

1.17. An alternative way to keep up with the row switches in DLINEQ is to simply set  $IPERM(I) = L$  after loop 15, to remember which row was switched with row  $I$ , and then in DRESLV, instead of permuting the elements of  $C$  at the beginning, switch  $C(I)$  and  $C(L)$  ( $L = IPERM(I)$ ) immediately before loop 15, replicating the switches done to  $B$  in DLINEQ. We now must only switch the “nonzero” portions of rows  $I$  and  $L$  of  $A$  in DLINEQ, and not the multipliers saved in the previous columns, that is, loop 20 should only run from  $K = I$  to  $N$ . We also need to remove the code that switches  $IPERM(I)$  and  $IPERM(L)$ . The new approach has the advantage that if  $A$  is a band matrix and pivoting is done, you can save the multipliers without fill-in below the band in the lower triangle. However, note that the lower triangle of  $A$  returned by the new DLINEQ is no longer equal to the lower triangle of the  $L$  matrix of the  $LU$  decomposition (of  $PA$ ).

- a. Make the changes to DLINEQ and DRESLV suggested above and solve  $A\mathbf{x} = \mathbf{b}$  using the new DLINEQ, where  $A$  is a 6 by 6 tridiagonal matrix with 1 in each main diagonal position and 2 in each sub- and super-diagonal, and  $\mathbf{b}$  is chosen so that the exact solution is  $(1, 1, 1, 1, 1, 1)$ . Then also solve  $A\mathbf{x} = 2\mathbf{b}$  using the new DRESLV. Print out the matrix  $A$  after calling the modified DLINEQ, and note that there is no fill-in below the band. Repeat this problem using the original versions of DLINEQ and DRESLV, and you will see that the saved multipliers do cause fill-in below the band.
- b. Make two copies of the band solver DBAND, call one DBLINEQ and the other DBRESLV; each should have an additional argument  $IPERM$ . Set  $A(J, I - J) = LJI$  immediately before the end of loop 30 in DBLINEQ so that the multiplier used to zero each element is saved in the same location, as done in DLINEQ, and set  $IPERM(I) = L$  after loop 15 to keep track of the row switches, as done in part (a). Now modify DBRESLV so that it solves another system with the same matrix, using  $A$  and  $IPERM$  as output by DBLINEQ. DBRESLV should set  $L = IPERM(I)$  before  $B(L)$  and  $B(I)$  are switched, and set  $LJI = A(J, I - J)$  in loop 30 to retrieve the multiplier. All code that changes elements of  $A$  should be eliminated from DBRESLV, and code that checks for zero pivots can also be removed.

Test DBLINEQ and DBRESLV using the same band matrix as in part (a), and solve the same two systems.

DBLINEQ will do the same number of multiplications as DBAND (about  $N N_{LD}(N_{UD} + N_{LD})$ ), to solve the first system. Approximately how many multiplications will DBRESLV do to solve each

additional system?

- 3.17g. If  $D$  is an  $M$  by  $N$  diagonal matrix, its *pseudoinverse*  $D^+$  is found by taking its transpose, then inverting the *nonzero* elements on the diagonal. The pseudoinverse of a general  $M$  by  $N$  matrix is defined by  $A^+ = VD^+U^T$ , where  $A = UDV^T$  is a singular value decomposition of  $A$ . Note that if  $D$  is a square nonsingular matrix,  $D^+ = D^{-1}$  and also  $A^+ = A^{-1}$ . Show that  $A^T A(A^+ \mathbf{b}) = A^T \mathbf{b}$  for any vector  $\mathbf{b}$ , thus  $\mathbf{x} = A^+ \mathbf{b}$  is a solution of the normal equations, and therefore a solution to the least squares problem,  $\min \|A\mathbf{x} - \mathbf{b}\|_2$ . (Hint: first show that  $D^T(DD^+ - I)$  is the zero matrix.)
- h. Show that  $\mathbf{x} = A^+ \mathbf{b}$  (see Problem 17g) is the minimum norm least squares solution. (Hint: if  $\mathbf{x} + \mathbf{e}$  is any other solution of the normal equations,  $A^T A\mathbf{e} = \mathbf{0}$ ; show that this implies the first  $K$  components of  $V^T \mathbf{e}$  are zero, and that this implies that  $(A^+)^T \mathbf{e} = \mathbf{0}$ . Then it is easy to show that  $\|\mathbf{x} + \mathbf{e}\|_2^2 = \|\mathbf{x}\|_2^2 + \|\mathbf{e}\|_2^2$ .)
- i. Suppose we want to find the general solution to  $A\mathbf{x} = \mathbf{0}$ , and we have a singular value decomposition of  $A = UDV^T$ . Since  $U$  is nonsingular,  $A\mathbf{x} = \mathbf{0}$  means  $DV^T \mathbf{x} = \mathbf{0}$ , or  $D\mathbf{y} = \mathbf{0}$ , where  $\mathbf{x} = V\mathbf{y}$ . The components  $y_i$  of  $\mathbf{y}$  corresponding to zero columns (if any) of  $D$  can thus have arbitrary values, the other components must be 0. Then what will be the general solution of  $A\mathbf{x} = \mathbf{0}$ ?
- 4.9b. Solve the resource allocation problem 4.4.1 using an interior point method, as follows. Replace the equations  $x_i z_i = 0$  by  $x_i z_i = \mu$ , and solve the system of Problem 9a using Newton's method, with  $\mu = 1$ , and  $x_i = z_i = 1$  ( $y_i = 0$ ) initially. Then re-solve this system repeatedly, with  $\mu$  cut in half each time, using the solution from the previous problem to start the Newton iteration for each new  $\mu$ .  $x_i$  and  $z_i$  should remain positive as  $\mu$  goes to 0.

For this system, Newton's method has the form:

$$\begin{bmatrix} \mathbf{x}^{k+1} \\ \mathbf{y}^{k+1} \\ \mathbf{z}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \mathbf{z}^k \end{bmatrix} - \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & -I \\ Z^k & 0 & X^k \end{bmatrix}^{-1} \begin{bmatrix} A\mathbf{x}^k - \mathbf{b} \\ A^T \mathbf{y}^k - \mathbf{z}^k - \mathbf{c} \\ X^k Z^k \mathbf{1} - \mu \mathbf{1} \end{bmatrix}$$

where  $X^k$  and  $Z^k$  are diagonal matrices with the vectors  $\mathbf{x}^k$  and  $\mathbf{z}^k$ , respectively, along the diagonals.

- 5.6c. A function  $u(x, y)$  defined in  $(0, 1) \times (0, 1)$  and extended periodically, can be expanded in a Fourier series:

$$u(x, y) = \sum_{M=-\infty}^{\infty} \sum_{L=-\infty}^{\infty} b_{LM} \exp(i2\pi Lx) \exp(i2\pi My)$$

Truncate the series, letting  $L$  and  $M$  vary from  $-\frac{N}{2}$  to  $\frac{N}{2} - 1$ , and then make the changes of variables  $l = L + \frac{N}{2} + 1$ ,  $m = M + \frac{N}{2} + 1$ , to get:

$$u(x, y) \approx \sum_{m=1}^N \sum_{l=1}^N b_{lm} \exp(i2\pi(l-1-\frac{N}{2})x) \exp(i2\pi(m-1-\frac{N}{2})y)$$

and get a similar expansion for  $f(x, y)$ . Plug these expansions directly into the PDE of Problem 6a, rather than into its finite difference approximation, to find another formula for  $b_{lm}$  in terms of  $c_{lm}$ . Now if you evaluate the expansions for  $u$  and  $f$  at  $x = x_j = (j-1)/N$ ,  $y = y_k = (k-1)/N$  and simplify them, you will see that the solution values  $u(x_j, y_k)$  can again be found from the coefficients  $b_{lm}$  using DFFT, and the values  $f(x_j, y_k)$  and coefficients  $c_{lm}$  are similarly related. Rerun your program from Problem 6b with these minor changes, and get another estimate of  $u(0.5, 0.5)$ . In part (b) you were using the FFT simply to rapidly solve the finite difference equations exactly; now you are using a Fourier series method, with a truncated series. Is your answer more or less accurate (at the midpoint) than in part (b)?

Answer: When  $N = 2^9$ ,  $U(0.5, 0.5) = -0.262116$ .

- 5.6d. Write a program which solves the 3D PDE  $u_{xx} + u_{yy} + u_{zz} - u_x - 2u = f(x, y, z)$ , with periodic boundary conditions on  $0 \leq x \leq 2\pi$ ,  $0 \leq y \leq 2\pi$ ,  $0 \leq z \leq 2\pi$ , using centered finite differences on an  $N$  by  $N$  by  $N$  grid, where  $N = 2^8$ , and fast Fourier transforms to solve the  $N^3$  by  $N^3$  linear system. You should generalize the procedure for 2D problems outlined in Problem 5. Choose  $f(x, y, z)$  so that the exact solution is  $u = \cos(x - y + 2z)$ , and calculate the maximum error over the grid points.

Answer: When  $N = 2^8$ , maximum error =  $1.12 * 10^{-4}$ .

- 6.6. Repeat Problem 3(a), only this time use a parallelized version, PREDH, of REDH (Figure 2.3.1) instead of REDQ. Again, distribute the columns of  $A$  cyclically over the available processors. In PREDH, whoever has the active column  $L$  should call CALW to compute the vector  $W$ , and broadcast  $W$  to the other processors.  $A(I, L)$  will also need to be broadcast to the other processors after it changes.
- 6.7. Write a parallel version, PHESSH, of the subroutine HESSH (Figure 3.4.2) which reduces a matrix to a similar upper Hessenberg matrix using Householder transformations. Each processor should store the entire matrix, but never touch any columns but its "own" ( $ITASK +$

$1 + K * NPES, K = 0, 1, 2, \dots$ ), until the end. The processor which owns column  $I - 1$  should call `CALW`, and then broadcast the vector  $W$  to all processors, since all of them need  $W$ . At the end of `PHESSH`, before the QR iteration begins, the entire Hessenberg matrix should be redistributed to all processors, using `MPLBCAST`.

Test `PHESSH` by generating a random 1000 by 1000 symmetric matrix, which is passed to `DEGNON` (Figure 3.3.4), modified to call `PHESSH` instead of `HESSQ`. Since the Hessenberg matrix output by `PHESSH` will still be symmetric, and thus tridiagonal, modify QR as indicated in the comments, to take advantage of this. Make runs with 1 and 4 processors. Since you have only parallelized the reduction to Hessenberg form, and not the QR iteration, the speed-up observed should be substantially less than 4. Explain why it would be more difficult to parallelize QR (or `HESSQ`) efficiently, than `HESSH`.

- 6.8. Write a parallel version, `PLPRV`, of the revised simplex method routine `DLPRV` (Figure 4.6.1). Each processor should store the entire `ABINV` matrix, but never touch any columns but its own ( $ITASK + 1 + K * NPES, K = 0, 1, 2, \dots$ ). Test `PLPRV` by calling it from `DTRAN`, with `DTRAN` used to solve a transportation problem with  $NS = 100$  stores and  $NW = 150$  warehouses, with the `SREQ`, `WCAP` and `COST` arrays generated by a random number generator. Make runs with 1 and 4 processors.