Editor: Dianne P. O'Leary, oleary@cs.umd.edu

Solving Sparse Linear Systems: Taking the Direct Approach

By Dianne P. O'Leary

OUR STARTUP COMPANY, POISSONISUS.COM, HAS HIRED YOU AS A CONSULTANT TO ADVISE US ON SOLVING LINEAR SYSTEMS OF EQUATIONS. OUR BUSINESS IS TO SOLVE ELLIPTIC partial differential equations in two and three dimensions, but we have limited venture capital funding, so we're starting with a very specific mission: to solve the Poisson equation

$$-u_{xx} - u_{yy} = f(x, y),$$

when $(x, y) \in \Omega \subset \mathcal{R}^2$, or

$$-u_{xx} - u_{yy} - u_{zz} = f(x, y, z),$$

when $(x, y, z) \in \Omega \subset \mathcal{R}^3$. The complete problem specifications also include information about the behavior of the solution u on the boundary of Ω.

Problem Solving

The standard method for solving such problems is to *discretize* using either *finite differences* or *finite elements*, and then solve the resulting system of linear equations. To get accurate estimates of the solution, the system is made very large (involving thousands or millions of unknowns), so it's important to be very efficient in our solution algorithm. Your job is to evaluate some alternatives.

For testing purposes, we've developed two problems that we believe are typical of what our customers will provide. In the first problem, the domain is a sector of a circle, and the differential equation is discretized using an adaptive finite element grid. In the second, the domain is a 3D box with discretization using finite differences. The Web site (www.computer.org/cise/homework/) has Matlab functions to generate these problems.

The $n \times n$ matrices $\boldsymbol{A}$ we consider have three important properties:

- They are *real symmetric*, so element $a_{jk} = a_{kj}$ for $j, k = 1, \ldots, n$. This forces all the eigenvalues to be real.
- They are *positive definite*, so all the eigenvalues are positive.
- They are *sparse*, meaning that most of the matrix entries are zero and that the number of nonzero elements grows as n rather than n^2 as the discretization is refined.

The first two properties ensure that if we perform Gauss elimination on the linear systems, we never need to *pivot for stability*—interchange rows of the matrix to put a larger-magnitude element on the main diagonal. Also, we can take advantage of the symmetry of $\boldsymbol{A}$ and use the *Cholesky decomposition* of the matrix, factoring $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix. This requires half the work of Gauss elimination, but it's only stable if $\boldsymbol{A}$ is positive definite.

Because we don't need to pivot for stability, we are completely free to *pivot to preserve sparsity*; later, we'll turn our attention to why this is necessary and how to do it effectively. In addition to *direct methods* such as the Cholesky decomposition, we should also consider *iterative methods* such as conjugate gradients, but we'll focus on direct methods here.

In this homework assignment, we explore the importance of ordering when solving large, sparse systems of linear equations. Our examples are drawn from the solution of partial differential equations; such problems are a prime source of such linear systems. Nothing that we do is specific to these problems, however, and you might prefer to work with a matrix from a standard test set (for example, 'wathen' from Matlab's `gallery` function or a matrix from the Matrix Market at http://math.nist.gov/MatrixMarket/) or a matrix of particular interest in your work.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

Storing and Factoring Sparse Matrices

Several possible storage schemes exist for sparse matrices. Matlab chooses a typical one: store the indices and values of the nonzero elements in column order, so

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 5 & 7 & 0 \\ 1 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

is stored as

$$\begin{array}{ll} (1,1) & 2 \\ (3,1) & 1 \\ (2,2) & 5 \\ (2,3) & 7 \\ (3,3) & 6 \\ (4,4) & 8\,. \end{array}$$

Storing a sparse matrix in this way takes $3nz$ storage locations, where nz is the number of nonzeros. For small matrices, the choice of dense or sparse storage schemes doesn't matter much, but for problems with thousands or millions of unknowns, it matters a lot! If our matrix has only three nonzeros per row, for example, then we require $9n$ locations to store it in sparse format (rather than the n^2 required for dense); these numbers are quite different for large n.

If our problem involves a sparse matrix $\boldsymbol{A}$, then we would want $\boldsymbol{L}$ to be sparse, too, but this isn't always the case. Consider, for example, a linear system involving the "arrowhead" matrix

$$\boldsymbol{A}\mathbf{x} \equiv \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ \times & 0 & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & 0 & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} \equiv \mathbf{b},$$

where × denotes a nonzero value (we don't care what it is) and 0 denotes a zero. The number of nonzeros is $3n - 2$.

Suppose we use Gauss elimination (or the LU factorization or the Cholesky factorization—they all have the same trouble). In the first step, we would add some multiple of the first row to every other row to put zeros in the off-diagonal elements of column 1. Disaster! The matrix is now fully dense with n^2 nonzeros!

This problem has a simple fix, though. Let's rewrite our problem by moving the first column and the first row to the end, thus producing a reordered system

$$\begin{bmatrix} \times & 0 & 0 & 0 & 0 & \times \\ 0 & \times & 0 & 0 & 0 & \times \\ 0 & 0 & \times & 0 & 0 & \times \\ 0 & 0 & 0 & \times & 0 & \times \\ 0 & 0 & 0 & 0 & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_1 \end{bmatrix}.$$

We can express this reordering by using a *permutation matrix*

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

A permutation matrix is just an identity matrix with its rows reordered; we can use the reordering sequence $r = [2, 3, 4, 5, 1]$ to represent this one.

In the next problem, we see the dramatic effect that reordering has on sparsity.

Problem 1.

a. Verify that the reordered system has the same solution as the original one and that when we use Gauss elimination (or the Cholesky factorization), no new nonzeros are produced in our reordered system. (In particular, the Cholesky factor has $2n - 1$ nonzeros.)

b. Show that our reordered system is $(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T)(\boldsymbol{P}\mathbf{x}) = \boldsymbol{P}\mathbf{b}$.

Reordering the variables and equations is a powerful tool for maintaining sparsity during factorization, and we'll investigate some strategies for determining good permutations later. Notice that to preserve symmetry, we'll always pair $\boldsymbol{P}$ with $\boldsymbol{P}^T$ in reordering $\boldsymbol{A}$, but for nonsymmetric problems, choosing a different column permutation in place of $\boldsymbol{P}^T$ can be more advantageous.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

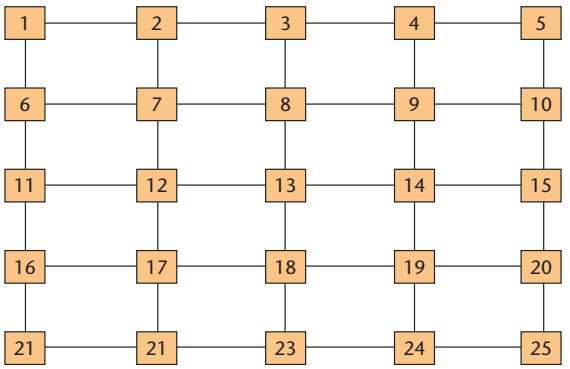

Figure 1. The graph corresponding to the matrix *S* in Figure 2.

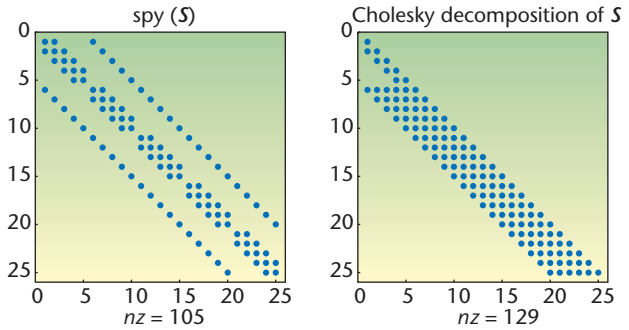

Figure 2. Sparsity pattern. The finite difference matrix *S* for Poisson's equation on the unit square is discretized with a 5 × 5 grid of unknowns. The display was made with Matlab's `spy` function.

Which Matrix Patterns Preserve Sparsity?

Finding the reordering that minimizes the number of nonzeros in L is generally too expensive, so we rely on *heuristics* that can give us an inexpensive algorithm to find a reordering but that aren't guaranteed to produce an optimal ordering. Usually the heuristics do well, but sometimes they produce a very bad reordering. The heuristics all aim to permute the matrix to a form that has little *fill-in* (new nonzeros produced in the factorization). As a guide to designing heuristic strategies, let's investigate nonzero patterns for which sparsity is preserved.

PROBLEM 2.

a. A matrix A is a *band matrix* with bandwidth ℓ if $a_{jk} = 0$ whenever $|j - k| > \ell$. (An important special case is that of a *tridiagonal matrix*, with $\ell = 1$.) Show that the factor L for A also has bandwidth ℓ.

b. Define a matrix's profile to stretch from the first nonzero in each column to the main diagonal element in the column, and from the first nonzero in each row to the main diagonal element. For example, if

$$A = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & \times \end{bmatrix},$$

then the profile of A contains its nonzeros as well as those zeros marked with ⊗:

$$\text{profile}(A) = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & \otimes & 0 & \times & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ \times & \otimes & \otimes & \times & \otimes & 0 \\ 0 & 0 & \times & \otimes & \times & 0 \\ 0 & \times & \otimes & \otimes & \otimes & \times \end{bmatrix}.$$

Show that the factor L for a symmetric matrix A has no nonzeros outside the profile of A.

From this problem, we can conclude that a good reordering strategy might try to produce a reordered matrix with a small bandwidth or a small profile.

Representing the Sparsity Structure

We can encode a matrix's sparsity in a graph. For example, a symmetric matrix

$$A = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & \times \\ \times & 0 & \times & 0 & \times & 0 \\ 0 & 0 & 0 & \times & 0 & 0 \\ 0 & \times & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & \times \end{bmatrix}$$

has upper-triangular, nonzero, off-diagonal elements a_{13}, a_{25}, a_{26}, and a_{35} and corresponds to a graph with six nodes, numbered 1 to 6, and four edges that connect nodes (1,3), (2,5), (2,6), and (3,5). (We omit the edges corresponding to the

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

main diagonal elements because these elements are always nonzero for a positive definite matrix.)

In Figure 1, we draw the graph for the matrix S given in Figure 2, corresponding to the finite difference matrix for Poisson's equation on the unit square and discretized with a 5×5 grid of unknowns. Notice that the degree of any node (the number of edges it has) is at most four, and that there are four nodes (nodes 1, 5, 21, and 25) of minimum degree, which is two.

Some Reordering Strategies

We now have the jargon and concepts necessary to review some reordering strategies.

Strategy 1: Cuthill-McKee

One of the oldest strategies is *Cuthill-McKee*, which uses the graph to order the rows and columns:

- Find a node with minimum degree and order it first.
- Until all nodes are ordered
  - —For each node that was ordered in the previous step, order all the unordered nodes connected to it, in order of their degree.

Reverse Cuthill-McKee (doing a final reordering from last to first) often works even better. Figure 3 shows the result of the ordering on matrix S. The ordering tends to give a matrix with small bandwidth.

Strategy 2: Minimum Degree

- Until all nodes are ordered
  - —Choose a node that has minimum degree, and order that node next, removing it from the graph. (If there is a tie, we choose any of the candidates.)

This strategy works rather well in practice, but it's relatively expensive because the degree counts are updated every time a node is deleted. In Figure 4, we see the results on our matrix S: the reordering gives a small profile, but not small bandwidth.

Strategy 3: Nested Dissection

- Try to break the graph into two pieces plus a *separator*, with
  - —approximately the same number of nodes in the two pieces,
  - —no edges between the two pieces, and
  - —a small number of nodes in the separator.

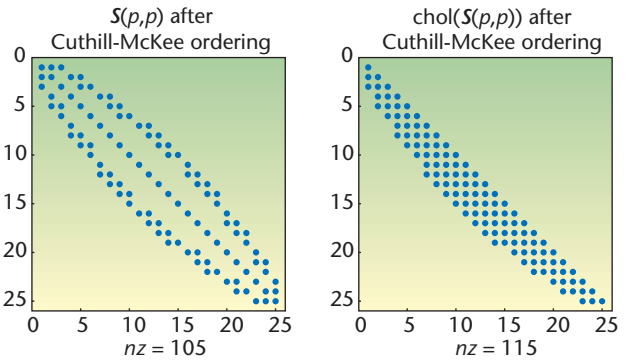

Figure 3. Sparsity pattern after reordering by using reverse Cuthill-McKee.

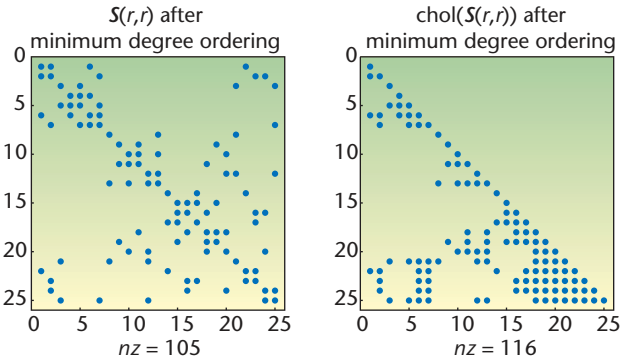

Figure 4. Sparsity pattern after reordering using the minimum degree algorithm.

- Do this recursively until all pieces have a small number of nodes.
- Then order the nodes piece by piece
- Finally, order the nodes in the separators.

For our example, if we bisect the 5×5 grid graph vertically and then bisect the remaining two pieces horizontally, this produces the following renumbering:

$$\begin{matrix} 1 & 2 & 17 & 5 & 6 \\ 3 & 4 & 18 & 7 & 8 \\ 22 & 23 & 19 & 24 & 25 \\ 9 & 10 & 20 & 13 & 14 \\ 11 & 12 & 21 & 15 & 16 \end{matrix}.$$

Figure 5 shows the results. The matrix looks quite disordered, but the number of nonzeros in the factor is smaller than for our original ordering because the profile is small.

In the next problem, we construct the graph for a sparse matrix and try these three reorderings on it.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

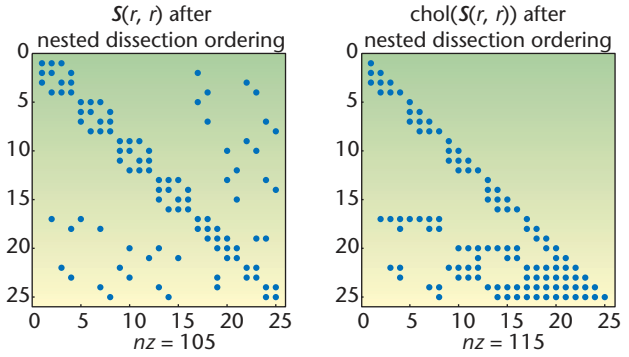

Figure 5. Sparsity pattern after reordering using nested dissection.

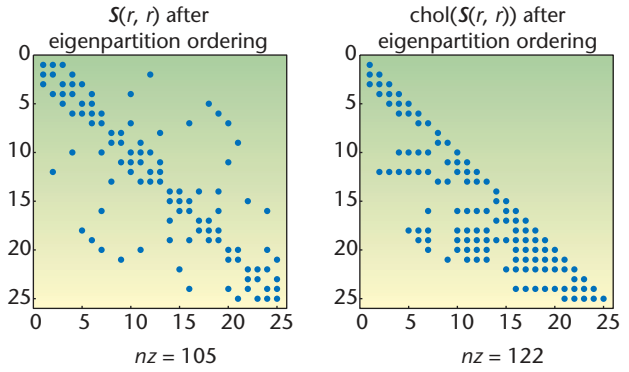

Figure 6. Sparsity pattern after reordering using eigenpartitioning.

PROBLEM 3.

Draw the graph corresponding to the matrix

$$\begin{bmatrix} \times & 0 & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & 0 & \times & 0 & \times \\ 0 & 0 & \times & 0 & \times & 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & \times & 0 & \times & \times & 0 & \times & \times \\ \times & 0 & \times & 0 & \times & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 & \times & \times & 0 & 0 & \times \\ 0 & 0 & 0 & \times & 0 & \times & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & \times & \times & 0 & 0 & \times & 0 & \times & 0 \\ 0 & \times & 0 & \times & 0 & \times & 0 & 0 & 0 & \times \end{bmatrix}$$

Try each of the three reorderings on this matrix. Compare the sparsity of the Cholesky factors of the reordered matrices with the sparsity of the factor corresponding to the original ordering.

Strategy 4: Eigenvector Partitioning

- First, form an auxiliary matrix—the *Laplacian* of the graph corresponding to our sparse matrix. This matrix $\boldsymbol{B}$ has the same size and sparsity pattern as $\boldsymbol{A}$. It has –1 in place of each nonzero off-diagonal element of $\boldsymbol{A}$, and the main diagonal elements of $\boldsymbol{B}$ are set so that each of the row sums is zero.
- The matrix $\boldsymbol{B}$ is symmetric and has no negative eigenvalues. It has a zero eigenvalue (since $\boldsymbol{Be} = \mathbf{0}$, where $\mathbf{e}$ is the vector of all ones). We compute the eigenvector $\mathbf{v}$ corresponding to its next smallest eigenvalue.
- Partition the graph into two pieces—one corresponding to nodes with positive entries in $\mathbf{v}$, and the other containing the remaining nodes.
- If desired, repeat the algorithm recursively on each of the two subgraphs formed by this partition.

This method is less intuitive and much more expensive, but it produces useful orderings. Unlike the previous strategies, we can't determine the ordering by hand computation because it involves an eigenvector computation. Because of the expense compared to the other strategies, you should probably use it only on matrices you plan to use multiple times.

The eigenvector computation is accomplished, for example, by asking the Lanczos algorithm to produce approximations to the two smallest eigenvalues and their eigenvectors. Figure 6 shows the results on our example. The matrix again looks quite disordered, but the profile remains rather small. This algorithm isn't very effective on this matrix.

Results on bigger problems show trends more clearly. Instead of a 5×5 grid, let's consider a 50×50 one, which gives a matrix of size $n = 2{,}500$. Table 1 summarizes our results. For this very regular graph, minimum degree works the best.

Now that we have our candidate algorithms, we can evaluate them using our test problems.

PROBLEM 4.

Use `slit2.m` and `laplace3d.m`, found on the Web site (www.computer.org/cise/homework/), to generate three linear systems (with $n = 15$). Solve the linear systems using as many of these algorithms as possible:

- Cholesky on the original matrix.
- Cholesky using the reverse Cuthill-McKee ordering.
- Cholesky using the (approximate) minimum degree ordering.
- Cholesky using the nested dissection ordering.
- Cholesky using the eigenvector ordering.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

Tools

Matlab's `symrcm` implements the reverse Cuthill-McKee ordering. The Matlab program's `symamd` is an approximate minimum degree permutation; `symmmd` is exact but more expensive. Nested dissection and the eigenvector orderings aren't built-in, but Matlab's `eigs` can be used for the eigenvector computation. (If `eigs` complains about a singular matrix, send it the Laplacian plus a multiple of the identity; this shifts the eigenvalues but preserves the eigenvectors.) These two orderings are also available in the Mesh Partitioning and Graph Separator Toolbox written by John Gilbert and Shang-Hua Teng (www.cerfacs.fr/algor/Softs/MESHPART/).

For more information on sparse matrices, reordering strategies, and graph representation, see the book by Alan George and Joseph Liu discussing the symmetric positive definite case.[1]

I.S. Duff and colleagues discuss the general case, including the complications added by stability considerations.[2] A. Pothen, H. Simon, and K.-P. Liou discuss the eigenvector partitioning method;[3] James Demmel gives an intuitive approach to the topic (www.cs.berkeley.edu/~demmel/cs267/lecture20/lecture20.html).

References

1. A. George and J.W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, 1981.
2. I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Oxford Press, 1986.
3. A. Pothen, H. Simon, and K.-P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Analysis and Applications*, vol. 11, no. 3, 1990, pp. 430–452.

Table 1. Summary of different reordering strategies.

Ordering	Nonzeros in L
Original	274,689
Reverse Cuthill-McKee	189,345
Minimum degree	68,828
Nested dissection	89,733
Eigenpartition	86,639

Make a table reporting, for each method,

- time to solve the system (include reordering, factorization, and forward and back substitution);
- storage for the matrix factors; and
- the final relative residual $\|b - Ax_{computed}\|_2 / \|b\|_2$. (These should all be well below the errors due to discretization, so they won't be a factor in your recommendation.)

If possible, run larger problems, too.

Considering the 2D and 3D problems separately, report to the CEO of PoissonIsUs.com the performance of the various methods and your recommendation for what ordering to use.

When problems get large enough to barely fit in memory, even a good reordering strategy isn't enough to let us keep the Cholesky factor in memory, and *iterative methods* must be considered as an alternative. We'll discuss them in a later homework assignment.

If you need to solve a linear system involving a matrix that is symmetric but not positive definite, or is nonsymmetric, then reordering for the stability of the factorization must take priority over reordering for sparsity. See the "Tools" sidebar for more information.

Tried any new gadgets lately?

Any products your peers should know about? Write a review for *IEEE Pervasive Computing*, and tell us why you were impressed. Our New Products department features reviews of the latest components, devices, tools, and other ubiquitous computing gadgets on the market.

Send your reviews and recommendations to pvcproducts@computer.org today!

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

PARTIAL SOLUTION TO LAST ISSUE'S HOMEWORK ASSIGNMENT

EIGENVALUES: VALUABLE PRINCIPLES

By Dianne P. O'Leary

IN THE LAST ISSUE, WE STUDIED EIGENVALUE PROBLEMS ARISING FROM PARTIAL DIFFERENTIAL EQUATIONS. EIGENVALUES HELP US SOLVE DIFFERENTIAL EQUATIONS ANALYTICALLY, BUT they also provide valuable information about physical systems' behaviors.

PROBLEM 1.

Define the domain $\Omega = (0, b) \times (0, b)$. Consider the elliptic partial differential equation

$$-u_{xx} - u_{yy} = \lambda u$$

for $(x, y) \in \Omega$, with $u(x, y) = 0$ on the boundary of Ω.

Show that the function

$$w_{m\ell}(x, y) = \sin(m\pi x/b)\sin(\ell\pi y/b),$$

where m and ℓ are positive integers, satisfies this equation. Determine the corresponding eigenvalue $\lambda_{m\ell}$.

Answer:

We can verify by direct computation that $-\partial^2 w_{m\ell}/\partial x^2 - \partial^2 w_{m\ell}/\partial y^2$ is $(m^2 + \ell^2)\pi^2/b^2 \times w_{m\ell}$, so $\lambda_{m\ell} = (m^2 + \ell^2)\pi^2/b^2$.

PROBLEM 2.

In this problem, we study the elliptic eigenvalue problem $-\nabla \cdot (\nabla u) = \lambda u$ on the square $(-1, 1) \times (-1, 1)$ with zero boundary conditions. We know the true eigenvalues, so we can determine how well the discrete approximation performs.

a. Form a finite difference or finite element approximation to the problem and find the eigenfunctions corresponding to the five smallest eigenvalues.

- Describe in words the shape of each of these eigenfunctions. How does the shape change as the eigenvalue increases?
- Theory tells us that we have good approximations with a coarse grid only for the eigenfunctions corresponding to the smallest eigenvalues. How does the shape of the eigenfunctions make this result easier to understand?

b. Create five plots—for eigenvalues 1, 6, 11, 16, and 21—of the error in the approximate eigenvalue versus $1/h^2$. (Use at least four different matrix sizes, with the finest $h < 1/50$.) Discuss:

- What convergence rate do you observe for each eigenvalue?
- How does it compare with the theoretical convergence rate? (Explain any discrepancy.)
- Are all the eigenvalues well-approximated by coarse meshes?

Answer:

a. The eigenvalues are

$$\lambda_{jk} = \frac{j^2 + k^2}{4}\pi^2$$

for $j, k = 1, 2, \ldots$. One expression for the eigenfunction is $v_{jk} = \sin(j\pi(x + 1)/2)\sin(k\pi(y + 1)/2)$.

This isn't unique: some of the eigenvalues are multiple—so, for example, $\lambda_{12} = \lambda_{21}$, and any function $av_{12} + bv_{21}$ for arbitrary scalars a and b is an eigenfunction. Even for simple eigenvalues, we can multiply the function v_{jj} by an arbitrary constant, positive or negative.

Figure A plots the first six v_{jk}. Note that as the eigenvalue increases, so does the number of oscillations in the eigenfunction. To capture this behavior in a piecewise linear approximation, we need a finer mesh for the eigenfunctions corresponding to larger eigenvalues than we do for those corresponding to smaller eigenvalues.

b. When using piecewise linear finite elements, the jth computed eigenvalue lies in an interval $[\lambda_j, \lambda_j + C_j h^2]$, where h is the mesh size used in the triangulation. This is observed in our computation using the code `problem1b.m`, found on the Web site (www.computer.org/cise/homework). Figure B shows the error plots. The horizontal axis is the number of triangles, which is approximately proportional to $1/h^2$.

The errors in the approximate eigenvalues are as follows:

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

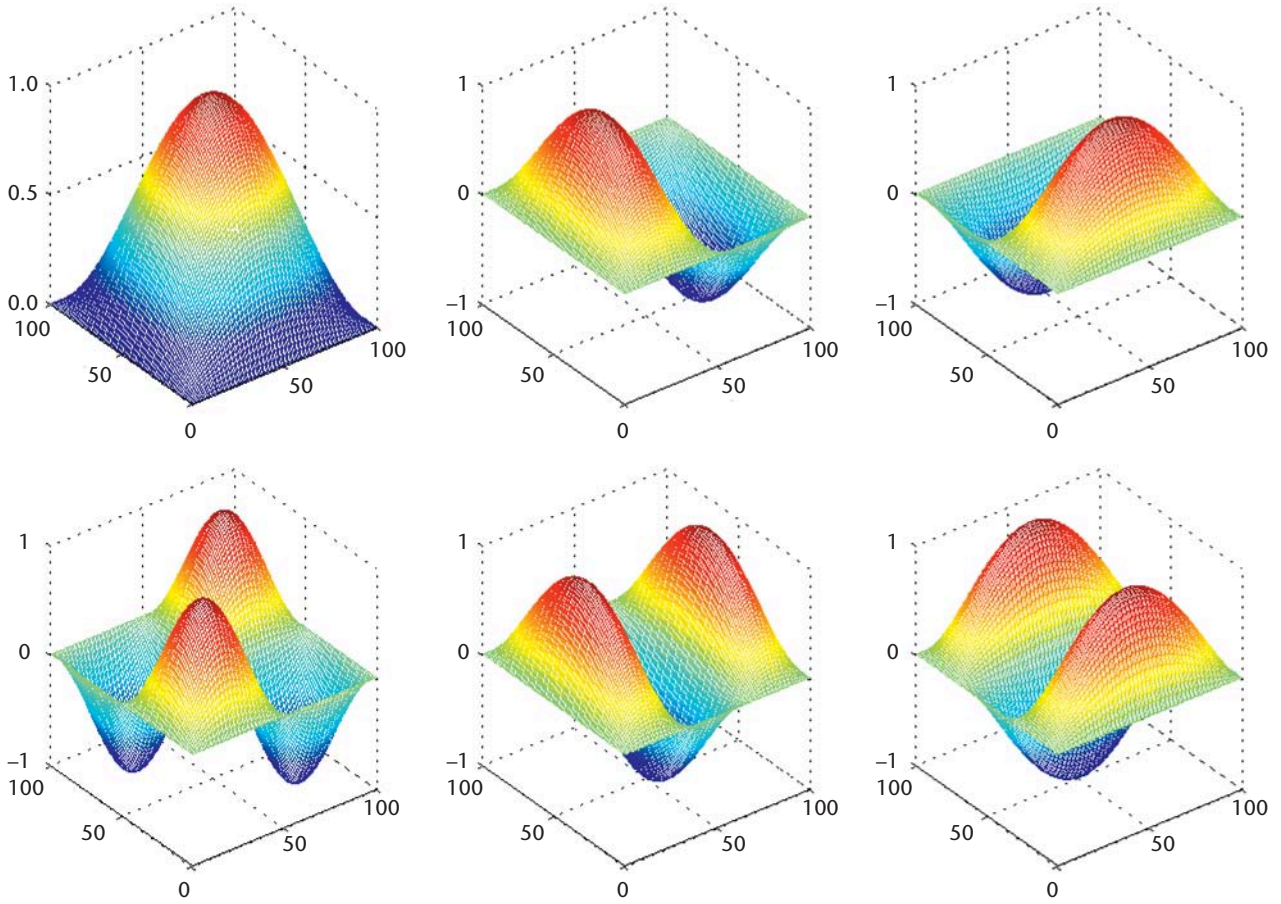

Figure A. The eigenfunctions corresponding to the eigenvalues $\lambda = 4.9348$, 12.3370, and 12.3370 (top row) and $\lambda = 19.7392$, 24.6740, and 24.6740 (bottom row).

Lambda	Mesh1	Mesh 2	Mesh 3	Mesh 4
1	5.03e-02	1.27e-02	3.20e-03	8.02e-04
6	1.29e+00	3.25e-01	8.15e-02	2.04e-02
11	4.17e+00	1.04e+00	2.60e-01	6.50e-02
16	8.54e+00	2.12e+00	5.28e-01	1.32e-01
21	1.49e+01	3.67e+00	9.15e-01	2.29e-01

The error ratios are as follows:

Lambda	Mesh 1vs2	Mesh 2vs3	Mesh 3vs4
1	3.95e+00	3.98e+00	3.99e+00
6	3.98e+00	3.98e+00	3.99e+00
11	4.02e+00	4.00e+00	4.00e+00
16	4.04e+00	4.01e+00	4.00e+00
21	4.05e+00	4.01e+00	4.00e+00

Therefore, the error is reduced by a factor of 4 as the side of each triangle is reduced by a factor of 2, so the error is $O(h^2)$, as expected, but the larger the eigenvalue, the finer the mesh necessary to achieve a given accuracy.

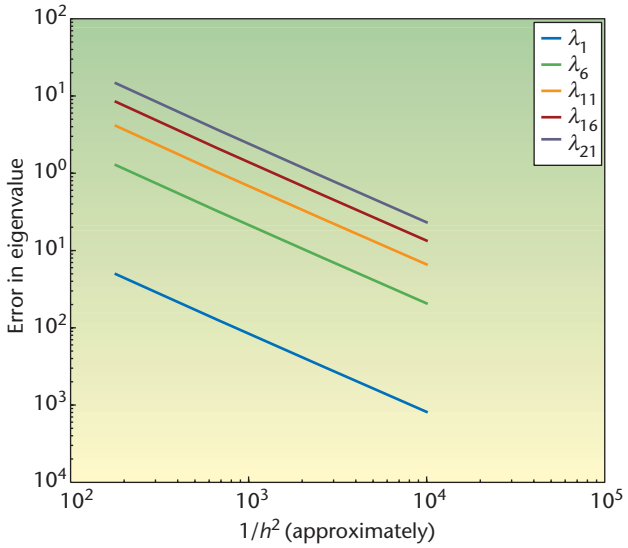

Figure B. The errors in the eigenvalue approximations as a function of $1/h^2$; the horizontal axis is the number of triangles.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.

PROBLEM 3.

a. Suppose $\mathcal{A}w = \lambda w$ in Ω where

$$\mathcal{A}w = -\nabla \cdot (a\nabla w),$$

and $a > 0$. Prove that $\lambda_1 > 0$. Hint: use integration by parts to replace $(w, \mathcal{A}w)$ with $\int_\Omega a(x)\nabla w(x) \cdot \nabla w(x)dx$.

b. Suppose we have two domains $\Omega \subseteq \tilde{\Omega}$. Prove that $\lambda_1(\Omega) \geq \lambda_1(\tilde{\Omega})$. Hint: Notice that the eigenfunction for Ω can be extended to be a candidate for the minimization problem for $\tilde{\Omega}$.

How to Reach *CiSE*

Writers

For detailed information on submitting articles, write to cise@computer.org or visit www.computer.org/cise/author.htm.

Letters to the Editors

Send letters to Jenny Ferrero, Contact Editor, jferrero@computer.org. Provide an email address or daytime phone number with your letter.

On the Web

Access www.computer.org/cise/ or http://cise.aip.org for information about CiSE.

Subscribe

Visit https://www.aip.org/forms/journal_catalog/order_form_fs.html or www.computer.org/subscribe/.

Subscription Change of Address (IEEE/CS)

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify CiSE.

Subscription Change of Address (AIP)

Send general subscription and refund inquiries to subs@aip.org.

Missing or Damaged Copies

Contact membership@computer.org. For AIP subscribers, contact kgentili@aip.org.

Reprints of Articles

For price information or to order reprints, send email to cise@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.

www.computer.org/cise/

Answer:

a. Suppose (for convenience of notation) that $\Omega \subset \mathcal{R}^2$. (Other dimensions are just as easy.) First, we apply integration by parts (with zero boundary conditions) to see that if $w \neq 0$

$$\begin{aligned}(w, \mathcal{A}w) &= -\iint_\Omega w\nabla \cdot (a\nabla w)dx\ dy \\ &= \iint_\Omega \nabla w \cdot (a\nabla w)dx\ dy \\ &= \iint_\Omega a_1(x,y)\left(\frac{\partial w}{\partial x}\right)^2 + a_2(x,y)\left(\frac{\partial w}{\partial y}\right)^2 dx\ dy \geq 0\end{aligned}$$

because $a_1(x)$, $a_2(x) > 0$. Suppose $\mathcal{A}w = \lambda w$. Then, $0 \leq (w, \mathcal{A}w) = \lambda(w, w)$, so $\lambda \geq 0$.

b. We know that

$$\lambda_1(\Omega) = \min_{w \neq 0} \frac{(w, \mathcal{A}w)}{(w, w)},$$

where the integrals are taken over Ω and w is constrained to be zero on the boundary of Ω. Suppose the w that minimizes the function is $\tilde{w}$ and let's extend $\tilde{w}$ to make it zero over the part of $\tilde{\Omega}$ not contained in Ω. Then,

$$\lambda_1(\tilde{\Omega}) = \min_{w \neq 0} \frac{(w, \mathcal{A}w)}{(w, w)} \leq \frac{(\tilde{w}, \mathcal{A}\tilde{w})}{(\tilde{w}, \tilde{w})} = \lambda_1(\Omega).$$

PROBLEM 4.

Determine the dimension of a square drum that has a fundamental frequency equal to 1 when $c = 1$. Use numerical methods to find an elliptical domain $\alpha x^2 + 2\alpha y^2 < 1$ with the same fundamental frequency.

Answer:

From Problem 1, we know that the smallest eigenvalue for a square with dimension b is $2\pi^2/b$, so we want $b = \sqrt{2}/2$. Using Matlab's `Pdetoolbox` interactively, we discover that $\alpha \approx 1.663$.

Dianne P. O'Leary is a professor of computer science and a faculty member in the Institute for Advanced Computer Studies and the Applied Mathematics Program at the University of Maryland. She has a BS in mathematics from Purdue University and a PhD in computer science from Stanford. O'Leary is a member of SIAM, the ACM, and AWM. Contact her at oleary@cs.umd.edu; www.cs.umd.edu/users/oleary/.

Authorized licensed use limited to: to IEEExplore provided by Virginia Tech Libraries. Downloaded on October 24, 2008 at 09:46 from IEEE Xplore. Restrictions apply.