Figure 4, we see results for the model parabolic equation (8) that are completely analogous to the results for Poisson's equation (3) found in Figure 2. In Figure 4(a), we used  $\tau = 0.05$  and the same values of  $m$  as in Figure 2(a). In Figure 4(b), we again chose  $m = 10$  and chose the forcing function, boundary conditions, and initial conditions (corresponding to  $t = 0$ ) so that the exact solution was  $u(x, y, t) = x^2y(1 + e^{-t})$ . We set  $\theta = \frac{1}{2}$  and ran the code over one time step from  $t = 0$  to  $t = 0.1$ . The stopping criterion was again  $\|\mathbf{r}\|_\infty < \epsilon = 10^{-8}$ . As above, the circle corresponds to using the Gaussian value for  $\xi$ , while the asterisk corresponds to using the optimal value for  $\xi$ , found by MINOS to be 0.16545.

We also had MINOS determine the optimal value of  $\xi$  for the model parabolic equation for several values of  $\tau$  for all even  $m$  between 2 and 50, inclusive (see Figure 5). We see here that the optimal value of  $\xi$  is rather insensitive with respect to both  $\tau$  and  $m$  for sufficiently large  $m$ . It is therefore not critical to optimize each time we wish to solve the model parabolic equation (8) with new values of  $\tau$  and  $m$ .

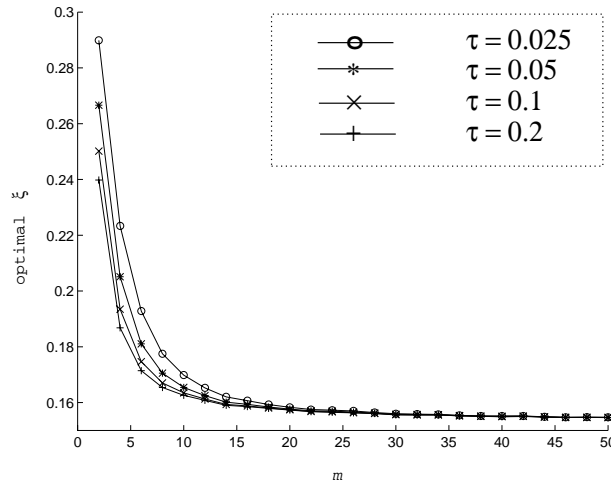


FIG. 5. optimal  $\xi$  as a function of  $m$  for several values of  $\tau$  for the model parabolic equation

## 5 Conclusion

We studied herein a preconditioned bi-conjugate gradient (PBCG) algorithm for solving sparse matrix equations arising from the Hermite collocation discretization of Poisson's equation and of a model parabolic PDE. We analytically determined formulas for the eigenvalues of the matrix  $J$  associated with our choice of preconditioner (the block red-black Gauss-Seidel matrix, which makes our method amenable to parallel processing). The MINOS optimization software enabled us to determine the optimal location for the collocation points, allowing the PBCG algorithm to converge in a minimal number of iterations.

## References

- [1] S. H. Brill and G. F. Pinder, *A Block Red-Black SOR Method for a Two-Dimensional Parabolic Equation Using Hermite Collocation*, Proceedings of the MAFELAP Conference, Brunel University, U.K., 1995, in press.
- [2] R. Fletcher, *Conjugate Gradient Methods for Indefinite Systems*, in Dundee Conference on Numerical Analysis, Springer-Verlag, Berlin, 1975.
- [3] I. Gohberg, P. Lancaster, L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.
- [4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1989.
- [5] Y.-L. Lai, A. Hadjidimos, E. N. Houstis, and J. R. Rice, *On the Iterative Solution of Hermite Collocation Equations*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 254-277. (Also Technical Report CSD-TR-92-094, Purdue University, 1992.)
- [6] B. A. Murtaugh and M. A. Saunders, *MINOS 5.1 User's Guide*, Technical Report SOL 83-20R, Stanford University, 1987.
- [7] T. S. Papatheodorou, *Block AOR Iteration for Nonsymmetric Matrices*, Math. Comp., 41 (1983), pp. 511-525.
- [8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.